مفاهيم البرمجة بين القديم والحديث

Programming Concepts

مهندس

عزب محمد عزب

استشارى البرمجة والتطوير

OCP, MCP

دار طيبة للنشر والتوزيع والتجهيزات العلمية

23 شارع الفريق محمد إبراهيم - مكرم عبيد - مدينة نصر - القاهرة

تليفون : 00202 - 22725312 - 22725376 - 26706912

فاكس 00202 - 26706912

الترقيم الدولي 977-6102-46-8
رقم الإيداع 2008/26584

مفاهيم البرمجة بين القديم والحديث

تأليف: عزب محمد عزب

© حقوق النشر والتوزيع محفوظة لدار طيبة للنشر والتوزيع والتجهيزات العلمية – 2008
23 شارع الفريق محمد ابراهيم - متفرع من مكرم عبيد – مدينة نصر القاهرة ج.م.ع
تليفون : 22725312 -22725376- 26706912 (02)
فاكس : 26706912(02)

لا يجوز نشر أي جزء من الكتاب أو أعادة طبعه أو اختصاره بقصد الطباعة أو اختزان مادته العلمية أو نقله بأي طريقة سواء كانت الكترونية أو ميكانيكية أو بالتصوير أو خلاف ذلك دون موافقة كتابية من الناشر مقدماً .

بسم الله الرحمن الرحيم
إن الله هو الرزاق ذو القوة المتين

إهداء

ما رايتنى أحب بعد الله ورسوله وأصحابه

اكثر من ابى وأمى

أحبهما لانهما أبى وأمى ..... وأحبهما لأنى أرى بعينى أنهما لا يستحقا الحب فقط لذلك بل لما عليه من الطباع والصفات الطيبة وكم الهدوء والحكمة الذى رايته فى هذا الرجل البسيط الطيب العاقل الحكيم أبى .

اللهم

إنى لأرجو لأعمالى من ثواب وإنى لأرجو أن تشمل أبى بكرمك الذى لا حد له ....كرم الرب الكريم .
اللهم إنك أرحم بأبى منـــى فانى لا أسألك بحقى عليك لأنه ليس لى حق عليك ولكنى أسالك بكرمك وجودك وإحسانك أن تعامل أبى بما أنت أهله.

اللهم

اشمل أبى وأباءنا برحمتك فرحمتك وسعت كل شئ ولك الحمد فى الأول ولك الحمد فى الآخرة.

لمن هذا الكتاب ؟

1- لكل طالب يدرس مادة مفاهيم البرمجة ويحتاج الى مرجع بسيط اللغة واضح المفاهيم .

2- لكل من يهوى الدخول فى عالم البرمجة والتطوير ويريد أن يتعلم القواعد العامة للغات البرمجة .

3- لمن يستعمل أحد لغات البرمجة ولكن يحتاج مرجع للقواعد العامة للغات البرمجة .

4- لمن يجيد استعمال أحد لغات البرمجة مثل VB.NET ولكن يريد أن يتعرف على المقابل لموضوعاتها فى اللغات الأخرى .

5- لمن يستعمل أحد لغات البرمجة القديمة مثل لغة C ويحتاج الى تحديث معلوماته ومعرفة قواعد لغات البرمجة الحديثة وما طرأ من تغيير كبير فى عالم البرمجة والتطوير .

المحتويات

الجزء الأول :

الموضوع	الصفحة
• الفصل الأول : مقدمة عن البرمجة Programming Basics	11
• الفصل الثاني : القواعد العامة للغات البرمجة	43
• الفصل الثالث : البرنامج فى لغات البرمجة والمتغيرات والإدخال والإخراج	82
• الفصل الرابع : جمل التحكم في مسار البرنامج وجمل التكرار	102
• الفصل الخامس : البرامج الفرعية Subprograms (Proc,Functions)	136
• الفصل السادس : التعامل مع المصفوفات فى لغات البرمجة	161
• الفصل السابع : مفاهيم البرمجة باستعمال أسلوب OOP	183
• الفصل الثامن : تابع مفاهيم البرمجة باستعمال أسلوب OOP	214
• الفصل التاسع : تابع مفاهيم البرمجة باستعمال أسلوب OOP	235
• الفصل العاشر : التعامل مع أخطاء البرنامج Error Handling	254
• الفصل الحادى عشر : استعمال مفهوم الـ Visual Programming	286
• الفصل الثانى عشر : مقدمة عن قواعد البيانات و أوامر لغة الاستعلامات SQL	311
• الفصل الثالث عشر : مفاهيم وطرق الاتصال والتعامل مع قواعد البيانات	353
• الفصل الرابع عشر : تابع التعامل مع قواعد البيانات وفكرة عن التقارير	400
• الفصل الخامس عشر : نظرية إعادة الاستخدام Reusability والمكتبات	429
• الفصل السادس عشر : الرسم فى اللغات Graphics	467
• الفصل السابع عشر : التعامل مع الملفات النصية والتعامل مع النصوص String	484
• الفصل الثامن عشر : برمجة الشبكات Networking Programming	501
• الفصل التاسع عشر : مفهوم الـ Threading	527
• الفصل العشرون : برمجة الإنترنت Web Programming	551

الفصل الأول

مقدمة عن البرمجة
Programming Basics

فى هذا الفصل Programming Concepts 1

نعرض مراحل تطور البرمجه بأشكالها المختلفه وما وصلت اليه الأن وكذلك عرض المصطلحات الجديدة اليوم و اللغات القديمة واللغات الموجودة على الساحه وذلك من خلال النقاط التالية

- مفاهيم البرمجه بين القديم و الحديث
- مفهوم البرمجة بالأهداف OOP
- البرمجة المرئية Visual Programming
- لغات تطوير تطبيقات الإنترنت Web Developing
- قواعد البيانات Database
- أدوات التطوير حاليا
- تطبيقات الإنترنت Web Applications
- ماذا افعل لكى أكون مطور؟ Developer

مفاهيم البرمجة بين القديم و الحديث

فى بداية عصر تصنيع الكمبيوتر كانت تتم عملية البرمجة عن طريق مجموعة كبيرة من المفاتيح Switch وكان كل مفتاح يمثل معلومة صغيرة وكان القائمون على عملية البرمجة لابد وأن يكونوا من العلماء والفنيون فى الإلكترونيات والرياضيات وكانت برمجة برامج صغيرة عملية شاقة ومجهدة.

ولكن مع تطور الكومبيوتر وإنتشاره زادت الحاجة إلى برامج اكبرولغات اكثر مرونة ومن هنا زادت العملية تعقيداً.

لغات البرمجة

ومن أوائل اللغات التى ظهرت كانت لغة الأسمبلى(التجميع) Assembly ولكنها كانت لغة صعبة للغاية وكان من يتعامل معها لابد وأن يكون من الفنيين.

ثم تلى ظهور Assembly لغات تستخدم مفردات اللغة الإنجليزية مثل لغة فورتران Fortran وهى تنتمى إلى لغات المستوى العالى وأدى هذا إلى تصنيف لغات البرمجة إلى:

لغات منخفضة المستوى Low Level Language

وهى اللغات القريبة من الآلة (الكمبيوتر) مثل :

- لغة الآلة (Machine Language)

وهى عبارة عن مجموعة من الاكواد التى يفهمها الكمبيوتر ولا يفهمها المبرمج ،فقط أكواد ولكل كود وظيفة

- لغة الترميز (Assembly Language) والتى تتكون من رموز صعبه تمثل الأوامر المطلوبة لإعداد برنامج وهى أسهل من لغة الآلة Machine Language وتحتوى على أوامر الى حد ما مقروءة مثل mov وAdd وغيرها من التعليمات

لغات مرتفعة المستوى High Level Language

وهى لغات البرمجة التقليدية و التى تتكون من كلمات باللغة الإنجليزية قريبه من الكلمات العادية مثل Read, Write وغيرها ، ومن هذه اللغات :

- PASCAL, COBOL ,FORTRAN ,C,C++,Java,....

وكان المفهوم الوحيد للتطوير هو البرمجة (Programming) ، وكانت خطوات إعداد برنامج ليست سهلة وعليك إتقان اللغة ومعرفة الكثير من الأوامر والدوال لإعداد برنامج وكذلك الالتزام بكثير من القواعد المعقدة ، وقد تضطر لإعادة كتابة نفس مجموعة الأوامر أكثر من مرة عند الحاجة إليها.

فمثلا فى لغة مثل لغة BASIC القديمة الشهيرة لإعداد تطبيقات عامة عليك بكتابة كم كبير من السطور بل لمجرد تكوين شكل البرنامج من قوائم اختيارات وتغيير الألوان تكتب الكثير من الأوامر.

وكانت لغات البرمجة مقسمه حسب المجال ، فإذا كنت تريد إعداد تطبيقات تجاريه عليك تعلم لغة COBOL التى توفر التعامل مع الملفات والبيانات ،وإذا كنت تريد إعداد تطبيقات علميه عليك بتعلم لغة FORTRAN الشهيرة بالأوامر والدوال التى تخدم التطبيقات العلمية والمعادلات ، وإذا كنت تريد إعداد تطبيقات تتعامل مع النظام (system) عليك بتعلم لغة C/C++ ،وبالطبع الأمر اختلف الآن.

ومع ظهور لغات المستوى العالى High Level English لم يصبح العلماء والفنيين فقط هم الذين يقومون بالبرمجة فقط بل أصبح الكثير من الناس يستطيعوا أن يقوموا بتلك العملية لأن تلك اللغات تتعامل بمفردات اللغة الإنجليزية المفهومة.

وأدى تطور لغات البرمجة إلى ظهور الكثير من المفاهيم وكان من أهم تلك المفاهيم هو البرمجة التركيبية structured programming.

البرمجة التركيبية Structured programming

كانت لغات البرمجة قبل ظهور البرمجة التركيبية structured programming يقوم فيها المبرمجون بكتابة البرنامج كله قطعة واحدة وكان ذلك يؤدى إلى صعوبات كثيرة فى التعامل مع الكود ويؤدى أيضا إلى صعوبة فهم الكود والأخطاء الكثيرة.

ولكن البرمجة التركيبية structured programming غيرت ذلك حيث يقوم المبرمج بتقسيم البرنامج إلى أجزاء (مقاطع) ويعطى لكل جزء اسم خاص به ثم يقوم بعد ذلك باستدعاء اسم ذلك الجزء (فقط ليقوم البرنامج بتنفيذ ذلك الجزء) فى الدالة الرئيسية Main Function بحيث يقوم كل جزء بأداء مهمة محدده.

كما قلنا يتم تقسيم البرنامج إلى دالة رئيسية main function ودوال فرعية function ويتم استدعاء الدوال الفرعية داخل الدالة الرئيسية وعملية الاستدعاء تسمى Calling

ولابد أن تكون تلك الدوال مرتبة فى ترتيب منطقى مترابط .

إذن نستطيع أن نخرج من المناقشة السابقة بأن البرنامج فى structured programming هو مجموعة من الدوال Functions. والدالة تتكون من مجموعة من البيانات والأوامر.

ثم مع تطور حركة البرمجة وزيادة التعقيد فى البرامج كان من اللازم تطوير المفاهيم مثل Structured Programming . وجاء ذلك التطوير فى شكل جديد وهوالبرمجة بالأهداف Object Oriented Programming أو إختصاراً OOP وتقيدنا البرمجة بالأهداف OOP فى أننا نستطيع تمثيل الأشياء المحيطة بنا تمثيل حقيقي وهو ما سيتضح خلال الفقرة القادمة.

اللغات التقليدية Traditional

أطلق على لغات هذا الجيل باللغات التقليدية (Traditional) حيث تتصف ببعض المواصفات منها:

تبنى على الدوال Functions والأوامر statments:

فكانت اللغة عبارة عن مجموعة من الأوامر والدوال التى تلتزم بها لإعداد برنامج ، وعليك استخدامها لبناء البرامج من البداية حتى النهاية.

برامج رئيسية و برامج فرعية Structure programming:

كانت هذه هى فكرة معظم اللغات مثل لغة c ، ولغة PASCAL حيث يحتوى البرنامج على دالة رئيسية (برنامج رئيسى) يستدعى دوال فرعيه .

الدوال أولا ثم البيانات : وهى فكرة جميع اللغات التقليدية حيث يحدد الدوال المطلوبة وداخلها يحدد البيانات المطلوبة لهذه الدوال .

والسطور التالية تعرض نص برنامج بلغة C يوضح شكل البرنامج فى هذه الفترة (اللغات التقليدية)

```
#include <stdio.h>
main()
{
int a,b,c;
a=5;
b=10;
c=a+b;
printf("\n c=%d",c);
}
```

مفهوم البرمجة بالأهداف OOP

ظهر بعد ذلك مفهوم جديد هو البرمجة بواسطة الأهداف (object oriented programming) ، وهى جديدة تحاول توفير الوقت فى إعداد البرامج حيث أنها تتصف ببعض الصفات الجيدة منها :-

إن اللغات تبنى على الفصائل (classes) ، وهى وحدة بناء البرامج بدلا من الدوال وكذلك خاصية التوريث Inheritance ، حيث أن ظهور فكرة البرمجة بالأهداف (oop) قد أدخل مفهوم جديد وهو بناء البرامج باستعمال أسلوب جديد هو الفصائل (classes) وليس الدوال ، والفصيلة تحتوى على دوال وبيانات ، وبالتالى اجتمعت وحدة بناء البرنامج وهذا يوفر الكثير ، فبدلا من استعمال دالة لرسم إطار ، ودالة لتغيير لون خلفية والكثير من الدوال المطلوبة والبيانات لبناء نافذة (Windows) يمكنك فى البرمجة بواسطة الأهداف استعمال فصيلة لرسم نافذة بكل عناصرها وتغيير خصائصها بسهولة.

ولتوضيح هذه الفكرة :

تخيل لو انك قمت بكتابة أوامر لإعداد زر الأمر الشهير (BUTTON) الموجود فى برنامج النوافذ سوف تحتاج إلى الكثير من الأوامر وذلك باستعمال اللغات التقليدية ، إما باستعمال البرمجة بالأهداف (OOP) ومع وجود مكتبة فصائل فليس عليك سوى استعمال الفصيلة (BUTTON) لتعريف زر أمر ثم التحكم فى خصائصه .

كما وفرت البرمجة بالأهداف oop الكثير من الخصائص التي تخدم البرمجة مثل خاصية التوريث (inheritance) التي تسهل عملية بناء الفصائل وذلك بتوريث فصيلة قديمه لفصيلة جديدة مما يسهل بناء مكتبة فصائل ، والكثير من الخصائص الأخرى ، وقام على هذا المفهوم بعض اللغات مثل JAVA , C++ .

البرمجة المرئية Visual programming

بعد تقديم مفهوم البرمجة بالأهداف oop تم تقديم مفهوم آخر يعتبر امتداد له وهو البرمجة المرئية وهي فكره جيدة توفر عليك كتابة أوامر لاستعمال فصيلة معينة ، فمثلا بدلا من كتابة أوامر لاستعمال الفصيلة button وتعريف button لاستعمالها يمكنك سحب الأداة button وتوقيعها على نموذج البرنامج فيتم التعريف تلقائيا ولذلك سمي برمجة مرئية حيث يمكنك تكوين شكل البرنامج بشكل مرئي وليس عليك كتابة الأوامر لإنشاء قائمة اختيارات menu بل تعدها بشكل مرئي (visual) ، وقامت على هذه الفكرة معظم اللغات الحديثة مثل اللغات التالية:

Delphi, Borland C, Oracle, Visual Basic, Visual Basic.Net , Visual C++,Visual C#,

ويعتبر هذا هو المفهوم السائد حاليا ومعظم اللغات تطبقه حيث تتميز اللغات الحالية بأنها :

1ـ برمجة مرئية Visual Programming

وذلك يسهل عملية إعداد البرامج ويزيد من سرعة عملية التطوير والإعداد فليس عليك كتابة أوامر تعريف الفصائل التي تستعملها فقط تستعملها بشكل مرئي ، حيث تقوم بسحب الادكة من شريط الأدوات Tool Box وتوقعه على نموزج خلفية التطبيق Form والشكل .. يعرض نموزج خلفية التطبيق Form موقع بها أدوات Tools وذلك في لغة Visual Basic

والشكل ... يعرض مربع الأدوات ToolBox المستخدم في اعداد التطبيق

2. تبنى على مفهوم OOP

معظم اللغات الحديثة تعمل بمفهوم البرمجة بالأهداف (OOP) مثل:

لغة C++ VISUAL ولغةVB.NET ولغة C# ولغة JAVA حيث بنى التطبيق على فصائل

Classes وليس دوال والشكل .. يعرض نص به فصائل classes من لغة Java

```
1  package mypackage();
2  public class Ex1
3  {
4    public Ex1()
5    {
6    }
7  public static void main(String[] args)
8    {
9    new Ex1();
10     System.out.println("Welcome With First Java Program");
11   }
12 }
13
```

3. توفر معظم أغراض البرمجة

توفر اللغات الحديثة التعامل مع معظم أغراض التطوير ، فمعظمها يوفر التعامل مع قواعد البيانات Database وهي من أشهر مجالات استعمال لغات البرمجة والتطوير ، والتعامل مع الوسائط المتعددة (multimedia) ، وتطوير تطبيقات الإنترنت Web programming وهي من الاتجاهات الحديثة للبرمجة والتطوير والتي تتجه اليها التطبيقات بشكل كبير حيث يتم التخطيط لتحويل جميع التطبيقات الى تطبيقات انترنت

لغات تطوير تطبيقات الإنترنت : Web Developing

مع ظهور شبكة الإنترنت وانتشارها وتزايد الإقبال على استعمالها ، ظهرت لغات إعداد تطبيقات تعمل على شبكة الإنترنت والتي يطلق عليها تطبيقات الإنترنت (web application) ولم يعد التطوير قاصر على كتابة البرامج التقليدية ، بل ظهر مفهوم مهم وهو تطبيقات الإنترنت مثل إعداد تطبيق التوظيف الذي نستطيع استخدامه من خلال شبكة الإنترنت للبحث عن وظيفة و العكس للبحث عن موظف من خلال قاعدة بيانات يتم التسجيل فيها والتعامل منها ، وكذلك إعداد تطبيقات التجارة الإليكترونية (E-Commerce,E-Business) التي تمكن الزائر من الدخول على موقع شركة أو مكتبة

والإطلاع على المنتجات أو الكتب ، وعقد صفقة أو شراء مجموعة من الكتب من خلال شبكة الإنترنت وأصبح اتجاه الإنترنت وإعداد مواقع وتطبيقات يطلق عليها مواقع تجارة الكترونية E-Commerce أو مواقع تجارة أعمال E-Business مهم جداً.

وعلى هذا الأساس ظهر الكثير من الأدوات و اللغات التى تخدم ذلك.

الأدوات المشهورة لتصميم مواقع الإنترنت البرامج التالية :

البرنامج	استخدامه
Front page	تصميم المواقع
Dream waver	تصميم المواقع
Flash	تصميم المواقع وإعداد الأشكال المتحركة
Swish	إضافة حركة للنصوص لعرضها على صفحة الإنترنت
Ra3d	إعداد أزرار أوامر وكذلك التعامل مع النصوص المجسمة
Photo shop	إعداد الصور والصور المتداخلة
Anfy	التعامل مع الصور و النصوص وإضافة تأثيرات جيلة

كما ظهرت اللغات المشهورة والسكربتات Scripts للتعامل مع مواقع الإنترنت :

اللغة	استخدامها
Java	إعداد تطبيقات الإنترنت وإعداد تطبيقات من نوع Applet
Java Script	كتابة أوامر تنفذ عند مستخدم الإنترنت
Asp	كتابة أوامر تنفذ على أى جهاز خادم يستضيف الموقع
Php	كتابة أوامر تنفذ على أى جهاز خادم يستضيف الموقع
Perl	كتابة أوامر تنفذ على أى جهاز خادم يستضيف الموقع
Jsp	كتابة أوامر تنفذ على أى جهاز خادم يستضيف الموقع
VB.NET	تستعمل لإعداد تطبيقات إنترنت كاملة

قواعد البيانات database

في الفترات القديمة والأولى لتطوير الكمبيوتر لم يكن لقواعد البيانات مكان بل كانت البيانات تحفظ في ملفات تسمى ملفات مستوية Flat Files ، ولكن مع التطور اصبح مفهوم قواعد البيانات ، من أهم المفاهيم ، وهو تسجيل البيانات في شكل منظم عبارة عن حقول وسجلات في جداول ، ومجموعة الجداول تشكل قاعدة بيانات وتوفر عمليات متعددة مثل البحث ، والترتيب ، والعمليات الحسابية على الحقول وغيرها .

وبدأت قواعد البيانات قديما ببرامج بسيطة مثل dbase III+ ، وطورت بعد ذلك لتشتمل الآن على الكثير من البرامج منها :

1- برنامج MS Access الذي يأتي ضمن مجموعة MS Office

2- قاعدة بيانات Oracle والتي تمثل من أقوى قواعد البيانات حتى الآن

3- MS SQL server والذي أصدرته مايكروسوفت ليضاهي قاعدة بيانات Oracle

4- قاعدة بيانات DB2 من انتاج شركة IBM ، والكثير من لغات قواعد البيانات

أدوات التطوير حاليا

في الوقت الحاضر استبدل لفظ البرمجة (Programming) بلفظ جديد أطلق عليه التطوير (developing) لأنه لم يعد قاصر على كتابة الأوامر وإعداد التطبيقات التقليدية لذلك ينقسم التطوير developing حاليا إلى :

إعداد تطبيقات لكمبيوتر واحد :Desktop Application

ويطلق على هذا النوع من التطبيقات تطبيق مكتبي desktop app ، وهناك الكثير من اللغات لإعداد هذا التطبيق مثل لغة visual basic أو لغة Visual C++ , أو لغة c++ أو النسخة الشخصية من Oracle (Oracle personal edition) والكثير من اللغات الأخرى.

إعداد تطبيقات لشبكة Distributed Application

ويطلق على هذا النوع من التطبيقات تطبيقات الشبكة client/server app وفيها يتم

وضع قاعدة البيانات على الجهاز الخادم (server) وإدخال البيانات من اكثر من جهاز مثل تطبيقات البنوك والعقارات وغيرها وأشهر اللغات لهذه التطبيقات هى لغة Oracle وكذلك اللغات الأخرى مثل VB.,VC, JAVA .

تطبيقات الإنترنت web Applications

هذا هو الشكل الحديث من التطبيقات والذى يعمل من خلال شبكة الإنترنت مثل تطبيقات بيع الكتب وغيرها

وظهر لذلك الكثير من اللغات وسكريبتات الانترنت مثل Java , java script, asp

وكذلك توفر لغة Oracle إعداد تطبيقات قواعد بيانات خاصة بشبكة الإنترنت.

و السؤال :

ماذا افعل لكي أكون مطور developer؟

الإجابة: حدد أحد اختيارين

تطبيقات كمبيوتر وشبكة Desktop &networe app تطبيقات إنترنتweb Application

إذا كانت الإجابة هى الاختيار الأول (تطبيقات كمبيوتر أو شبكة) فعليك بتعلم أحد اللغات التى توفر ذلك وليكن لغة visual basic

- وإذا كانت التطبيقات كبيرة تحتاج قاعدة بيانات قوية تعلم oracle
- إذا كانت الإجابة هى الاختيار الثانى (تطبيقات الإنترنت)

فعليك إن تسير فى المسار التالى :

- أحد البرامج الجاهزة وليكن Front Page – قواعد لغة HTML
- أحد لغات الأوامر مثل Java Script – أحد لغات الخادم مثل asp أو jsp
- لغة قويه مثل Java أحد قواعد البيانات (Access, Oracle, Sql, Server)

نبذة عن لغات البرمجة الحديثة

لغة البرمجة C/C++

بدأت لغة C عام 1960 بلغة سميت CPL طورت عام 1967 وسميت BCPL على يد مارتن وريتشارد ثم طورت وسميت لغة B اختصارا ،وفى عام 1972 طورت فى معامل AT&T لتصبح لغة C ، وأصبحت لغة قياسية عام 1973 حيث انتجت ANSI C وتعتبر من اللغات الام فى لغات البرمجة ،

تركيب برنامج C

يمكن تنفيذ برنامج لغة C فى بيئة نظام التشغيل DOS ومع مترجمات لغة C مثل Borland C, Turbo C وذلك بإعداد TURBO C/C++ ثم تشغيله ثم كتابة البرنامج كما فى الشكل التالى

```
1:#include <stdio.h>
2: main()
3: {
4: printf("\n Welcome With First Program in C Language\n");
5:        }
```

فى الشكل يظهر الهيكل الاساسى لبرنامج لغة C ويتكون مما يلى

السطر #include "stdio.h" ويقوم بفتح (الاشتمال على) الملف stdio.h الذى يحتوى على تعريفات الدوال المهمة والمستعملة خلال البرنامج.

السطر main() فيه الدالة الرئيسية للبرنامج main() وبين أقواسها .

السطر { :هو قوس بداية البرنامج الذى يكتب بعد تعريفات المتغيرات وسطور البرنامج والقوس } لاغلاق الدالة الرئيسية وكثيرا من اللغات قامت بتقليد قواعدها وطرق كتابة أوامرها مثل لغة Java,C#,PHP,... لذلك تسمى هذه اللغات C-Like لالتزامها بالقواعد العامة للغة C.

من أوائل اللغات التي طورت لغة C بعد ذلك وإضافة قواعد البرمجة بأسلوب OOP لتخرج باسم C++ واعتمدتها شركة ميكروسوفت وقامت باعداد منتجها المعروف Visual C++

لغة البرمجة Java

هى لغة من اللغات الحديثة أنتجتها شركة SUN Microsystems لتناسب التطبيقات الحديثة عام 1995.

وهى تناسب تطبيقات الإنترنت حيث أصبحت هى قلب برمجة الإنترنت بما توفره من إمكانيات وكذلك الأنواع المختلفة من التطبيقات مثل تطبيقات تعمل على جهاز Desktop Application أو تطبيقات تعمل على شبكة Client/Server Application وتتصف لغة Java بالصفات التالية:-

1- لغة تلتزم بالبرمجة بواسطة الأهداف OOP
2- لها بيئة تشغيل خاصة بها JVM
3- لها مكتبة فصائل Class Libraries
4- تقوم على لغة C/C++
5- تعمل على معظم نظم التشغيل PLATFORM INDEPENDENT

وفيما يلى شرح هذه النقاط

1- لغة تلتزم بقواعد البرمجة بواسطة الأهداف OOP:

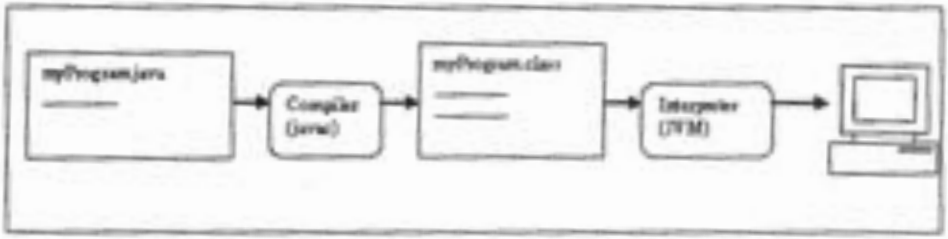
البرمجة بواسطة الأهداف OOP (object oriented programming) :

من الطرق الحديثة للبرمجة والتى وفرت كثير فى الجهد الذى كان يبذل باستخدام البرمجة التقليدية، حيث كانت البرمجة التقليدية توفر للمبرمج مكتبة من الدوال وتركيب تقليدى للبرنامج وعليه هو أن يستعمل الدوال مع تركيب البرنامج لإنشاء التطبيقات مما يضطره لكتابة السطور الكثيرة وكذلك يضطره لإعادة كتابة السطور أكثر من مرة، فكانت وحدة بناء البرنامج هى الدالة. فى حين أتت البرمجة بواسطة الأهداف OOP بفكرة جديدة هى إنشاء عناصر متكاملة تحتوى على بيانات ودوال وهى أساس إنشاء البرنامج وبالتالى

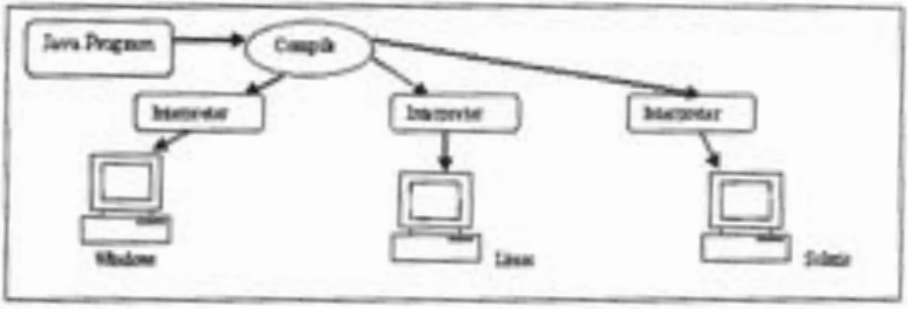
أصبحت وحدة بناء البرنامج وحدة كبيرة هي الفصيلة أو العنصر Object مما سهل واختصر الكثير وسوف نلخص هذه القواعد فيما بعد

2. لغة لها بيئة تنفيذ خاصة JVM :

قامت لغة Java بإعداد بيئة تشغيل للبرنامج هي JVM التي تقوم بترجمة البرنامج للغة الآلة وبالتالي فأن لغة Java غير مرتبطة بنظام تشغيل معين فقط يتم ترجمة نفس البرنامج على نظام التشغيل ووجود بيئة التنفيذ JVM يؤدى إلى تنفيذ البرنامج وهذه أحد أهم مزاياها.



3. لا تعتمد على نظام التشغيل Platform Independent



3. لها مكتبة فصائل قوية Classes Lib:

نظر لأن لغة Java لغة تعتمد على OOP فهي تحتوى على مكتبة فصائل قوية توفر مكتبة فصائل معظم أو كل الفصائل المطلوبة لإعمال مثل التعامل مع الملفات والتعامل مع قواعد البيانات Data Base والتعامل مع الشبكات Networks وكذلك التعامل مع الرسومات المجسمة 3D Graphics والحركة Animation وكذلك التعامل مع الإنترنت. Web Developing

لغة مبنية على لغة C/C++ :

عندما تم إنشاء لغة Java كان أساس بنائها لغة من أشهر وأقوى اللغات وهى C/C++ وبالتالى فهى لم تبدأ من حيث بدأ الآخرون بل من حيث انتهى الآخرون وهو لغة C/C++ ثم إضافة لها الجديد فى اللغة Java.

تعريفات هامة

لغة البرمجة Visual Basic.Net

من أكثر اللغات شهرة خلال الفترة السابقة والحالية ،فقد جزبت الكثير والكثير من محبى البرمجة ... وذلك لما تتمتع من سهولة الاستخدام وإمكانية الانتاج بسهولة ، وقد تدرجت فى الاصدارات حتى الإصدار 6.0 وكان من أشهر الاصدرات ثم تطورت كثير مع مجموعة التطوير Visual Studio.Net وأصبحت Visual Basic.Net

من أكثر اللغات استعمالا وحتى كتابة هذا الكتاب كان أخر الاصدرات هو Visual Studio.Net 2005 والاصدار 2008 ما زال نسخة تجريبية Beta

ما هى .NET.

تطلق مايكروسوفت على الإصدار السابع من مجموعة أدوات البرمجة الخاصة بها والتى تأخذ الاسم Visual Studio تطلق عليها الاسم Visual Studio. NET وكانت تشتمل فى الإصدار السادس (Visual Studio 6) على مجموعة اللغات والأدوات التالية:

1- Visual C++
2- Visual Basic
3- Visual Fox pro
4- Visual J++
5- Visual Inter Dev

فقامت شركة مايكروسوفت بتطويرها وأخرجت الإصدار السابع المنتج النهائى فى الأسواق والذى أطلقت عليه الاسم Visual studio.NET فى نهايات شهر فبراير سنة 2002

(ثم قامت بتطوير طفيف وعلاج بعض الأخطاء فى هذا الإصدار وأصدرت Visual studio. NET وأطلقت عليه 2003) ونشرت عنه الكتب وقالت أن الفرق بين الإصدار 6 (السابق) وهذا الإصدار (.NET) كالفرق بين الإصدار الأول والإصدار 6 وأن فيه مزايا تعتبر تكنولوجيا جديدة فى عالم البرمجة وتطوير التطبيقات وحذفت من الإصدار السابق بعض الأدوات وأضافت بعض الأدوات الجديدة وأصبح الإصدار السابع والذي يسمى Visual studio.net يشتمل على اللغات والأدوات التالية:

1- Visual C++ .NET
2- Visual Basic .NET
3- Visual C# (C Sharp)
4- ASP.NET
5- JScript. NET

هذه هى أدوات Visual Studio.NET والتى سوف نتحدث عنها بعض الشيء فيما بعد ، تركز شركة ميكروسوفت فى المنتج الجديد .NET على توفير متطلبات تطوير برامج وتطبيقات الإنترنت مثل إعداد التطبيقات التى تعمل من خلال الإنترنت، التحدث من خلال الإنترنت، التعامل مع البريد الإلكترونى Email وغيره من متطلبات التعامل مع الإنترنت حيث ترى شركة مايكروسوفت أن هذا هو الاتجاه المقبل وهو اتجاه استعمال شبكة الإنترنت فى كل شئ.

سؤال: لماذا تستعمل vb.net ولا تستعمل اللغة الجديدة C# ؟

الإجابة:

- اذا كنت من مبرمجى vb فأنت تعلم أساسيات اللغة وبالتالى فأسهل لك استعمال vb.net حيث أنها امتداد للغة بدلا من تعلم أساسيات لغة جديدة.
- لغة C# تبنى على لغة C/C++ وبالتالى لتتعلم لغة C# عليك تعلم أساسيات C/C++ التى تحتوى على قواعدها الخاصة مثل المؤثرات المختلفة مثل المؤثر (==) بدلا من المؤثر = و الكثير من النقاط الأخرى.

تتميز لغة VB.Net بالسهولة فى القواعد وعدم التقيد بقواعد شديدة فهى لا تتقيد مثلا بطبيعة الحروف المكتوبة كبيرة Capital أو صغيرة Small و لا تفرض إنهاء السطور بالفاصلة المنقوطة ; كما تفرض لغة C ولغة البرمجة Visual C#.Net

المواصفات العامة للغة C#

لغة C# من أحدث اللغات وجائت مع مجموعة التطوير Visual Studio.Net من إنتاج شركة ميكروسوفت وهى لغة تجمع بين سهولة لغة Visual Basic وقوة Visual C++ ،كما وفرت فيها ميكروسوفت بعض مزايا لغة Java ، وتعتمد عليها شركة ميكروسوفت حاليا كثيرا ..وتعتبرها لغة العصر وتوفر المزايا التالية :

- مفاهيم البرمجة بالأهداف object Oriented programming التى تعتبر أساس أدوات التطوير والبرمجة الحديثة.
- مكتبة الفصائل Classes التى توفرها اللغة والتى توفر الكثير على المطور.
- استخدام طرق حديثة لمعالجة أخطاء البرنامج Try Catch.
- سهولة إعداد تطبيق إنترنت Web Application الجديد فى vb.net الذى أضافته شركة ميكروسوفت إلى vb.net وترى أنه تكنولوجيا جديدة يجب الانتقال إليها وأنها أحدثت ثورة و هى:

1. الجديد فى أساسيات .NET framework

قامت ميكروسوفت بإجراء تغيير كبير فى أساسيات vb والذى تطلق عليه vb.net framework، فقد غيرت ما يلى:

- أضافت طريقة جديدة لتجميع الفصائل (مكتبة فصائل) وأطلق على هذة الطريقة الكلمة namespaces.
- تطبيق قواعد البرمجة بواسطة الأهداف oop وأهمها الخاصية التوريث inheritance وباقى الخصائص الهامة.
- تطبيق إمكانية تنفيذ أكثر من عملية فى نفس الوقت والذى يسمى threading.

- سهولة التعامل مع لغة الترميز الجديدة XML والتى توصف بأنها لغة المستقبل فى وصف البيانات.
- إضافة لغة وسيطة لجميع لغات vs. net تسمى IL(intermediate language) وهى لغة تحول إليها برامج جميع برامج اللغات مما يوفر التعاون بين اللغات.
- إضافة بيئة تنفيذ مشتركة بين لغات vs. Net ويسمى CLR(common language runtime) والتى تتولى ترجمة نصوص البرامج المحولة إلى IL مما يوفر إمكانية إعداد برامج تعمل على أكثر من نظام تشغيل (platform independent).
- توفير بيئة تصحيح أخطاء واحدة (debugger) لجميع لغات vs.net حتى لو كان التطبيق يحتوى على أكثر من لغة.
- إضافة تحسينات فى إمكانيات إعداد تطبيق أنترنت web application وذلك بالتعاون مع asp. Net.
- توحيد أنواع البيانات (data types) و إحجامها بين اللغات مما تسهل التعاون بين اللغات.
- بعد عرض بعض النقاط الجديدة فى vb.net بشكل مختصر نبدأ فى تناول هذة النقاط بالتوضيح:
- بوجود بيئة تنفيذ (CLR) لتنظيم التشغيل المختلفة مثل windows أو unix وغيره يصبح البرنامج الذى تكتبه غير مرتبط بنظام التشغيل وهذا ما يسمى plateform independent وهذا مايسعى اليه شركة ميكروسوفت وإن كانت حتى الآن لم تعد بيئة تنفيذ (CLR) لغير نظام التشغيل windows.

برمجة الانترنت ASP.NET

هى من الطرق الحديثة لاعداد تطبيقات انترنت Web Application من انتاج ميكروسوفت وتوفر معظم عناصر ومهارات برمجة مواقع الانترنت ، ويتم استعمالها مع لغة VB.NET أو مع C#

لغة Oracle PL/SQL

تحتوى Oracle على جميع أدوات إعداد تطبيق متكامل فهى تحتوى على محرك قاعدة البيانات القوى DB Engine و الذى نتعامل معه باستعمال جمل SQL ، وتحتوى على مصمم النماذج Form builder وتحتوى على مصمم التقارير Report builder ، وكثير من الأدوات الأخرى.

وتحتوى أيضاً على لغة برمجة خاصة بها تسمى PL/SQL وفيها الحرفين PL اختصاراً للعبارة Procedure language والحروف SQL كما سبق ، وتم تسميتها PL/SQL لأنها عبارة عن مجموعة أوامر اللغة التى توفر العمليات التقليدية مثل أى لغة برمجة بالاضافة لسهولة دمج أوامر SQL إليها .

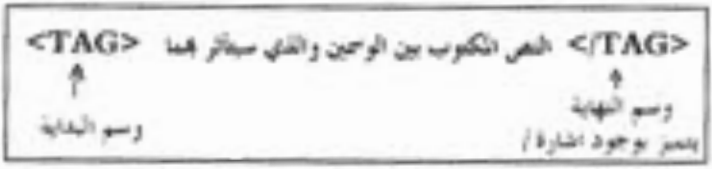
سكربتات الانترنت Web Scripts

لغة الترميز HTML

لغة HTML هى اللغة المستخدمة لإنشاء صفحات الإنترنت والكلمة اختصار لمجموعة كلمات إنجليزية هى (Hyper Text Markup Language) وهى ليست لغة برمجة بالمعنى والشكل المتعارف عليه للغات البرمجة الأخرى كلغة C فهى مثلاً لا تحتوى على جمل التحكم والدوران ، وعند الحاجة لاستخدام هذه الجمل يجب تضمين لغات أخرى مثل (Java, JavaScript, CGI) كذلك فهى لا تحتاج إلى Compiler (مترجم خاص) به وهى غير مرتبطة بنظام تشغيل معين، لأنه يتم تفسيرها وتنفيذ تعليماتها مباشرة من قبل متصفح الإنترنت وبغض النظر عن النظام المستخدم .

لذلك فهى لغة بسيطة جداً ، وسهلة الفهم والتعلم ولا تحتاج لمعرفة مسبقة بلغات البرمجة المستخدمة ، بل ربما تحتاج قليل من التفكير المنطقى وترتيب الأفكار.

تتكون مفردات لغة Html من شفرات تسمى TAGS أى الرسوم وهى تستخدم بشكل أزواج وتكتب بالصيغة التالية (من اليسار إلى اليمين)



فعلى سبيل المثال الرسم <B> يستخدم لكتابة الكلمات بخط أسود عريض Bold وذلك بالشكل التالي <B> Text </B> وهناك بعض الرسوم الخاصة التي تستخدم بصورة مفردة مثل رسم نهاية السطر
 أو قد تستخدم بكلتا الحالتين مثل رسم الفقرة <P>.

سكربتات JavaScript

هي عبارة عن أوامر برمجة تكتب بصيغة الـ Java داخل ملفات الـ html لتحقيق وظائف معينة مثل التأكد من صحة بيانات نموزج تسجيل على صفحات الانترنت وتكتب داخل صفحات الانترنت HTML كما في السطور التالية

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>

</body>
</html>
```

سكربتات VB Script

هي أوامر سكربتات من نوع Client Script الى التي تنفذ على جانب جهاز المستخدم وليس جهاز Server لتحقق وظائف مثل وظائف الـ JavaScript ،وتأخذ شكل أوامر لغة الـ Visual Basic

سكربتات ASP

هي أوامر سكربتات تكتب لتنفذ على جانب الـ Server لتحقيق وظائف مثل التعامل مع قواعد البيانات وتحقيق عمليات الاضافة والحذف والتعديل على جانب الـ Server وتأخذ شكل أوامر لغة الـ Visual Basic

سكربتات ASP.NET

هى التطوير الخاص بـ ASP حيث تم تطويره ليعمل ضمن مجموعة أدوات ولغات Visual Studio.Net مما سهل إعداد تطبيقات الإنترنت.

سكربتات PHP

هى أحد السكربتات التى تكتب لتنفذ على جانب الـ Server لتحقيق وظائف مثل التعامل مع قواعد البيانات وتحقيق عمليات الإضافة والحذف والتعديل على جانب الـ Server ولكن تتميز بانها

1- مجانية لانها أحد الأدوات التى تخضع لاتفاقية الـ Open Source

2- يمكنك الاضطلاع على الـ Source Code

3- بسيطة ولاتحتاج الى امكنيات كثيرة للتطوير

4- تبنى على لغات الـ C فى شكل الاوامر

قواعد البيانات Database

بالرغم من اننا نتكلم عن البرمجة Programming الا انه لم يعد يصلح الحديث عن البرمجة بدون الحديث عن قواعد البيانات Database ،حيث ان معظم تطبيقات اليوم تحتاج التعامل مع قواعد البيانات بشكل أو أخر

لذلك تعالى معنا نتناول فكرة عن قواعد البيانات وكذلك أدواتها الحالية

ما المقصود بقواعد البيانات WHAT IS DATABASE

بصرف النظر عن استعمال الكمبيوتر لو نظرت ورقيا تجد أن أى مؤسسة تحتفظ بالبيانات فى شكل قاعدة بيانات فمثلاً.

بيانات العاملين

تجد إدارة شئون العاملين تحتفظ بملف لكل موظف يحتوى هذا الملف على بياناته الشخصية ومؤهلاته والشهادات الحاصل عليها حتى الآن. وتواريخ الترقى وكذلك العلاوات والجزاءات والدورات الحاصل عليها وهذه تسمى قاعدة بيانات فى أبسط صورة.

صديقى

إذا كنت على دراية بأساسيات قواعد البيانات واستخدام قواعد البيانات MS ACCESS أو SQL SERVER أو ORACLE (كما هو متوقع) يمكنك الانتقال إلى الفقرة المناسبة لمستواك حتى لا تشعر بالملل حيث إننا فى هذه الفقرة نراجع أساسيات فى قواعد البيانات يجب الاتفاق عليها.

بيانات العملاء

فى إدارة المبيعات تلاحظ الإدارة تحتفظ بيانات العملاء الشخصية وحركات البيع وحجم المبيعات والمواقف المالية وغيرها ، وهذه تسمى قاعدة بيانات. وهناك الكثير من الأمثلة الأخرى ، ولذلك تلاحظ أنه لا غنى عن وجود قواعد بيانات ولو ورقية ، ولكن السؤال لماذا تحول قاعدة البيانات الورقية إلى كمبيوتر؟

الإجابة:

- صعوبة البحث فى قاعدة البيانات الورقية والوقت المستغرق فى حين يتوفر ذلك فى قاعدة البيانات الكمبيوتر بسهولة.
- صعوبة الحصول على معلومات تحليلية من البيانات فى حين يتوفر ذلك فى قاعدة بيانات الكمبيوتر.
- صعوبة التأكد من دقة المعلومات لاحتمال الخطأ فى حين توفر الدقة فى معلومات قاعدة بيانات الكمبيوتر وكثير من المزايا التى سوف نشرحها.

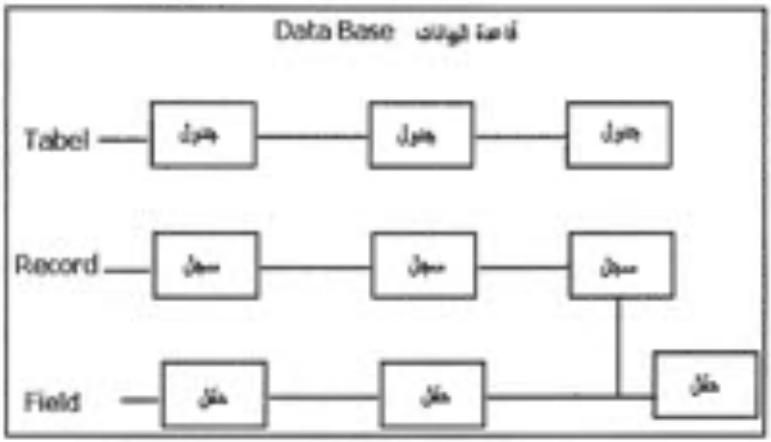
مزايا استعمال الكمبيوتر فى إنشاء قاعدة بيانات

- الاحتفاظ بكم هائل من البيانات بعيداً عن الأوراق وسهولة عمل نسخ كثيرة من هذه البيانات.
- سهولة ربط البيانات ببعضها مثل بيانات العاملين وبيانات أصناف المخازن.
- سرعة البحث الهائلة عن البيانات التى لا تقارن مع قاعدة البيانات الورقية.
- إمكانية استخلاص البيانات (المعلومات) التحليلية من كثير من البيانات مثل بيانات موظفين على درجة معينة أو موظفين حاصلين على دورة معينه ..إلخ.

- دقة المعلومات المستخلصة حيث يخرج الكمبيوتر ما تم تسجيله.
- إمكانية إجراء العمليات المختلفة على جدول أو أكثر من البيانات فى وقت قصير مثل زيادة أجر الموظفين بنسبة معينه أو صرف مكافئة لهم.
- إمكانية استخراج التقارير بأشكال مختلفة مثل الرسوم البيانية وغيرها.

1ـ تركيب قاعدة البيانات

تأخذ قاعدة البيانات التركيب التالى (1):



شكل(1-1)

من هذا الشكل نلاحظ أن:

قاعدة البيانات مثلا العاملين تتكون من جداول مثل جدول البيانات الأساسية – جدول الإجازات – جدول الدورات. [TABLES]

والجدول (TABEL) يتكون من سجلات (RECORDS) حيث يمثل كل موظف داخل الجدول بسجل ويسمى أحيانا ROW أى صف.

والسجل (RECORD) يتكون من حقول (FIELDS) حيث يتكون سجل الموظف من حقول مثل كود الموظف، اسم الموظف العنوان، التلفون، تاريخ الميلاد ..إلخ ، أى تأخذ قاعدة البيانات التركيب التالى :

DATA BASE –TABLES – RECORDS- FIELDS

عناصر قاعدة البيانات Relational Data Base components

هناك مجموعة من العناصر التي تتكون منها قاعدة البيانات والتي سوف تقوم بإنشائها هي:

1. الجداول tables

وهي أول وأهم العناصر حيث يتم انشائها قبل جميع العناصر لأنها المسؤلة عن حفظ البيانات وعليها تبنى باقى العناصر مثل الاستعلام والتقارير وغيره ،وتتكون الجداول من سجلات (RECORDS) تسمى أحيانا صفوف Rows والسجلات تتكون من حقول (FIELDS) تسمى أحيانا أعمدة Columns وللجداول أهمية خاصة لأنها أساس بناء قاعدة البيانات وبالتالى أساس بناء التطبيق.

2. العنصر VIEW

وهو عنصر يعرض جزء من جدول أو اكثر من جدول باستعمال جملة SELECT ويستعمل لأكثر من غرض مثل تحديد صلاحيات المستخدم للتعامل مع سجلات محددة أو التعامل مع اكثر من جدول.

3. العنصر SYNONYM

وهو اسم بديل للجدول و يسهل التعامل مع جدول TABLE فى قاعدة بيانات وذلك بوضع اسم مختصر للجدول بدلاً من اسمه الذى ربما يكون طويل وغير مناسب للتعامل به.

4. العنصر INDEX

وهو عنصر يرتب جدول البيانات حسب حقل أو اكثر وذلك لزيادة سرعة البحث فى هذا الجدول عن طريق هذا الحقل.

5. العنصر TRIGGERS

هو كود (أوامر) تكتب وتلصق بقاعدة البيانات تسمى database triggers ويتم تحديد حالة وقت حدوثها مثل الإضافة والحذف والتعديل وتستخدم لمراجعة صحة البيانات قبل تنفيذ العملية فمثلا عن طريقها تستطيع تنفيذ مجموعة من الاوامر قبل أو بعد الإضافة مثلا.

6. العنصر CONSTRAINTS

ومعناه قيود (شروط) وهي قيود توضع على الحقول للتحكم في البيانات المدخلة مثل الشرط primary key الذي لا يسمح بتكرار قيمة هذا الحقل أو الشرط not null الذي لا يسمح بترك حقل معين بدون قيمة.

7. الصلاحية PRIVILEGES

وهي الصلاحية التي تمنح للمستخدم عند التعامل مع قاعدة البيانات هل يسمح له بالإضافة فقط أم بالإضافة و الحذف أو أي صلاحية أخرى وبالطبع توجد بعض الصلاحيات المعرفة باوراكل.

8. العنصر ROLES

وتعني قاعدة وهو تجميع لمجموعة صلاحيات Privileges معاً لتسهيل منحها لمستخدم وبالطبع توجد بعض القواعد Roles المعرفة باوراكل.

9. العنصر procedures and functions

هو تجميع أوامر البرنامج في برامج فرعية تسمي procedure أو دوال function حسب الطلب ثم استعمالها وقت الحاجة وهي أحد طرق توفير إعادة كتابة الأوامر أكثر من مرة Code Reuse.

10. العنصر package

وهو تجميع مجموعة من الدوال و البرامج الفرعية (procedures & function).

تحت اسم معين يسمى package (حزم) وبعد ذلك تنادى الحزم package ومنها الدالة او البرنامج الفرعي المطلوب استعماله وهي طريقة أكثر تطور من طريقة إعداد دوال وبرامج فرعية منفصلة فبهذه الطريقة تستطيع إعداد حزمة package تحتوى على برامج فرعية ودوال لغرض معين تتفق فيه الدوال.

لغة قواعد البيانات SQL

لغة SQL عبارة عن مجموعة من الأوامر الخاصة بقواعد البيانات وهى اختصار للعبارة Structure Query Language وهى تحقق جميع عمليات قواعد البيانات مثل إنشاء الجداول والعناصر الأخرى وتسجيل وحذف وتعديل البيانات وكذلك جميع الاستعلامات بأنواعها .

وتتكون لغة SQL من الإجزاء التالية:

1. قسم التعريفات DDL

والحروف DDL اختصار للعبارة Data Definition Language

أي لغة التعريفات وهي مجموعة من الأوامر المسؤلة عن تعريف (إنشاء)عناصر قاعدة البيانات مثل الجداول ومن أشهر هذه الأوامر create.

2. قسم التعامل مع البيانات DML

والحروف DML اختصار للعبارة Data Manipulation Language أي لغة التعامل مع البيانات حيث توفر مجموعة الأوامر المسؤلة عن إضافة وحذف وتعديل البيانات و أشهر هذه الأوامر Insert, delete وعرض البيانات و الاستعلامات لجميع أشكالها ومن أشهر أوامر SELECT وهو أشهر هذه أوامر SQL .

3. قسم الاستعلامات DCL

والحروف DCL اختصار للعبارة Data Control Language أي لغة التحكم في عرض البيانات حيث توفر الأوامر المسؤلة عن الصلاحيات والتحكم فى امكانيات المستخدم من منحه صلاحيات ومنع صلاحيات منه.

أدوات (برامج) قواعد البيانات DBMS

برنامج قواعد البيانات MS Access

وهو أحد البرامج الموجودة فى حزمة البرامج (MICROSOFT OFFICE) والتى أنتجتها شركة مايكروسوفت ولا يخلو جهاز كمبيوتر الآن من هذه المجموعة ، ووظيفة إنشاء

قواعد البيانات والتعامل معها ويعتبر برنامج MS ACCESS من أسهل الطرق الحالية في التعامل مع قواعد البيانات.

برنامج قواعد البيانات MS SQL Server

هو أحد برامج قواعد البيانات الحالية والحديثة من انتاج شركة MicroSoft وبالطبع يوفر عناصر قوة أكثر بكثير من البرنامج MS Access من حيث السعة التخزينيه للبيانات وسرعة التعامل مع البيانات والبحث فيها.

وكذلك الإمكانيات المختلفة مثل الإجراءات المخزنة Stored Procedures وكذلك توفيره إمكانية إنشاء قاعدة بيانات مركزية لتعمل بنظام Client /Server Database Application.

برنامج قواعد البيانات Oracle

يمتاز نظام قواعد البيانات أوراكل بالأداء العالي بسبب التزامه بالمقاييس والكثير من الأسباب منها ما يلي:

- أن تكون إمكانية التخزين كبيرة جداً.
- إمكانية التعامل مع عدد كبير من المستخدمين بشكل متزامن ومتوازي.
- نظام آمن من الممكن عدم توقف التعامل معه على مدار اليوم وقدرته على إصلاح الأعطال حتى الفيزيائية منها.
- حماية البيانات من العبث بها أو الحصول عليها من غير المستخدمين المحددين.
- الأداء العالي إذ أن زيادة البيانات والمستخدمين لا تؤثر على سرعة وعمل أوراكل.
- المواصفات القياسية الخاصة بأوراكل مواصفات عالمية.
- إمكانية العمل على اكثر من نظام تشغيل.
- إمكانية الاتصال القوية عبر الشبكات.
- التعريب إذ يمكن التعامل معه باللغة العربية.

برنامج قواعد البيانات DB2

هو برنامج قوي لادارة قواعد البيانات من إنتاج شركة IBM ويوفر بدرجة عالية

التعامل مع قواعد البيانات الكبيرة وكذلك امكانية انشاء قاعدة بيانات مركزية لتعمل بنظام Client /Server Database Application.

برنامج قواعد البيانات MySQL

هى قاعدة بيانات جديدة خرجت إلى الواقع (السوق) بنظرية البرامج المجانية مفتوحة الكود التى يمكن للجميع من استعمالها وتطويرها (GPL) General Public License وتستخدم الآن من قبل كثير من كبار العملاء مثل Yahoo! Finance.

وهى سهلة الاستعمال

ولها كثير من المزايا منها :

1- قاعدة بيانات خفيفة وسريعة ومستقرة **fast and stable** بالمقارنة بجميع قواعد البيانات الموجودة بالسوق مثل Oracle, Microsoft SQL Server, DB2 فعند قيام أحد المؤسسات بمقارنة MySQL 4.0 بقاعدة بيانات Oracle 9i وجد أن MySQL 4.0 أسرع ويمكنك قراءة المقارنة فى هذا العنوان : www.mysql.com/information/benchmarks.html

2- تتميز قاعدة بيانات MySQL بأنها مجانية free مما يشجع كثيرا من المؤسسات التى لا ترغب فى الدخول فى قصة الشراء والبيع والتراخيص.

3- توفر معظم ما توفره قواعد البيانات من مزايا مثل أساسيات قواعد البيانات من انشاء العناصر والتعامل معها ، السماح بأدارة الحركة Transactions وكذلك التحكم فى اغلاق lock البيانات على مستوى الصف Row ، كما توفر الاستعلامات Queries وكذلك العلاقات باستعمال foreign keys كما توفر الخاصية المتقدمة وهى البحث فى النصوص full text searching وقد أضاف كذلك الاصدار 5.0 امكانية انشاء أجراءات مخزنة stored procedures

4- سهلة الاستعمال **Ease of Use**: فعلا عند استعمالنا لها سوف نلاحظ مدى سهولة اجراء عمليات قواعد البيانات باستخدام MySQL

وهذه هى المزايا كما عرضها الموقع الرسمى لها MySQL.com:

- توفر معظم القواعد الاساسية للغة SQL Support for most SQL 92 syntax.
- توفر امكانية التعامل مع الاستعلامات الفرعية Subqueries.
- توفر خاصية البحث فى النصوص Full text search.
- توفر إمكانية التعامل مع الجداول الكبيرة Support for huge tables and databases.
- توفر إمكانية عمل نسخ من قاعدة البيانات متزامنة Replication.
- التعامل مع نظم التشغيل Cross-platform support.
- وقد أضاف الإصدار 5.0 كما أشرنا إمكانية إنشاء Stored procedures.

أسئلة:

وقبل مغادرة هذا الفصل نود أن نعرض عليك بعض الأسئلة التى ربما ما تجول كثيرا بالخاطر.

- **ما هى أنواع SoftWare ؟**

ينقسم SoftWare إلى نوعين نظام و Application Software و System Software.

- **ما المقصود بـ System Software ؟**

هى نوعية من البرامج تصمم لتخدم الجهاز وتتعامل معه مثل نظم التشغيل ومن أمثلة ذلك نظام إدارة البيئة الرسومية الخاص بنظام تشغيل الحاسب Windows وعملية تصميم النظم تخضع لعمليات التجربة والخطا حتى الحصول على تصميم مستقر.

- **ما هو علم الذكاء الاصطناعى ؟**

هو العلم الذى يهدف لوضع نموذج حسابى أو الجورزم للتطبيقات التى تقوم بعمل محاكة للذكاء الإنسان مثل لعبة الشطرنج ومحاولة تحقيق حواس الإنسان مثل السمع والبصر والتفكير فى الإنسان .

- **ما هى الـ Appliaction Programming**

هى البرامج بصفة عامة غير برامج النظم ومن أمثلة ذلك برامج قواعد البيانات والألعاب وغيرها.

- **ما هى أنواع التطبيقات الشائعة فى الوطن العربى ؟**

من أشهر التطبيقات المشهورة فى دول الوطن العربى تطبيقات قواعد البيانات ثم البرامج التعليمية وقليل من برامج التحكم بالحاسب.

- **ما هى مواصفات المبرمج الذى يجيد تطوير انظمة قواعد البيانات ؟**

1- الإلمام بمراحل تطوير تطبيق.
2- الإلمام بالقواعد الاساسية لقواعد البيانات و القدرة على إنشاء قاعدة البيانات والتعامل معها.
3- القدرة على عمل واجهة جيدة للمستخدم باستخدام أحد لغات وأدوات التطوير الحديثة مثل VB.NET,C#,JAVA,Oracle.
4- الإلمام بالبرمجة باستعمال اسلوب OOP ليتسنى له تصميم نصوص البرنامج.
5- القدرة على تصميم أنظمة قواعد البيانات باسلوب Client /Server وهو الأسلوب الحديث التى توفره كل لغات البرمجة.
6- القدرة على الاستفادة من إمكانيات نظام التشغيل وبيئة العمل والتعامل مع الاجهزة الملحقة بالحاسب وغيرها من المهارات التى تميز بين المبرمجين.
7- القدرة على الإبداع وحل المسائل وذلك بتوافر مهارات البرمجة لديه.

- **ما هى الفترة الزمنية الكافية لتعلم البرمجة ؟**

إذا أمكن للشخص أن يتعلم لمدة 8 ساعات يوميا فإن ثلاثة شهور كافية للحصول على المعلومات الكافية ستة شهور لاتقان ما تعلمه وثلاثة شهور أخرى لعمل مشروع متكامل يكفى لأن يظهر المبرمج فيه ما تعلمه وسنة كاملة لاكتساب خبرة السوق والتعامل مع الأفراد من مبرمجين زملاء و عملاء ، أى أن عامين فترة جيدة للحصول على مبرمج محترف فى تطوير أنظمة قواعد البيانات.

- **ما هو المقصود بنظام تشغيل الحاسب Operating System؟**

هو البرنامج الاساسى لإدارة مكونات الحاسب ويتكون من

1- النواة والتى تكون مسئولة عن إدارة المكونات المادية للحاسب وتتكون هذه النواة من نظام لادارة العمليات ونظام لإدارة الذاكرة ونظام لإدراة الملفات.
2- واجهة النظام وقد تكون نصية أو رسومية.
3- إذا كانت الواجهة رسومية فانها تتكون من مكتبة جرافك للتعرف على كارت الشاشة والرسم عليه ونظام لإدارة البيئة الرسومية والتعامل معها.

- **ما المقصود ببرمجة الأحداث Event Driven!**

هو نظام البرمجة فى بيئة الوندوز حيث يمكن للمبرمج كتابة الأوامر المختلفة والتى تنفذ فى الأحداث المختلفة مثل الضغط على زر الماوس الأيسر أو الأيمن أو الضغط بالماوس على زر أمر button.

- **ما المقصود بتعدد المهام Multi Tasking!**

هى قابلية البرنامج لعمل أكثر من عملية فى وقت واحد وذلك بتقسيم زمن المعالج Processorعلى العمليات حيث أن سرعة الحاسب كبيرة.

- **مما تتكون لغة البرمجة ؟**

1- تحتوى على مترجم يقوم بتحويل البرنامج الذى يتم كتابته إلى لغة الآلة بعد التأكد من خلوه من الأخطاء.
2- تحتوى اللغات المتطورة على العديد من الأدوات لزيادة سرعة عمل التطبيقات وهى مفاهيم الـ Visual Programming التى تناولناها من قبل.
3- تحتوى على معالجات لسهولة اعداد الشكل الأولى لأنواع التطبيقات.

- **ماهى أجيال لغات البرمجة ؟**

1- لغة الآلة machine Language.
2- لغة التجميع Assembly Language.
3- لغات عالية المستوى hight Level.
4- لغات متخصصة فى قواعد البيانات Database Tools.

- **ما هى أنواع البرمجيات التى يتم تطويرها ؟**

1- أنظمة التشغيل Operating Systems.

2- برمجيات النظام System Applications.

3- لغات البرمجة Programming Languages.

4- أدوات ومكتبات للغات Libiries.

5- أنظمة قواعد البيانات Database Tools.

6- الألعاب Games.

7- البرامج التعليمية Education Applications.

8- برامج الاتصالات والشبكات Network Applications.

9- برامج معالجة الصوت.

10- برامج التحكم فى المكينات Control Applications.

11- البرامج الرياضية والهندسية.

12- البرامج المكتبية Office Applications.

13- برامج التصميم والجرافيك Graphics Prgrams.

الفصل الثانى

القواعد العامة للغات البرمجة
General Programming Concepts

فى هذا الفصل

Programming Concepts 2

نعرض مقدمة عن أساسيات لغات البرمجة والعناصر المشتركة بين اللغات مع عرض أمثلة بسيطة على ان يتم تفصيل ذلك فى الفصول التالية من خلال النقاط التالية:

- مقدمة
- البرنامج فى اللغات والمتغيرات والإدخال والإخراج
- جمل التحكم فى مسار البرنامج وجمل التكرار
- البرامج الفرعية Subprograms (Proc,Functions)
- التعامل مع المصفوفات فى لغات البرمجة
- مفاهيم البرمجة باستعمال اسلوب OOP
- خاصية التوريث فى مفاهيم البرمجة باستعمال اسلوب OOP
- التعامل مع أخطاء البرنامج Error Handling
- استعمال مفهوم الـ Visual Programming
- مقدمة سريعة عن أوامر قواعد البيانات SQL
- مفاهيم وطرق الاتصال والتعامل مع قواعد البيانات
- نظرية اعادة الاستخدام Reusability والمكتبات
- الرسم فى اللغات Graphics
- برمجة الشبكات Networking Programming
- مفهوم الـ Threading
- برمجة الإنترنت Web Programming

مقدمة

نحاول في هذا الفصل تجميع الخصائص العامة للغات البرمجة مع توضيح كل قاعدة وعنصر من عناصر اللغة مع عرض اختلافها من لغة لأخرى .

واذا قمت بقراءة هذا الفصل أكثر من مرة وبذلت فيه مجهود سوف يوفر عليك الكثير حيث أنه يجمع معظم خصائص اللغات التي سوف نتناولها في الفصول التالية بالتفصيل لذلك حاول قراءته أكثر من مرة .

البرنامج في اللغات

يأخذ البرنامج في معظم اللغات شكل وتركيب محدد وخاصة في اللغات الاولى التي كانت تكتب في بيئة نظام التشغيل DOS حيث كان البرنامج يكتب وحدة واحدة وذلك مثل لغة C .. وفيما يلي أمثلة لتركيب البرنامج في اللغات المختلفة.

1- البرنامج في لغة C

يأخذ البرنامج التركيب الموضح في السطور التالية

```
1:#include <stdio.h>
2: main()
3: {
4: printf("\n Welcome With First Program in C Language\n");
5:        }
```

في هذه السطور يظهر الهيكل الاساسي لبرنامج لغة C ويتكون مما يلي :

السطر #include "stdio.h" ويقوم بفتح (الاشتمال على) الملف stdio.h الذي يحتوى على تعريفات الدوال المهمة والمستعملة خلال البرنامج.

السطر main() فيه الدالة الرئيسية للبرنامج main() وبين أقواسها.

السطر { :هو قوس بداية البرنامج الذي يكتب بعد تعريفات المتغيرات وسطور البرنامج والقوس } لإغلاق الدالة الرئيسية.

لغة Java

برنامج من نوع تطبيق Java Application ، يأخذ التطبيق فى لغة Java الشكل التالى:

```
1  package mypackage3;
2  public class Ex1
3  {
4   public Ex1()
5   {
6   }
7  public static void main(String[] args)
8   {
9   new Ex1();
10    System.out.println("Welcome With First Java Program");
11  }
12 }
13
```

هذا هو الشكل الأساسي لبرنامج Java وفيه يبدأ البرنامج

بالأمر Package الذى يخبر البرنامج بوضع فصائل البرنامج فى حزمة (مجلد) بالاسم mypackage3 ثم يبدأ بفصيلة class بالاسم Ex1 ثم قوس لغة الـ C الشهير) ثم الدالة ()main التى تشابه الدالة الرئيسية فى برنامج ال C وداخل أقواس الدالة ()main تكتب السطور المطلوب تنفيذها ثم يغلق قوس الدالة ()main الذى فتح بالقوس وكذلك يغلق قوس الفصيلة class الذى فتح بالقوس {

هذا بالاضافة لكتابة الاوامر فى أحداث التطبيق مثل Button_clciked وغيره فى حالة التطبيق فى بيئة النوافذ Windows application وهذا هو التطبيق الغالب لذلك سوف نتناول الأحداث Events ومعناها وكيفية توظيفها.

البرنامج فى لغة C#

فى حالة كتابة برنامج فى بيئة نظام التشغيل DOS يقترب تركيب البرنامج فى لغة C# من نفس تركيب البرنامج فى لغة Java ويسمى Console Application ويأخذ الشكل الموضح فى السطور التالية:

```
using System;

namespace ConsoleApplication10
{
    /// <summary>
    /// Summary description for Class1.
    /// </summary>
    class Class1
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            //
            // TODO: Add code to start application here
            //
        }
    }
}
```

يبدأ بالأمر using الذى يخبر البرنامج بالإشارة إلى المكتبة system لإمكانية استعمال الفصائل المعرفة بها ثم يبدأ بفصيلة class بالاسم class1 ثم قوس لغة الـ C الشهير { ثم الدالة ()main التى تشابه الدالة الرئيسية فى برنامج ال C وداخل أقواس الدالة ()main تكتب السطور المطلوب تنفيذها ثم يغلق قوس الدالة ()main الذى فتح بالقوس وكذلك يغلق قوس الفصيلة class الذى فتح بالقوس {

هذا بالإضافة لكتابة الأوامر فى أحداث التطبيق مثل Button_clicked وغيره فى حالة التطبيق فى بيئة النوافذ Windows application وهذا هو التطبيق الغالب لذلك سوف نتناول الاحداث Events ومعناها وكيفية توظيفها.

البرنامج فى لغة VB.Net

يأخذ البرنامج فى VB.NET نفس تركيب البرنامج فى لغة C# فى حالة كتابة برنامج فى بيئة نظام التشغيل DOS ويسمى Console Application هذا بالإضافة لاستعمال الأحداث Events فى حالة إعداد برامج تعمل فى بيئة Windows.

البرنامج فى Oracle PL/SQL

تحتوى Oracle على لغة برمجة خاصة بها تسمى PL/SQL ويسمى البرنامج فيها بلوك Block ويأخذ الشكل التالى :

```
DECLARE
/* Declare variables which will be used in SQL statements */
v_EName          VARCHAR2(30) ;
v_EmpNo          Number := 7788;
BEGIN
/* Select Data from  Emp table. */
SELECT Ename
INTO  v_EName
FROM emp
WHERE EmpNo = v_EmpNo;
END;
```

فى هذه السطور :

- فى السطر رقم 1 يبدأ البلوك بالكلمة المحجوزة DECLARE ومعناها إعلان وهى تسبق الاعلان عن المتغيرات المستعملة فى البرنامج.
- فى السطر رقم 3 و4 يتم الإعلان عن متغيرين الأول من النوع VARCHAR2 أى حرفى والثانى من النوع Number مع إعطائه قيمة ابتدائية.
- فى السطر رقم 5 تبدأ العمليات داخل البلوك بالكلمة المحجوزة BEGIN وهى تسبق العمليات المطلوب إجراتها فى البلوك.
- فى السطور 7 و 8 يتم كتابة الأوامر المطلوب تنفيذها.
- فى السطر رقم 11 يتم انهاء البلوك بالكلمة المحجوزة END;.
- وبالتالى ملخص البلوك هو كلمة DECLARE تسبق الاعلان عن المتغيرات التى يتم الاعلان عنها بين DECLARE وbegin بين الكلمة المحجوزة BEGIN والكلمة المحجوزة END; يتم كتابة سطور العمليات.

من خلال الفقرات السابقة نكون قد قمنابعرض شكل البرنامج فى أكثر من لغة حتى يكون لديك تصور عن ذلك وأن الفروق بسيطة:

المتغيرات Variables و الثوابت Constants فى اللغات

المتغير يستعمل لحفظ قيمة رقمية أو حرفية أو تاريخ أو غيره لاستعمالها أكثر من مرة داخل البرنامج ولا يخلو برنامج من استعمال المتغيرات فعن طريق المتغيرات تستطيع تكوين التعبيرات Expressions وتستطيع تمثيل بيانات البرامج و والمتغيرات تأخذ أنواع كثيرة وتختلف أنواع البيانات الى حد ما من لغة الى لغة ولكن بدرجة بسيطة ،ربما فقط فى المسميات ، فمثلا فى لغة VB النوع Integer هذا النوع فى جميع لغات الـ C (C,C++,Java,C#) يسمى int وهكذا ، مع ملاحظة استعمال نفس الأنواع تقريبا فى لغات الـC التى أطلق عليها C Like وهى C,C++,Java,C#,

الإعلان عن المتغيرات varible declaration

الإعلان عن المتغير هو إعطاء فكرة مسبقة لمترجم البرنامج بأن هناك متغير varible أو متغيرات بنوع معين ويجب الإعلان عن المتغير قبل استخدامه حسب اللغات كما يلى :

فى لغة VB.NET

يتم الإعلان عن المتغير بالصورة التالية :

```
Dim a1 As Integer
```

و في هذا السطر يتم استعمال كلمة DIM في الإعلان عن متغير بالاسم A1 من نوع INTEGER مع استعمال AS قبل النوع وكأنك تقول هناك متغير بالاسم a1 من النوع Integer.

فى لغات الـ C

كما ذكرنا لغات C Like هى لغات C,C++,Java,C# يتم الاعلان بطريقة واحدة وهى كما فى السطور.

```
int  a,b,c;
float k,l,m;
double a1,b2;
```

فى هذه السطور :

يتم كتابة نوع البيانات أولا وليكن int كما فى السطر الأول ثم كتابة أسماء المتغيرات وليكن

a,b,c ، وفى هذه الحالة يكون معنى السطر المتغيرات a,b,c من النوع int أى تقبل قيم صحيحة مثل 10,100,66

فى لغة Oracle PL/SQL

يتم الاعلان بكتابة المتغيرات فى جزء الإعلان داخل البلوك Block ويأخذ الشكل التالى :

```
DECLARE
/* Declare variables which will be used in SQL statements */
v_EName          VARCHAR2(30) ;
v_EmpNo          Number := 7788;
BEGIN
/* Select Data from  Emp table. */
SELECT Ename
INTO  v_EName
FROM emp
WHERE EmpNo = v_EmpNo;
END;
```

فى هذه السطور :

- فى السطر رقم 1 يبدأ البلوك بالكلمة المحجوزة DECLARE ومعناها إعلان وهى تسبق الاعلان عن المتغيرات المستعملة فى البرنامج.
- فى السطر رقم 3 و4 يتم الاعلان عن متغيرين الأول من النوع VARCHAR2 أى حرفى والثانى من النوع Number مع إعطائه قيمة ابتدائية.

التحويلات بين الأنواع فى اللغات

بالإضافة لإمكانية الإعلان عن المتغيرات بالمستويات المختلفة واستعمالها توجد امكانية التحويل من نوع إلى نوع إذا كنت بحاجة إلى ذلك وتوجد أكثر من طريقة.

فى لغة VB.NET

1- كلمات التحويل conversion keyword

توجد مجموعة من الكلمات التى توفرها Visual Basic .NET للتحويل إلى الأنواع المختلفة وهى تستعمل بسهولة كما فى الصورة التالية:

```
Dim K1 As Integer
Dim Q1 As Double
K1= 432
Q1= K1
Q1 = Math.Sqrt(Q1)
K1 = CInt(Q1)
```

- في هذه السطور تم الإعلان عن متغيرين الأول من النوع Integer والثاني من النوع Double.
- ثم تم استدعاء الدالة Sqrt لحساب الجزر التربيعي لقيمة المتغير Q1.
- ثم تم وضع قيمة المتغير Q1 في المتغير K1 ولكن بعد وضع المتغير Q1 كمعامل للكلمة CInt(Q1) التي تقوم بتحويله أولا إلى النوع Integer.
- كما يوجد جدول يعرض كلمات التحويل المتاحة في الكتاب الفني لميكروسوفت.

في لغة Java

يتم استعمال طريقتين من التحويل

1- استعمال طريقة Casting

وهو إما تحويل تلقائي كما في السطور التالية

```
byte a=20;
short b;
int c;
long d;
float e;
double f;
b=a;
c=b;
d=c;
E=d;
F=e;
```

أو باستعمال أوامر التحويل .

2- استعمال دوال التحويل في الفصائل

حيث تحتوى الفصائل classes على دوال للتحويل إلى أنواعها مثل الفصيلة Integer التي تحتوى على الدالة ()ParseInt و هكذا .

فى لغة C#

1- دوال التحويل conversion keyword

توجد مجموعة من الدوال الأعضاء بفصائل الأنواع توفرها C# للتحويل إلى الأنواع المختلفة وهى تستعمل بسهولة كما فى الصورة التالية:

```
Int k;
String q="144";
K=INT32.Parse(q)
```

فى هذه السطور تم الإعلان عن متغيرين الأول من النوع Integer والثانى من النوع string

ثم تم وضع قيمة المتغير q فى المتغير K ولكن بعد وضع المتغير q كمعامل للدالة Parse التى تقوم بتحويله أولا الى النوع Integer.

2- استعمال الفصيلة Convert Class

توجد فصيلة Class بالاسم Convert تحتوى على عدد من دوال التحويل التى يمكنك استعمالها لتحويل نوع الى نوع أخر .

دوال الإدخال و الإخراج I/O Functions

كما هو مشهور في جميع لغات البرمجة توجد جمل (دوال) الإدخال و الإخراج I/O Functions التى تستعمل في استقبال قيم من المستخدم و كذلك عرض الرسائل و القيم له .

و بالرغم من اختلاف ذلك شيئا ما مع برمجة النوافذ إلا أنه توجد دوال كما يلي :

فى لغة VB.NET

دوال الإخراج (عرض الرسائل)

الدالة MSGBOX

و تأخذ الصورتين:

```
1-MSGBOX MSG
2-MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

فى لغة C#

نفس الدوال الموجودة فى لغة VB.Net ويظهر ذلك من السطور

```
string v1, tit;
   v1 = textBox1.Text;
   tit = textBox2.Text;
   DialogResult r;
   MessageBoxButtons buttons = MessageBoxButtons.YesNo;
     r = MessageBox.Show(this, tit, v1,buttons,
   MessageBoxIcon.Question, MessageBoxDefaultButton.Button1,
   MessageBoxOptions.RightAlign);
```

بالاضافة لأمر طباعة المخرجات فى بيئة نظام التشغيل DOS كما فى السطر.

```
Console.WriteLine("This is The ConsoleApplication ");
```

فى لغة الـ Java

يتم طباعة المخرجات فى بيئة نظام التشغيل DOS كما فى السطر.

```
System.out.println("Welcome with first Java program..");
```

التكرار Looping

من العمليات المسموح بها فى جميع اللغات عملية التكرار وهى تكرار تنفيذ عملية معينه أو مجموع سطور أكثر من مرة وهى مفيدة بدرجة أنها لا يمكن الاستغناء عنها فى كثير من البرامج.

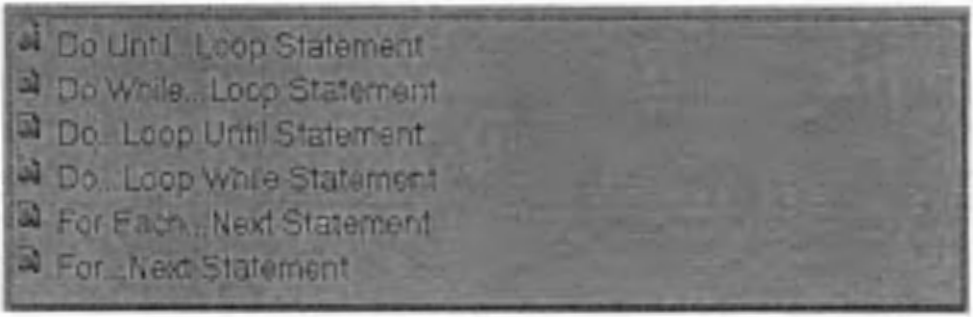
وينقسم التكرار Looping إلى نوعين:

- تكرار معلوم العدد: وفيه معروف عدد مرات التكرار وتستعمل له الجملة for بأشكالها المختلفة.
- تكرار غير معلوم: وأحياناً يسمى تكرار مشروط لأن التكرار يتوقف على شرط معين. والتكرار من أشهر الجمل في جميع لغات البرمجة حيث تسمح بتكرار عملية أو مجموعة سطور أكثر من مرة وكذلك جمل التي تحدد مسار البرنامج و تحقق رغبات المستخدم و هي جمل التحكم في مسار البرنامج حيث نتناول النقاط التالية:
  - جمل التكرار..... , FOR
  - جمل التكرارFor Each
  - الأشكال المختلفة للجملة FOR

- الجملة WHILE ...WEND
- الجملة While
- الجملة do ---- while
- Loop

فى لغة VB.NET

توجد الجمل التالية للتكرار:



جملة FOR

نستخدم جملة FOR لتكرار تنفيذ عملية عدد من المرات معلومة العدد كما فى السطور:

```
1: Dim i
2:       For i = 0 To 20
3:         MsgBox("I=" & i)
4:   Next i
```

و تأخذ أكثر من صورة يمكنك معرفتها بالرجوع لكتاب متخصص فى اللغة .

جملة DO.....WHILE

الجملة DO WHILE تستخدم لتكرار عملية عدد من المرات غير معلومة العدد و لكن يعتمد هذا التكرر على شرط WHILE حيث تأخذ الصورة العامة التالية :

```
Do [{While | Until} condition]
[statements]
[Exit Do]
[statements]
Loop
Or, you can use this syntax:
```

```
Do
[statements]
[Exit Do]
[statements]
Loop [{While | Until} condition]
```

في هذه الصورة :

يتم تكرار ما بين DO و الفرق الآخر LOOP طالما أن الشرط مكتوب أمام كلمة WHILE صحيح فإذا أصبح غير صحيح يتوقف التكرار .

و تركيبها كما في السطور التالية:

```
Do Until False

Loop
```

```
Do While True

Loop
```

```
Do

Loop Until True
```

```
Do

Loop While True
```

```
For Each Item As String In CollectionObject

Next
```

```
For index As Integer = 1 To 10

Next
```

```
    While True

End While
```

في لغات الـ C

تتقارب جمل التكرار بشكل كبير جدا في لغات الـ C like وهي كما أشرنا اللغات C,C++,C#,Java كما يلي:

الجملة for

وتأخذ الصورة العامة التالية :

```
for ([initializers]; [expression]; [iterators]) statement
```

و هي تستعمل في تكرار تنفيذ عملية معينة عدد من المرات .

فمثلا لطباعة كلمة C# 20 مرة سوف نكتب 20 سطر لتحقيق ذلك بدون استعمال جمل التكرار في حين تستعمل جملة تكرار مع سطر واحد لتنفيذ ذلك.

و يتضح ذلك من السطر التالي :

```
for (int i = 1; i <= 5; i++)
Console.WriteLine("C#");
```

في هذه السطور تم استعمال جملة التكرار FOR.

لتنفيذ السطر رقم 2 عدد 10 مرات و هذا ما يوفر علينا كتابة سطر 2 عدد 10مرات .

جملة DO.....WHILE

الجملة DO WHILE تستخدم لتكرار عملية عدد من المرات غير معلومة العدد و لكن يعتمد هذا التكرر على شرط WHILE حيث تأخذ الصورة العامة التالية :

```
do statement while (expression);
```

في هذه الصورة:

يتم تكرار ما بين DO و الفرق الآخر LOOP طالما أن الشرط مكتوب أمام كلمة WHILE صحيح فإذا أصبح غير صحيح يتوقف التكرار .

الجملة while

تستعمل هذه الجملة لتكرار تنفيذ مجموعة من السطور عدد من المرات غير محدد مسبقا ولكن يتوقف على شرط ولذلك يفضل استعمالها في حالة عدم معرفة عدد مرات التكرار وتاخذ الجملة التركيب التالي:

```
while (expression) statement
```

كما توجد foreach في لغة C# وتأخذ الشكل التالي

الجملة foreach

تستخدم هذه الجملة لتكرار اجراء عملية على عناصر مصفوفةArray أو مجموعة Collection وتأخذ الصيغة التالية:

```
foreach (type identifier in expression) statement
```

فى لغة Oracle PL/SQL

أوامر التكرار LOOPS

جمل التكرار هى أشهر جمل فى لغات البرمجة ،وهى مجموعة من الجمل التى تستعمل لتكرار تنفيذ عملية أكثر من مرة ، ولتقريب ذلك إلى ذهنك تصور لو أنك تريد التعويض فى المعادلة التالية:

```
Y=2*x
```

المطلوب التعويض فى هذه المعادلة فى قيمة x من القيمة 0 إلى القيمة 1000 وحساب قيمة y المقابلة لها ،

تخيل لو أنك استعملت الطريقة التقليدية وقمت بالتعويض بهذه القيم بنفسك بالتأكيد سوف يستغرق ذلك منك جهد كبيرا.

لذلك يوجد بديل فى جميع اللغات هو جمل التكرار LOOPS

ويوجد أكثر من أمر تكرار تنفيذ العمليات وهي :

التكرار LOOP---if condition –Exit

التكرار LOOP----Exit when

التكرار while----loop

التكرار For ----loop

التكرار LOOP---if condition –Exit

وفيه يتم تكرار الأوامر التي بينهما بدون توقف ولإيقاف هذا التكرار لابد من إضافة شرط داخل التكرار كما يلى:

```
LOOP                                    -- delimiter
  statement1;                           -- statements
  . . .
 IF condition THEN                      -- EXIT statement
  EXIT;
END IF;
END LOOP;                               -- delimiter
```

جملة التكرار LOOP----EXIT WHEN

يتم استعمال الامر EXIT WHEN بدلا من استعمال IF وهى أسهل وتأخذ الصورة العامة التالية:

```
LOOP                                    -- delimiter
  statement1;                           -- statements
  . . .
  EXIT [WHEN condition];                -- EXIT statement
END LOOP;                               -- delimiter
```

فى هذه الصيغة تم استبدال جملة الشرط IF وأمر الخروج EXIT بالامر EXIT WHEN

التكرار WHILE...LOOP

فى هذه الجملة يتم إضافة شرط التكرار مع While لتكوين تكرار while بدلا من تحديد شرط باستعمال جملة IF أو EXIT WHEN وبالتالى يختلف تركيب التكرار وذلك كما بالشكل:

```
WHILE condition LOOP  <----------  Condition is evaluated at the
  statement1;                      beginning of each iteration.
  statement2;
  . . .
END LOOP;
```

جملة التكرار FOR

هو من أشهر الطرق في جميع لغات البرمجة حيث يستعمل التكرار تنفيذ عملية عدد من المرات معروفة ، ويأخذ الشكل التالى:

```
FOR counter IN [REVERSE]
    lower_bound..upper_bound LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

- ويمكن تغير قيم التكرار بالعكس أي بدلاً من تكرارها من القيمة 1 إلى 1 ويتم تكرارها من 1 إلى 1 وذلك بإضافة كلمة REVERSE كما بالشكل:

FOR MY_VAR IN REVERSE 1—1 LOOP

```
|
END LOOP;
```

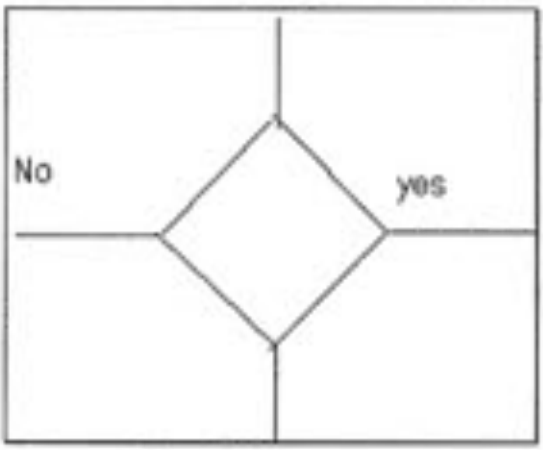
جمل التحكم في مسار البرنامج

في هذه الفقرة نتناول الجمل التالية:

- جملة ..... if
- جملة ....Else....if
- جملة ....End if..... else if ....if
- جملة select case....end select
- جملة GOTO

هذه الجمل هي التي تحدد مسار البرنامج هل يسير في الخط الطبيعي أم يتفرع إلى دالة فرعية و هي التي تحقق رغبات المستخدم حيث لا يمكن أن يكون هناك برنامج بدون جمل تحكم .

و يتضح ذلك من خريطة التدفق الموضحة في الشكل (12-7):



الشكل (12-7)

في هذا الشكل نلاحظ وجود جملة شرط تحدد مسار البرنامج فإذا كان yes يتجه التنفيذ إلي اتجاه معين و إذا كان NO اتجه اتجاه آخر .

و فيما يلي نشرح جمل التحكم :

فى لغة VB.NET

جملة IF

تستخدم جملة If كما هي بدون else إذا كان هناك اختبار تريد اختباره إذا تحقق تنفذ جمل معينة و إلا يسير البرنامج طبيعي و لا ينفذ هذه الجمل و يأخذ ذلك الصورة البسيطة التالية

```
if ... condation ... then
Statements 1
Statements 2
End If
```

في هذه الصورة إذا كان الشرط condation صحيح تنفذ الجمل , statements 2 statements 1 و ينهي ذلك كلمة end If . وهذا صور أخر سوف نتناولها بالتفصيل

فى لغات الـ C-Like

تتشابه الجمل فى هذه اللغات بدرجة عالية جدا كما يلى

جملة IF

تستخدم جملة If كما هي بدون else إذا كان هناك اختبار تريد اختباره إذا تحقق تنفذ جمل

معينة و إلا يسير البرنامج طبيعي و لا ينفذ هذه الجمل و يأخذ ذلك الصور العامة التالية :

```
if (expression)
   statement1
[else
   statement2]
```

في هذه الصورة إذا كان الشرط condation صحيح تنفذ الجمل , statements 2 statements 1 و ينهي القوس)

جملة if else

تستخدم جملة else .. if إذا كان هناك أكثر من احتمال نريد اختياره حيث تأخذ الشكل التالي :

```
if (expression)
   statement1
[else
   statement2]
```

في هذه الصورة يتم اختيار الشرط CONDATION بجملة IF فإذا كان الشرط صحيح تنفذ الجملتين STATEMENT1, STATEMENT2 و إلا (إذا كان الشرط غير صحيح) تنفذ الجمل التي بعد ELSE و هي في الصورة STATEMENT3, STATEMENT4.

تركيب if elseif

يستعمل هذا التركيب عندما يكون هناك أكثر من احتمالين (شرطين) نريد اختيارهما و يأخذ الصورة :

```
if CONDATION
else if CONDATION THEN
else if CONDATION THEN
else
```

في هذه الصورة

إذا تحقق الشرط الأول يختبر الشرط الثاني وهكذا

لاحظ المثال الموجود بالمسار EX12 باسم NESTED_IF

تركيب switch...CASE

يستعمل هذا التركيب محل التركيب السابق و هو اختيار أكثر من حالتين و هو أسهل في التركيب و الاستعمال عن التركيب السابق .

و يأخذ الصورة العامة التالية:

```
switch (expression)
{
   case constant-expression:
      statement
      jump-statement
   [default:
      statement
      jump-statement]
}
```

في هذه الصورة

وهي الصورة التقليدية والتي شرحناها في الصورة السابقة.

جمل الانتقال

توجد مجموعة من الاوامر التي تؤدي الى الانتقال الى سطر معين وهي كما يلي:

- goto
- break
- continue
- default
- return

جملة GOTO

تستخدم GOTO لتغير مسار البرنامج وتوجه تنفيذ البرنامج الى خطوة معينة وان كانت غير مستحبة في عالم البرمجة وتأخذ الشكل التالي:

```
goto identifier;
goto case constant-expression;
```

في لغة Oracle PL/SQL

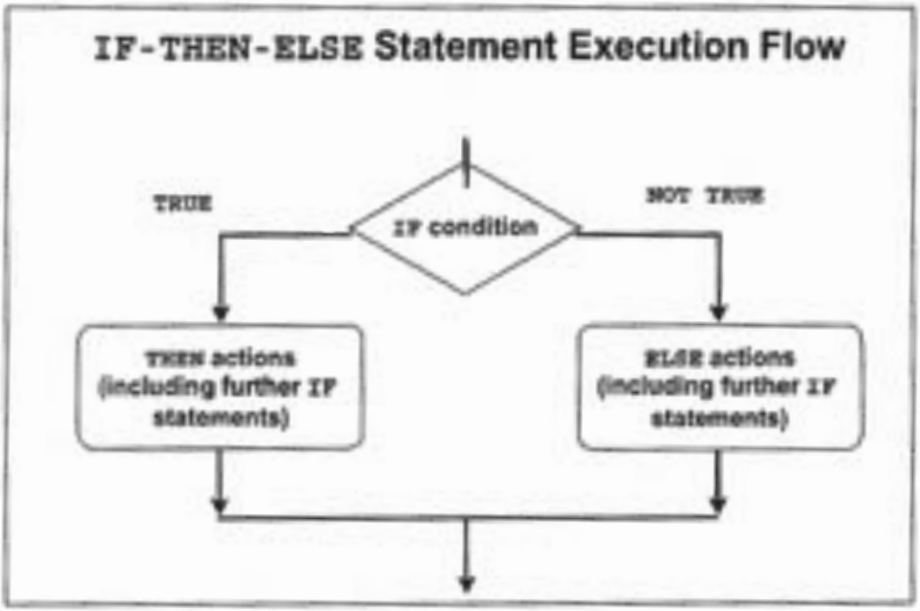
1. الأمر IF

وتأخذ جملة IF الشكل العام

```
IF condition THEN
  statements;
[ELSIF condition THEN
  statements;]
[ELSE
  statements;]
END IF;
```

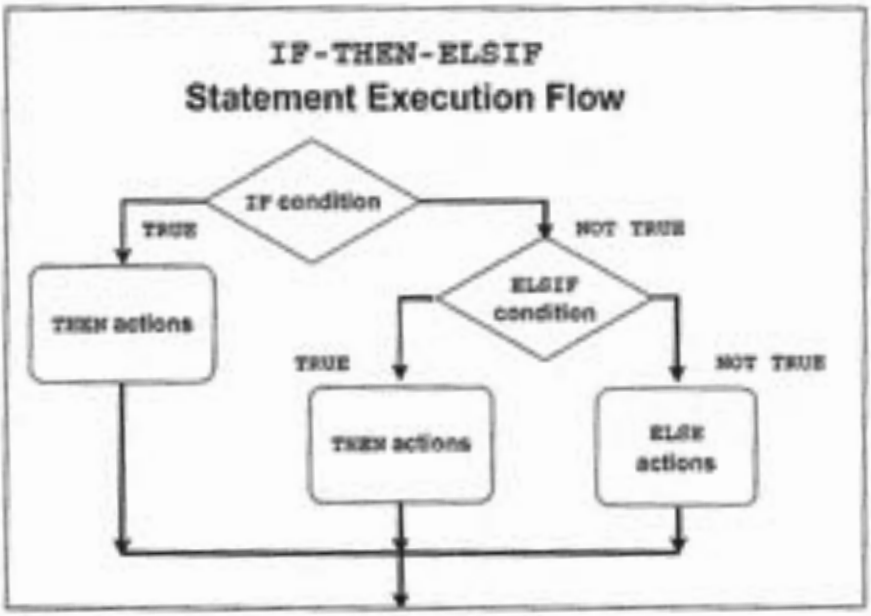
2. الصيغة IF ELSE

وتأخذ الصورة



الامر IF ELSIF

يستعمل هذا الامر ف حالة تعدد الشروط بمعنى انك لديك أكثر من اختيار ويتضح ذلك من الشكل 6



معني الدالة و البرنامج الفرعي FUNCTION & SUB

إذا سبق لك دراسة لغة من لغات البرمجة مثل لغة C و غيرها فأنت تعرف أن أي برنامج يتكون من برنامج رئيسي و برامج فرعية و دوال يتم استدعائها من البرنامج الرئيسي أي أن الدالة و البرنامج الفرعي هما مجموعة من الأوامر التي تكتب داخل اطار اسم برنامج فرعي يسمي دالة FUNCTION أو SUB يتم إنشائه مرة واحدة و يمكن استدعائة أكثر من مرة مما يوفر اعادة كتابة هذه الأوامر أكثر من مرة.

الفرق بين الدالة FUNCTION و البرنامج الفرعي Procedure

الفرق الأساسي بين ال FUNCTION و Procedure (الإجراء) هو أن ال FUNCTION لابد ان تعيد قيمة عند استدعائها مثل دالة حساب AVGأو SIN أو أي دالة من دوال اللغة. بينما يقوم البرنامج الفرعي Procedure (الإجراء) بتنفيذ مجموعة من السطور دون أن تعيد قيمة بينما ينفذ الجمل لتحقيق حدث و سوف يتضح ذلك عند التعامل معها .

التعامل مع البرنامج الفرعي

يتم التعامل مع البرنامج الفرعي SUB أو الدالة خلال مرحلتين هما :

أ) إنشاء البرنامج الفرعي (الإجراء)

ب) استدعاء البرنامج الفرعي (الإجراء)

فى VB.NET

ينشأ البرنامج الفرعى SUB (الإجراء) بالشكل

```
Sub s1()
MSGBOX " THIS LINE CALLED FROM SUB S1 "
End Sub
```

فى هذه السطور يتم انشاء برنامج فرعى من النوع Sub بالاسم S1 يعرض رسالة

ويتم استدعاء هذا البرنامج الفرعى بكتابة اسمه فقط بالشكل

```
S1
```

برنامج فرعي من نوع function (الدالة)

يأخذ البرنامج الفرعي من نوع function نفس خطوات الإنشاء و لكنه يختلف عن النوع sub في أنها لابد أن تعيد قيمة مثل الدوال الجاهزة في اللغة مثل دالة abs و دالة sin و غيرها من الدوال .

وتكتب سطورها كما فى الشكل

```
Function f1(ByVal N1 As Integer, ByVal N2 As Integer)
    Dim s1 As Integer
    s1 = N1 + N2
    Return s1
End Function
```

فى هذه السطر يتم انشاء دالة function بالاسم f1

ويتم استدعاء الدالة f1 بالشكل

```
R=F1 (5,7)
MSG BOX " RETURENED VALUE IS : " & R
```

تلاحظ إرسال معاملات لاننا أنشأنا الدالة بمعاملات وكذلك ربد من وضع القيمة المرتجعة فى متغير وفى هذه الحالة المتغير R

فى لغات C-Like

كما ذكرنا لغات C-Like هى اللغات التى تلتزم بقواعد لغة C فى كتابة الاوامر وهى C,C++,C#,Java تنشأ الدالة function كما فى السطور.

```
        void FirstFunc()
        {
        MessageBox.Show ("First Function.....");
        }
```

فى هذه السطور تم انشاء دالة بالاسم FirstFunc وتنحصر سطور الدالة بين الاقواس {} وكلمة void تعبر عن نوع القيمة المرتجعة وفى هذه الحالة void تعنى ليس هناك قيمة مرتجعة وبالتالى تقابل هذا النوع من الدوال نوع الإجراء Procedure فى لغة VB.NET ولكن إذا كانت الدالة تعيد قيمة يحدد نوع هذه القيمة كنوع للدالة ، ويتم استدعاء الدالة بكتابة اسمها ومعاملاتها إذا كان لها معاملات كما فى الشكل.

```
        private void button1_Click(object sender, System.EventArgs e)
        {
        FirstFunc();
        }
```

فى Oracle PL/SQL

يتم إنشاء الإجراء الفرعى Procedure كما فى السطور

```
    PROCEDURE DEPT_SEARCH IS
    BEGIN
      SELECT DNAME INTO :TEXT_ITEM6  FROM DEPT WHERE
DEPTNO=:TEXT_ITEM5;
    END;
```

تلاحظ أنها بدئت بالكلمة PROCEDURE ثم اسم الإجراء وتنحصر مكونات الإجراء بين الكلمتين BEGIN و END وينفذ الإجراء بكتابة اسمه كما فى السطر

DEPT_SEARCH;

إنشاء دالة CREATING PL/SQL FUNCTION

يتم ذلك كما فى السطور

```
1: Create function MySum (No1 Number,No2 Number )
2: RETURN NUMBER
3: IS
4: S NUMBER;
5: BEGIN
6: s=No1+No2;
7:  9: RETURN s;
1 : END;
```

المصفوفات Arrayes

في هذا الدرس نشرح موضوع المصفوفات Arrays بأنواعها وكيفية تمثيلها واستخدامها و ذلك من خلال النقاط التالية :

- **معنى المصفوفات و استخدامها .**
- **أنواع المصفوفات Arrays types.**
- **مصفوفة بعد واحد One Dimension.**
- **مصفوفة بعدين Two Dimension .**
- **خصائص المصفوفات**

معنى المصفوفة

كما تعلم من الرياضيات أن المصفوفة Array عبارة عن مجموعة من القيم (العناصر) من نفس النوع و تأخذ أسم و تأخذ عدد للعناصر .

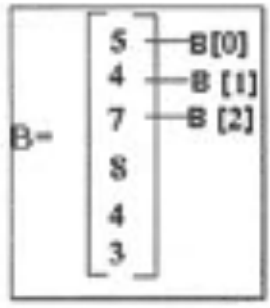
والمطلوب كيفية تمثيلها و استعمالها وكيفية التعامل مع عناصرها و للمصفوفة تطبيقات كثيرة لا تصلح إلا بها مثل عمليات البحث عن قيمة ضمن مجموعة قيم Search

ومثل عمليات ترتيب مجموعة من القيم سواء تصاعدى أو تنازلى Sorting ومثل ايجاد اكبر قيمة ضمن مجموعة قيم Max وايجاد أقل قيمة Min والكثير من التطبيقات وخاصة فى مجال الدرسات العليا والابحاث

أنواع المصفوفات :

(أ) مصفوفة بعد واحد One Dimension كما بالشكل (9.1)

A= [5 4 7 8 4 3]



الشكل (1-9)

- المصفوفة A مصفوفة صف One Row وهي ذات بعد واحد حيث لا يوجد صفوف وأعمدة بل هي صف واحد فقط .
- المصفوفة B مصفوفة عمود واحد One Column لذلك يسمي كلا منهما مصفوفة البعد الواحد و يتم ترتيب العناصر بأسم المصفوفة و ترتيب العنصر مثل A[1] ، B[2] وهكذا.

(ب) مصفوفة متعددة الأبعاد :

وفيها يوجد أكثر من صف Row وأكثر من عمود Column كما بالشكل (2-9) .

$$C=\begin{bmatrix} 5 & 70 & 10 \\ 20 & 40 & 5 \\ 30 & 40 & 7 \\ 5 & 30 & 5 \\ 9 & 40 & 6 \\ 30 & 10 & 7 \end{bmatrix}$$

الشكل (2-9)

مصفوفة البعد الواحد One Dimension :

و يتم التعامل مع هذه المصفوفة بالخطوات التالية :

1- الإعلان عن المصفوفة

2- استعمال عناصر المصفوفة بملئها بالقيم أو استعمال القيم أو طباعة القيم

في لغة VB.NET

الإعلان عن المصفوفة

- يتم الإعلان عن المصفوفة كما يتم الإعلان عن المتغيرات و بنفس درجات الإعلان فيمكن الإعلان عنها و كأنها محلية Local داخل برنامج فرعي Sub أو على مستوي form أو على مستوي التطبيق كما يحدث مع المتغيرات و ذلك حسب المكان و بالصورة التالية :

```
Dim a(10) As Integer
```

أو يتم استعمال الصورة الحديثة التالية

```
Dim Arr As Integer()=New Integer(10) {}
```

وطالما وجدت مصفوفة لابد من وجود كود التكرار for للتعامل مع عناصر المصفوفة كما في السطور:

```
1. Dim a(10) As Integer
2. Dim i As Integer
3. For i = 0 To 10
4. a(i) = i
5. Next
6. For i = 0 To 10
7. Console.WriteLine(a(i))
8. Next
```

في هذه السطور يتم الاعلان عن مصفوفة ثم استعمال for لملئ عناصر المصفوفة ثم استعمال for لطباعة عناصر المصفوفة.

في لغات C-Like

يتم الاعلان عن المصفوفة كما في السطور:

```
int[] myArray = new int [5];
```

في هذه السطور تم الاعلان عن مصفوفة بالاسم myArray من النوع int وعدد عناصرها 5 ويتم التعامل مع المصفوفة بنفس الطريقة السابقة ، ألا إن لغات C-Like تستعمل الأقواس {} مكان الكلمات next لتحديد بداية ونهاية البلوك.

مفاهيم البرمجة بواسطة الأهداف OOP

تعتبر مفاهيم البرمجة بواسطة الأهداف OOP من أهم مفاهيم البرمجة الحديثة حيث أصبحت معظم لغات البرمجة تطبقه وبناء الفصائل classes هو أساس البرمجة بواسطة الأهداف OOP وفي هذا الفصل نشرح الخطوات العملية لبناء الفصائل وذلك من خلال النقاط التالية :

- مفاهيم البرمجة بواسطة الأهداف OOP
- انشاء واستعمال فصائل Classes
- أضافة دوال أعضاء adding methods to class
- فصائل بدون دوال بناء classes without constructors
- دوال البناء في الفصيلة constructors
- استعمال أداة designer في التعامل مع الفصائل classes
- إنشاء خصائص للفصيلة properties
- إضافة خصائص قراءة أو كتابة فقط read-only and write- only

مفاهيم البرمجة بواسطة الأهداف Object Oriented Programming

تعتبر البرمجة بواسطة الأهداف (object oriented programming) من الاتجاهات الحديثة للبرمجة ومن أول اللغات التي طبقت هذا المفهوم لغة C++ وكذلك من اللغات التي تلتزم بمفهوم OOP لغة JAVA ولغة VB.NET زادت من اهتمامها بمفهوم OOP بل وتعتبر ذلك من التعديلات المهمة والتي زادت من قوة VB. حيث نتناول النقاط التالية :

- معنى البرمجة بواسطة الأهداف Object Orient Programming
- معنى الفصيلة class
- ما هي دالة البناء و دوال الهدم constructors & destructors
- ما هي خاصية التوريث inheritence
- ما هي خاصية over loading

معنى البرمجة بواسطة الأهداف OOP(Object oriented programming

تبني فكرة البرمجة بواسطة الأهداف (OOP) على استعمال الهدف (Object) كوحدة برمجة بمعنى انه بدلاً من استعمال الدوال والأوامر لبناء البرنامج مما يضطر المبرمج لإعادة كتابة الأوامر كل مرة لتحقيق فكرة معينة وهذه كانت فكرة البرمجة التقليدية .

ولكن أتت البرمجة بواسطة الأهداف (OOP) لتجعل وحدة بناء البرمجة كبيرة وهي هدف Object أو فصيلة class وبالتالي يتم إعداد مجموعة من الفصائل العامة classes التي تلبي معظم متطلبات إعداد برنامج والتي يحتاجها المبرمج حتى ان بعض المبرمجين يشبه البرمجة بواسطة الأهداف بالبناء باستعمال المباني الجاهزة والبرمجة بالطريقة التقليدية القديمة تشابه البناء باستعمال الأدوات الأولية وبالتالي الفرق بينهما في السرعة كبيرة جدا

معنى الفصيلة class

الفصيلة class هي أساس البرمجة بواسطة الأهداف (OOP) وهي التي يبنى عليها البرنامج وأخذت فكرة الفصيلة CLASS من الواقع فكل عنصر من عناصر الحياة عبارة عن فصيلة class ، فأنت تستطيع أن تطلق على جميع السيارات إنها من فصيلة CAR مع بعض الاختلافات ،ويمكنك ان تطلق على الطيور فصيلة Bird أي طائر وهكذا تنتمي جميع العناصر الى فصائل classes ، وكل فصيلة تستطيع تمثيلها بعنصرين هما البيانات والدوال (data, methods)، فمثلاً فصيلة الموظف Employee بياناتها هي بيانات الموظف العامة مثل كود الموظف ، اسم الموظف ،عنوان الموظف ،تليفون الموظف ، وباقي بياناته ، وكذلك الدوال (methods) هي دوال تحقيق العمليات التي يمكن أن تتم على الموظف مثل : عملية إضافة موظف جديد ، وحذف موظف موجود ،تعديل بيانات موظف وجميع العناصر يمكن تمثيلها بهذه الطريقة.

ما هي خاصية التوريث inheritance

معنى خاصية التوريث (inheritance) هو توريث فصيلة Base class قديمة موجودة بالفعل لفصيلة class جديدة عند إنشائها بحيث يضاف تركيب الفصيلة Base class القديمة من بيانات و دوال إلى الفصيلة الجديدة new class ثم تكمل عليها في الفصيلة الجديدة.

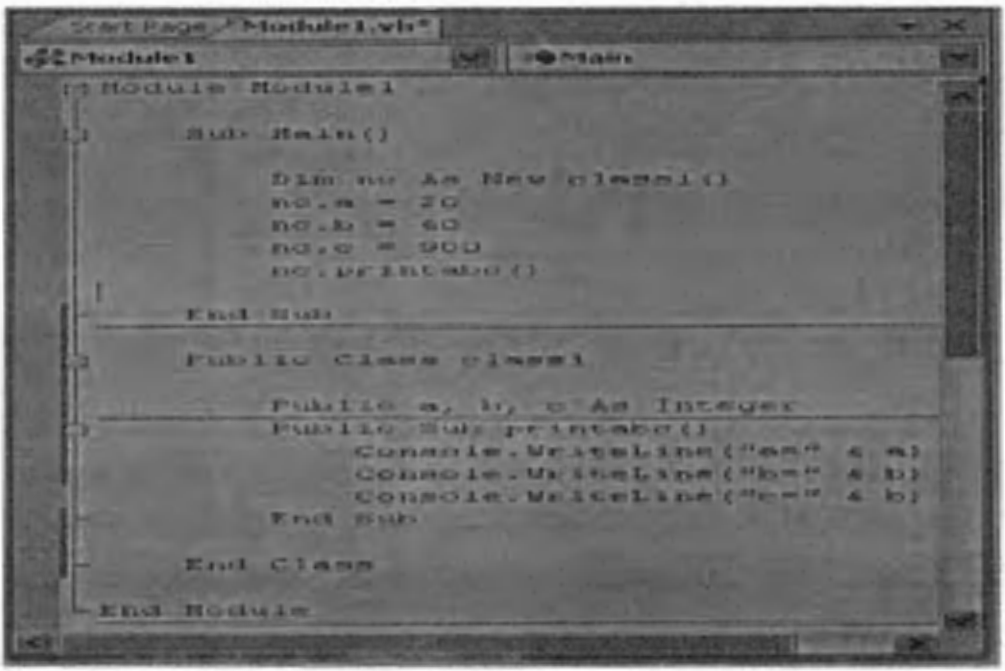
ما هي الخاصية Overloading

معنى خاصية **overloading** : هو إمكانية إنشاء اكثر من دالة Method بنفس الاسم مع تغيير عدد المعاملات Parameters مثلاً ،وهذا يفيد بإنشاء اكثر من دالة لنفس الوظيفة بمعاملات مختلفة بنفس الاسم ، مثل أنشاء دالتين بالاسم ( )find cust إحداهما تأخذ معامل رقمي هو كود الموظف ،والثانية تأخذ عبارة حرفية هي اسم الموظف وفي الحالتين تبحث الدالة ،و بالتالي يشعر المبرمج كأنها دالة واحدة ولكنها دالتين يتم استدعاء كل واحدة تلقائياً حسب المعامل المرسل لها.

ومن أهم ما تحب توضيحه في هذا الفصل شكل البرنامج باستعمال أسلوب OOP

في لغة VB.NET

يأخذ الشكل التالي



في هذا الشكل تم إنشاء فصيلة class باستعمال كلمة class وهي الكلمة المستعملة في جميع اللغات لإنشاء فصيلة.

ثم يلى ذلك أسم الفصيلة وهى فى هذه الحالة class1 ثم يلى ذلك محتويات الفصيلة من متغيرات ودوال وهو تركيب أى فصيلة فى أى لغة (متغيرات variables و دوال methods)

ثم يتم التعامل مع الفصيلة بتعريف متغير هدف object من الفصيلة كما فى السطور حيث تم تعريفه بالاسم nc

ثم التعامل مع مكونات الفصيلة من خلال متغير الهدف.

فى لغات C-Like

تأخذ فكرة انشاء الفصيلة class نفس الفكرة السابقة مع استعمال الاقواس كما هو معتاد كما فى الشكل:

تلاحظ أنها نفس القواعد السابقة.

مع ملاحظة ان القواعد واحدة والاختلافات بسيطة وبالتالى يمكنك تعلم أكثر من لغة حيث أن القواعد الاساسية واحدة.

معالجة الاخطاء Error Handling

في هذا الفصل نعرض موضوع الأخطاء الوارد حدوثها مع المبرمج ومع مستخدم البرنامج وذلك من خلال النقاط التالية:

- الأنواع المختلفة للأخطاء : Bugs
- الخطأ الهجائي Syntax error
- الخطأ أثناء التشغيل Runtime Error
- ON ERROR GOTO-
- ON ERROR RESUME NEXT-
- الخطأ المنطقي Logic Error
- طرق حل الأخطاء :
- طرق تلافي خطأ أثناء التشغيل
- طرق تتبع الخطأ المنطقي ومعرفة سببه
- استعمال الشاشات المختلفة
- استعمال التركيب الجديد Try.....Catch

عند القيام باعداد برنامج يصادفك أخطاء ما أثناء الإعداد أو بعد توزيع البرنامج للاستخدام ، ومن عوامل نجاح البرنامج عدم ظهور أي أخطاء ، وفيما يلي توضح أنواع الأخطاء وطرق تلافيها .

أنواع الأخطاء :

1ـ الخطأ الهجائي Syntax Error

وهو أبسط أنواع الأخطاء حيث يظهر عند كتابة نصوص البرنامج وهو خطأ في الحروف الهجائية للأوامر المعرفة للغة أو خطأ في قواعد اللغة فمثلاً عند كتابة جملة For بدون Next أو العكس.

- وعند كتابة جملة If بدون End If.
- عند كتابة أمر MSGBOX ناقص حرف وهكذا.
- وهذا الخطأ يظهر سريعاً ويسهل علاجه.

النوع الثانى : الخطأ أثناء التشغيل : Runtime Error

لا يظهر هذا الخطأ أثناء أعداد البرنامج الا إذا قمت بتجربة البرنامج بينما يظهر للمستخدم نتيجة قيام المبرمج بكتابة أوامر تحقق عملية غير محققة أثناء التنفيذ فمثلاً:

أ- عند صدور أمر فتح ملف من مشغل أقراص (A,B,...) والمشغل غير جاهز تظهر رسالة خطأ RunTime Error ويتقطع البرنامج ويعود الى بيئة التشغيل.

ب- عند صدور أمر فتح قاعدة بيانات غير موجودة وهكذا ويعتبر هذا النوع من الأخطاء التى تعيب البرنامج فعندما يظهر للمستخدم الخطأ يقطع البرنامج ويخرج وتعتبر هذه عيوب فى البرنامج ، وبالتالى يجب تلافى هذا الخطأ والا يتقطع العمل فى البرنامج فمثلاً يمكن التأكد من وجود القرص قبل تنفيذ أمر فتح الملف أو وجود قاعدة البيانات أو غيره ومن أشهر هذه الأخطاء القسمة على القيمة صفر Divid By Zero حيث تعطى خطأ Over Flow ولتوضيح ذلك تابع المثال التالى:

طرق علاج الخطأ

توجد أكثر من طريقة لعلاج هذا النوع من الأخطاء حسب أهمية العملية وهى :

1- استعمال الجملة ON Error Resume Next

2- استعمال الجملة ON Error GOTO

3- استعمال مصفوفة الأخطاء

4- استعمال التركيب الجديد Try.....Catch

استعمال التركيب الجديد Try.....Catch

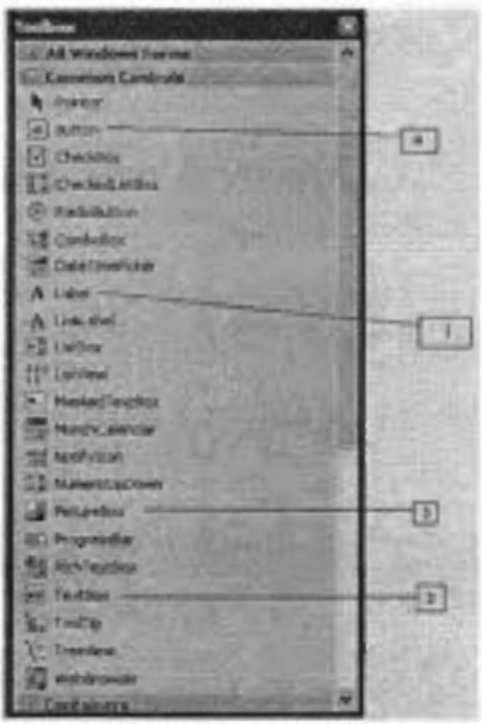
هذا الأسلوب يعتبر جديد فى الاصدار Vb.Net ولكنه كان موجود فى اللغة VC++ وكذلك لغة Java وياخذ الشكل.

```
Try
Statment1
Statment2

Catch Exception1
Action1
```

```
Catch Exception2
Action2

Catch Exception3
Action3
End Try
```

- فى هذه السطور نبدأ بالكلمة Try بمعنى حاول تنفيذ الجمل التالية وهى Statment1,Statment2 فاذا تم تنفيذهم بنجاح اذهب الى End Try وتابع البرنامج
- واذا ظهرت مشاكل اذهب الى مجموعة جمل Catch (أى امسك) التى تتعامل مع الخطأ الناتج.
- فاذا كان الخطأ الناتج من النوع Exception1 نفذ Action1 و كان الخطأ الناتج من النوع Exception2 نفذ Action2

استعمال مفهوم الـ Visual Programming



من المفاهيم الحديثة للغات البرمجة Visual Programming وفيه يتم استعمال أدوات مرئية تسهل البرمجة كثير وتترك للمبرمج التركيز فى منطق البرنامج حيث توفر جميع اللغات بمربع الأدوات ToolBox والشكل يعرض مربع الأدوات و على كل أداة رقم يشير إليها. وهذه هى الأدوات.

مقدمة سريعة عن أوامر قواعد البيانات SQL

فى الفصل الأول قدمنا فكرة عن لغة قواعد البيانات SQL وسوف نتناولها بتفصيل أكثر فى فصل منفصل.

مفاهيم وطرق الاتصال والتعامل مع قواعد البيانات

أصبح التعامل مع قواعد البيانات من الموضوعات المهمة جدا وخاصة فى الوطن العربى ، ولا يمكن تعلم البرمجة فى الوطن العربى بدون تعلم التعامل مع قواعد البيانات وإعداد تطبيقات قواعد البيانات.

لذلك عليك بتعلم أساسيات قواعد البيانات وأوامر SQL وبعض أدوات قواعد البيانات مثل Access,SQL Serverm,Oracle,MySQL.

فى لغة VB.NET

استعمال أوامر الاتصال المباشر Direct Connection

يوفر VB.Net مجموعة من الفصائل Classes التى توفر امكانية استعمال أوامر الاتصال المباشر Direct Connection بقاعدة البيانات حيث تستطيع فتح قاعدة البيانات واصدار الاوامر المختلفة لها مثل الفصيلة Connection class والفصيلة Command class وفصائل أخرى سوف نتناولها بالتفصيل.

تعريفات

تحقيق عمليات قواعد البيانات

من الضرورى تحقيق العمليات الاساسية لقواعد البيانات وهى: الاضافة Insert والحذف Delete والتعديل Update والبحث ويتم ذلك بسهولة بتغيير جملة الـ SQL التى تمثل العملية المطلوبة ، وقد تم شرح جمل SQL المسؤلة عن هذه العمليات.

أهمية التقارير ومولد التقارير Crystal Report

تعتبر التقارير من أهم أجزاء برنامج قواعد البيانات ، فهي الصورة التي يراها المستفيد من البرنامج والتي تخرج فيها مخرجات البرنامج ، فمثلاً عند تصميم برنامج لمتابعة حسابات العملاء لابد من توفير مجموعة من التقارير منها :

- تقرير بحسابات العملاء خلال فترة.
- تقرير بحسابات عميل خلال فترة.
- تقرير بأرصدة العملاء وتقرير برصيد عميل معين ، وهكذا .

وهذه التقارير هي التي تعطي قيمة للبرنامج ، لأن البرنامج بدون تقارير يعتبر بلا فائدة لأنه ينفذ العمليات ولا يعطي النتائج ، وبالتالي يصبح وسيلة لحفظ البيانات فقط .

وكانت عملية تصميم التقارير قبل برمجة النوافذ تتطلب جهد من المبرمج ، حيث يقوم المبرمج بتصميم التقرير باستخدام أوامر البرمجة كما يهتم بضبط الهوامش وغيرها من العمليات الصعبة ، ولكن مع ظهور النوافذ ظهرت برامج منفصلة لتصميم التقارير ثم ربطها بالبرنامج ، ومن هذه البرامج برنامج مولد التقرير المسمى Crystal Report الذي يوفر معظم العمليات المطلوبة لتنظيم التقارير مع إمكانية ربطه بالبرنامج والتقرير الذي يتم تصميمه باستخدام Crystal Report يمكن استدعاءه من برنامج مصمم باستخدام لغة VC++ أو باستخدام لغة Visual Basic مع مراعاة الاصدار فإذا سبق لك كتابة برامج باستخدام لغة VB فبالطبع تعرف كيفية استعمال Crystal Report في تصميم التقارير المطلوبة .

خطوات تصميم برنامج به تقارير

1. تصميم البرنامج المطلوب وفي هذه الخطوة يتم تصميم البرنامج وتحقيق عملياته وترك عملية التقارير لآخر خطوة .
2. تصميم التقارير المطلوبة وحفظها في فهرس البرنامج .
3. إضافة أداة استدعاء التقارير إلى البرنامج Crystal Report Control
4. ربط أداة التقارير بالتقرير المطلوب إظهاره مع ضبط الخصائص .
5. كتابة أوامر استدعاء التقرير في البرنامج.

نظرية اعادة الاستخدام Reusability والمكتبات في Visual Studio.Net

المكتبة class Lib

من المفاهيم المتقدمة هو إمكانية إنشاء مكتبة فصائل class Lib تحتوي على مجموعة من الفصائل classes كثيرة الاستعمال بالنسبة لك أو بالنسبة لفريق العمل ووضعها في مكتبة تترجم إلى ملف بالامتداد dll (Dynamic Link Lib) ثم استعمال هذه المكتبة من قبل

أى برنامج جديد لك أو لفريق البرمجة وهذا الإسلوب يوفر الكثير حيث يوفر إعادة كتابة هذه الفصائل classes أكثر من مرة ، كما يوفر إعادة ترجمة هذه الفصائل أكثر من مرة وتصحيح الأخطاء حيث تقوم بترجمتها مرة واحدة وتصحيح الاخطاء مرة واحدة ثم استعمالها كما تشاء كما أنها توفر صورة غير نصية من المكتبة DLL بحيث يمكن استعمالها دون الإطلاع على نص الأوامر أو التعديل فيها.

موضوع الرسم Graphics

من الموضوعات الجذابة والتى تضيف للبرنامج الكثير من المؤثرات التى قد تحتاجها فى بعض التطبيقات وتوفر معظم اللغات أدوات ودوال للتعامل مع عناصر الرسم.

فى Visual Studio.Net

التعامل مع الأداة Picture Box

يمكن عرض صورة فى الأداة Picture Box ولكن فى وقت التصميم وذلك بتوقيع أداة Picture Box على الـ Form ثم عرض الخصائص F4 واختيار الخاصية Image ثم اختيار الصورة المطلوبة ، ويمكن عرض صورة فى الأداة Picture Box باستخدام أوامر البرمجة وذلك بكتابة الصيغة التالية:

```
PictureBox1.image= New BitMap(filename)
PictureBox1.image= New Metafile(filename)
```

ونستخدم الدالة ()BitMap إذا كنا نتعامل مع صور بالامتداد bmp.

ما هى الفصيلة Graphics

توفر هذه الفصيلة إمكانيات الرسم Graphics من إنشاء عنصر رسم Graphics Object مع توفير دوال الرسم المطلوبة.

ويتم استعمال الفصيلة Graphics بتعريف هدف Object منها كما فى السطور التالية:

```
1. Dim MyGraphics as Graphics
2. MyGraphics=Me.CreateGraphics
```

- فى السطر رقم 1 تم الإعلان عن المتغير MyGraphics كهدف Object من نوع الفصيلة Graphics.

- فى السطر رقم 2 تم إنشاء هدف الرسم Graphics Object مع المتغير Me الذى يشير إلى الـ Form والإشارة إلى هذا الهدف بالمتغير المعرف فى السطر رقم 1 MyGraphics وبهذا عند استدعاء دوال الرسم مع الهدف MyGraphics يتم الرسم على الـ Form ولكن للرسم على أداة اخرى غير الـ Form يتم إنشاء عنصر الرسم مع الأداة الأخرى وسوف نوضح ذلك.

استعمال دوال الرسم Graphics Methods

بعد إنشاء هدف Object رسم يتم استعماله فى استدعاء دوال الرسم.

برمجة الشبكات Networking Programming

ما المقصود ببرمجة الشبكات NetWork Programming

المقصود بذلك اعداد تطبيقات تعمل من خلال الشبكة وأشهرها هو تطبيقات تبادل الرسائل Chating المشهورة وإمكانية إرسال الملفات من جهاز إلى جهاز أخر.

وفى هذا الفصل نتناول فكرة إعداد تطبيقات للشبكات NetWork Programming بأكثر من طريقة وفيها نقوم بإعداد نوعين من التطبيقات.

الأول: تطبيق خادم Server وهو التطبيق الذى يستقبل الاتصال من الجهاز المرسل كما يستقبل منه الرسائل.

الثاني: تطبيق مرسل Client وهو التطبيق الذى يقوم بالاتصال بالجهاز الخادم Server ويرسل و يستقبل منه الرسائل.

وتوفر اللغات إمكانية تحقيق تطبيقات الشبكات كما سنتناول فى فصول لاحقة.

مفهوم الـ Threading

ما هو Threading

هى طريقة تسمح بتنفيذ أكثر من عملية Task فى وقت واحد كل عملية تنفذ فى Thread مع إمكانية التحكم فى هذه العمليات مثل إيقاف الـ Thread مؤقتا أو نهائيا.

ومثال لذلك تطبيق الإنترنت الذى يأتيه أكثر من طلب إذا أمكن التعامل مع أكثر من طلب فى نفس الوقت زادت كفاءة التطبيق ويتم ذلك بإنشاء أكثر من Thread باستعمال الفصيلة thread.

ما هى الفصيلة Thread

هى الفصيلة المسؤلة عن تعريف Thread جديد مع توفير الاعضاء (الدوال Methods والخصائص Properties) التى تتحكم فى جميع عمليات الأت Thread مثل بدء تشغيل الـ Thread (الدالة Start()) وايقاف التشغيل Stop() والكثير من العمليات.

برمجة الانترنت Web Programming

ما هي برمجة وتطوير الإنترنت WEB Developing & programming

المقصود بتطوير وبرمجة الإنترنت هو إعداد المواقع و التطبيقات التي تعمل علي الإنترنت .

المواقع Web Sites

الموقع قد يكون بسيط عبارة عن صفحة رئيسية ومجموعة من الصفحات المرتبطة بها وذلك مثل كثير من المواقع ، والغرض منها هو التعرف بالشركة أو الجهة أو الهيئة وعرض أنشطتها ،وبياناتها و كيفية الاتصال بها .

وهناك الكثير من الأدوات التي تستعمل لذلك سوف نعرض لك بعضها .

تطبيق الإنترنت Web Application

تطبيق الإنترنت (web Application) هو تطبيق يعمل من خلال شبكة الإنترنت وذلك لتحقيق وظيفة معينة مثل التطبيق الموجود بموقع شركة التوظيف ، وكذلك تسجيل بيانات الزائرين الذين يريدون وظائف ، ويوفر كذلك إمكانية البحث يستطيع صاحب عمل البحث عن موظفين ، وكذلك يستطيع الزائر البحث عن وظيفة ، وكذلك توجد تطبيق أخرى للإنترنت مثل تطبيقات التجارة الإلكترونية التى تقوم بتوفير قائمة بالشركات المنتجات وتوفير إمكانية اختيار المنتجات وإعداد فاتورة شراء وكثير من التطبيقات الأخرى.

في الفصل السابق عرضنا لك أساسيات لغات البرمجة وشرحنا كم هي متشابه كما عرضنا عناصر لغات البرمجة باختصار.

و في هذا الدرس نبدأ بشرح هذه الأوامر و قواعدها حتى يكون لديك حصيلة من الأوامر و قواعد اللغات التي تساعدك في كتابة ما تريده من أوامر و تحقيق متطلبات البرنامج ونتابع هذه القواعد في الفصول التالية كما يتضح كيف يمكنك الانتقال من لغة الى أخرى بسهولة طالما تعلمت القواعد العامة للغات.

المتغيرات Variables و الثوابت Constants

المتغير يستعمل لحفظ قيمة رقمية أو حرفية أو تاريخ أو غيره لاستعمالها أكثر من مرة داخل البرنامج ولا يخلو برنامج من استعمال المتغيرات فعن طريق المتغيرات تستطيع تكوين التعبيرات Expressions وتستطيع تمثيل بيانات البرامج و والمتغيرات تأخذ أنواع كثيرة.

في VB.NET

لتوضيح بعض هذه الأنواع راجع الجدول التالي:

Integer	متغير يخزن رقم صحيح ولا يأخذ علامة عشرية
Double	متغير يقبل رقم حقيقي يقبل العلامة العشرية
String	متغير يخزن عبارة حرفية
Date	متغير يقبل تاريخ
Boolean	متغير يأخذ احد قمتين True أو False
Byte	متغير يأخذ Byte واحدة
VariantType	يحدد النوع حسب القيمة المدخلة في المتغير

الإعلان عن المتغيرات varible declaration

الإعلان عن المتغير هو إعطاء فكرة مسبقة لمترجم البرنامج بأن هناك متغير varible أو متغيرات بنوع معين ، وكان الإصدار 6.0 لا يفرض عليك الإعلان إلا إذا أردت أن تغير درجة الإعلان كما سيلي أو إذا أردت أن تجبر المترجم أن يبحث عن الإعلان تضبط ذلك من القائمة الرئيسية بينما الاصدار VB.Net يفرض عليك الإعلان عن المتغيرات قبل استعمالها.

و يتم الإعلان عن المتغير بالصورة التالية :

```
Dim a1 As Integer
```

و في هذا السطر يتم استعمال كلمة DIM في الإعلان عن متغير بالاسم A1 من نوع INTEGER مع استعمال AS قبل النوع وكأنك تقول هناك متغير بالاسم a1 من النوع Integer

وبنفس الإسلوب تستطيع الإعلان عن متغيرات من باقي الأنواع التي أشرنا إليها درجات مستوي و أشكال الإعلان .

يأخذ الإعلان عن المتغير درجه معينه تعتمد علي مكان الإعلان وهذه الدرجة تحدد هل يأخذ الإعلان أي الدرجات التالية :

1. هل هو بالمستوى LOCAL أي خاص بالدالة أو الإجراء الذى بداخله المتغير مثل البرنامج الفرعى ()Sub الخاص بزر أمر معين Button.
2. علي مستوي الـ FORM و يسمي PUBLIC وبالتالى يتم التعرف على هذا المتغير داخل جميع دوال وإجراءات هذا الـ Form واذا تم التغيير فيه فى اجراء على المستوى الـ Form يتأثر بهذا التغيير أى إجراء أخر حيث يرى المتغير كأنه معرف لدى جميع إجراءات الـForm بنفس الاسم بنفس النوع.
3. علي مستوي البرنامج GLOBAL ويتم التعرف عل المتغير داخل جميع إجراءات البرنامج.
4. مستوى الفصيلة class وسوف نتناول ذلك فى فصل البرمجة بالأهداف OOP.

وكذلك يمكن للموقع أن يحتوي علي نموذج أو أكثر من التطبيقات وليس تطبيق متكامل فمثلاً شاشة (نموذج) تسجيل البيانات Registration from في أي موقع ويتم بإعداد برامج الإنترنت وكذلك نموذج تطبيقات الزائر feed back وغيرها.

طرق وأدوات إعداد مواقع انترنت web site tools

كما أشرنا في الفقرة السابقة فأن متطلبات إعداد موقع web site تحتاج مجهود وخبرة أقل من متطلبات إعداد تطبيق إنترنت web application وهناك الكثير من الأدوات التي تساعد في إعداد موقع منها ما يلي:

1- استعمال أوامر وعلامات HTML

2- استعمال برامج جاهزة مثل Front Page وبرنامج Dream weaver

3- استعمال برامج حركة مثل Flash

4- استعمال برامج مساعدة مثل Amfy, Cool 3D, XARA 3D, SWICH

5- إضافة برامج JAVA والتي تسمي APPLETS

6- تحميل المواقع علي الإنترنت UPLOAD

هذه بعض أدوات ومتطلبات إعداد مواقع بعضها يجب أن تتعرف علي أساسياته والأخر اختياري يمكنك معرفته أو استبداله وفيما بعد سوف نتناول شرح أساسيات بعض هذه الأدوات.

الفصل الثالث

قواعـد اللغـة المتغيـرات والثوابـت
Constants & Variables

Programming Concepts في هذا الفصل

3

في هذا الدرس نبدأ في شرح قواعد لغات البرمجة التي نستعمل في كتابة الأوامر لتحقيق غرض المستخدم و ذلك بشرح النقاط التالية :-

- المتغيرات Variables و الثوابت Constants
- و أنواع البيانات Data Types
- طرق الإعلان عن المتغيرات
- دوال الإدخال و الإخراج Input/Output

فى لغة C#

لتوضيح بعض هذه الأنواع راجع الجدول التالى:

int	متغير يخزن رقم صحيح ولا يأخذ علامة عشرية
double	متغير يقبل رقم حقيقى يقبل العلامة العشرية
string	متغير يخزن عبارة حرفية
bool	متغير يأخذ احد قمتين True أو False
byte	متغير يأخذ Byte واحدة

والمتغيرات تأخذ أنواع كثيرة نعرضها كما ذكرتها ميكروسوفت فى الدليل الفنى للغة فى الجدول التالى:

C# Type	.NET Framework type
bool	System.Boolean
byte	System.Byte
sbyte	System.SByte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
uint	System.UInt32
long	System.Int64
ulong	System.UInt64
object	System.Object
short	System.Int16
ushort	System.UInt16
string	System.String

الإعلان عن المتغيرات varible declaration

و يتم الإعلان عن المتغير بالصورة التالية :

```
int a;
```

و في هذا السطر يتم استعمال كلمة int في الإعلان عن متغير بالاسم a من نوع

INTEGER وكأنك تقول هناك متغير بالاسم a من النوع Integer.

وبنفس الأسلوب تستطيع الإعلان عن متغيرات من باقى الأنواع التى أشرنا إليها.

فى لغة Java

أنواع البيانات

تنقسم أنواع البيانات فى لغة جافا إلى فئتين رئيستين:

أ- بيانات مبنية ومعرفة فى أصل اللغة Primitive Data Types.

ب- بيانات يقوم المبرمج ببنائها User Defined Data Type.

والآن لنتعرف على النوع الأول:

أ- البيانات المعرفة فى أصل اللغة Primitive Data Types.

وكما قلنا فهى البيانات التى تأتى مبنيه فى اصل اللغة وهى تنقسم إلى أربعة أنواع:

1- البيانات الرقمية الصحيحة Integers مثل (155،10،5،-1،-1).

2- البيانات الرقمية ذات الفاصلة العشرية floating Point Integers مثل (1.33،0.55،....).

3- البيانات الحرفية مثل 'A','B','X' --- char.

4- الاختيار المنطقى (false , true) boolean.

ولكل نوع من الأنواع السابقة نطاق ومساحة تحتلها فى الذاكرة كما فى الجدول التالى:

خواص أنواع البيانات المبنية فى اللغة

النوع	الحجم الذى تحتله فى الذاكرة	النطاق
1- integer		
byte	8 Bits	-127 to 127 $(2^{-7}$ to $2^{7}-1)$
short	16 Bits	-32768 to 32767 $(2^{15}$ to $2^{15}-1)$
int	32 Bits	-2147483648 to 2147483647

long	64 Bits	$(2^{31}$ to $2^{31}-1)$ $=-9.2x10^{15}$ to $=9.2x10^{15}$ $(-2^{63}$ to $2^{63}-1)$
2- floating	Point numbers	
float	32 Bits	$=-3.4x10^{38}$ to $=3.4x10^{38}$
double	64 Bits	$=-1.8x10^{308}$ to $=1.8x10^{508}$
3- character		
char	16 Bit	0 to 65535
4- boolean		
boolean	1 Bit	*false or true*

النطاق والمساحة التى يحتلها اى متغير مذكور فى الجدول السابق لا تختلف باختلاف نظام التشغيل كما فى اللغات الأخرى

الإعلان عن المتغير Variable Declaration

يتم الإعلان عن المتغير بكتابة النوع ثم اسم المتغير مع إمكانية اعطائة قيم مبدأية initializing values .

ولتسمية متغير ما فلك حرية تسمية المتغير على ألا يبدأ الاسم برقم ولا يكون بين حروفه مسافة أو قوس أو الحروف الخاصة مثل (@و#و...) وفيما يلى أمثلة للإعلان عن المتغيرات.

```
int Salary = 500;
boolean male = false;
float age=35.6F;
char c='k';
```

وكما ترى فى الأمثلة السابقة لكى نقوم بالإعلان عن متغير مثل Salary قد سبقه النوع الذى ينتمي إليه int ثم اسم المتغير Salary أما قيمة المتغير فهى 500 وذلك عن طريق المؤثر "=" والمتغير Salary يحتل مساحة فى الذاكرة 32 بت لأنه من نوع int كما فى الجدول السابق وهكذا فلكى تستخدم متغير لابد وأن تحدد نوعه ثم تكتب نوعه ثم اسم المتغير.

فى Oracle PL/SQL

يتم الإعلان بكتابة اسم المتغير ثم النوع كما فى السطور

```
DECLARE
/* Declare variables which will be used in SQL statements */
v_EName         VARCHAR2(30) ;
v_EmpNo         Number := 7788;
BEGIN
/* Select Data from  Emp table. */
SELECT Ename
INTO  v_EName
FROM emp
WHERE EmpNo = v_EmpNo;
END;
```

فى هذه السطور

- فى السطر رقم 1 يبدأ البلوك بالكلمة المحجوزة DECLARE ومعناها إعلان وهى تسبق الإعلان عن المتغيرات المستعملة فى البرنامج.
- فى السطر رقم 3 و4 يتم الإعلان عن متغيرين الأول من النوع VARCHAR2 أى حرف والثانى من النوع Number مع اعطائه قيمة ابتدائية.
- فى السطر رقم 5 تبدأ العمليات داخل البلوك بالكلمة المحجوزة BEGIN وهى تسبق العمليات المطلوب إجرائها فى البلوك.
- فى السطور 7و8 يتم كتابة جملة الـ SELECT مع تعديلها بما يناسب الـ PL وفيها يتم احضار اسم الموظف ENAME من جدول الموظفين EMP ووضع هذا الاسم فى المتغير V_Ename بشرط أن يساوى كود الموظف EmpNo القيمة المكتوبة فى المتغير V_EmpNo.
- فى السطر رقم 11 يتم انهاء البلوك بالكلمة المحجوزة END;.
- وبالتالى ملخص البلوك هو:

كلمة DECLARE تسبق الإعلان عن المتغيرات التى يتم الإعلان عنها بين DECLARE و begin .

بين الكلمة المحجوزة BEGIN والكلمة المحجوزة END; يتم كتابة سطور العمليات.

أنواع البيانات الخاصة بقاعدة البيانات Database Data type

توجد مجموعة من أنواع البيانات التي يمكن استعمالها مع حقول قاعدة البيانات وهي:

Number (size [, precision])

ويستعمل لتخزين قيمة رقمية عدد الأرقام هي size وعدد الأرقام بعد العلامة (القوسين) هي precision.

CHAR (Size)

ويستعمل لتخزين حروف (مثل اسم الموظف) عدد هذه الحروف هي Size.

Varchar2 (size)

يستعمل مثل النوع CHAR ولكن الفرق أنه متغير الحجم أي يتم حجز مساحة مقدارها SIZE كحد أقصى ولكن المساحة الفعلية تقدر بعدد الحروف الحقيقية عكس النوع CHAR الذي يحجز عدد ثابت من الحروف بصرف النظر عن استعمال أم لا.

Date

تستعمل لتخزين قيمة تاريخية (تاريخ).

LONG

يستعمل لتخزين كمية كبيرة من النصوص TEXT تصل إلى 2 Gigabits.

LONG RAW

يستعمل لتخزين كمية كبيره من البيانات لحفظ في صورة BINARY ليست نصوص مثل ملفات الصور وغيره.

-RAW

تستعمل لتخزين كمية من البيانات في صورة BINAEY ولكن أقل من Long raw ويمكن استعمالها لتخزين صور وصوت وفيديو.

أنواع غير مرتبطة بقاعدة البيانات , NON Database Data type

هناك بعض أنواع البيانات غير مرتبطة بأنواع البيانات في قاعدة البيانات بل هي أنواع من البيانات تستعمل فقط مع أوامر البرمجة VHX]t ومنها ما يلي:

-DEC, DECIMAL, REAL, DOUBLE-PRECISIO

هي أنواع فرعية من النوع Number تستعمل داخل الـ PL/SQL.

-INTEGER, INT, SMALLINT, NATURAL, POSITIVE, NUMERIC

وهى بعض الأنواع الأخرى المشتقة أيضاً من النوع Number لتناسب بعض العمليات -

CHARACTER

وهو يقابل النوع CHAR فى أنواع البيانات الخاصة بقاعدة البيانات.

VARCHAR

وهو يقابل النوع VARCHAR2 فى متغيرات قاعدة البيانات.

BOOLEAN

ويستخدم لتخزين قيمة منطقية أما TRUE أو FOLSE.

TABLE

لتخزين مجموعة من القيم المقابل لمصفوفة ARRAY أى مجموعة بيانات من نفس النوع

RECORD

لتخزين مجموعة من القيم المختلفة مثل بيانات سجل فى جدول RECORD.

وسوف نتناول كيفية التعامل مع الانواع المختلفة فى الفصول التالية.

ولتوضيح ذلك تابع السطور التالية كما فى الشكل 15.

الإعلان عن متغيرات من نوع قواعد البيانات

النوع %TYPE

بالإضافة لأنواع البيانات التى عرضناها أضافت ORACLE فكرة جيدة لإلغاء الاختلاف بين نوع البيانات فى قاعدة البيانات ونوع البيانات للمتغير المستعمل فى الأوامر فأضافت ذلك لسهولة الإعلان عن هذه المتغيرات ومنها:

- الأمر TYPE % للإعلان عن متغير من نفس نوع حقل بدون أن تعرف.
- نوع الحقل ويتضح ذلك من السطر التالى:

```
S_SAL  EMP.SALARY % TYPE: = ;
```

فى هذا السطر يتم الإعلان عن متغير بالاسم S_SAL هذا المتغير يأخذ نفس نوع الحقل SALARY المعرف فى الجدول EMP وهذا يوفر عليك مجهود معرفة نوع بيانات الحقول وذلك بإضافة TYPE % للحقل EMP. SALARY ثم إعطائه قيمة ابتدائية.

و بالتالى عند الرغبة فى الإعلان عن المتغير فى السطر PL للتعامل مع قيمة حقل ولا تعرف مواصفات الحقل ليس عليك إلا إضافة %TYPE إلى اسم الحقل و الجدول لتحديد نوع المتغير.

النوع %ROWTYPE

يستخدم هذا النوع (الطريقة) لتعريف متغير من نوع سجل كامل يحتوى على نفس حقول سجل جدول قواعد البيانات بحيث يعبر المتغير عن السجل RECORD ويمكن التعامل مع حقوله، ويتم تعريف هذا المتغير بطريقتين :

الطريقة الأولى: DECLARE

```
MY_Employee EMP % ROWtype;
```

فى هذه السطور ثم تعريف متغير بالاسم my_employee.

هذا المتغير نوعه هو نفس تركيب سجل من الجدول EMP وذلك بإضافة RowType/ إلى اسم الجدول EMP وبالتالى إذا كان الجدول يحتوى على الحقول EMPNO, name salary يصبح المتغير MY_employee له نفس التركيب أى نفس الحقول.

الطريقة الثانية:

وفيها يتم تكوين حقول المتغير بدون الإشارة إلى جدول موجود بالفعل وذلك كما فى السطور التالية:

```
   DECLARE
1: Type A_EMPREC IS RECORD (
2: MY_EMPID        EMP.EMPNO % TYPE,
3: MY_ENAME        EMP.ENAME % TYPE);
4: MY_EMPVAR       A-EMPREC;
BEGIN
```

فى هذه السطور:

- فى السطر رقم 1 تم استعمال الكلمة TYPE للإعلان عن نوع جديد غير الأنواع المعروفة ،هذا النوع أسمه A-EMPREC ثم بعد ذلك IS Record وتعنى أن المتغير عبارة عن سجل record.

- فى السطور رقم 2،3 يتم بناء تركيب النوع الجديد (السجل) وذلك بتحديد مجموعة من حقول الجدول EMP.
- فى السطر رقم 4 إعلان عن متغير بالاسم MY-EMPVAR من النوع الجديد الذى أنشأناه بالاسم A-EMPREC وبالتالى يأخذ المتغير نفس التركيب و النوع أى عدد ونوع حقوله.

التحويلات بين الأنواع

بالاضافة لامكانية الإعلان عن المتغيرات بالمستويات المختلفة واستعمالها توجد إمكانية التحويل من نوع إلى نوع إذا كنت بحاجة الى ذلك وتوجد اكثر من طريقة حسب اللغة.

فى لغة VB.NET

1. كلمات التحويل conversion keyword

توجد مجموعة ن الكلمات التى توفرها .NET Visual Basic للتحويل الى الأنواع المختلفة وهى تستعمل بسهولة كما فى الصورة التالية:

```
Dim K1 As Integer
Dim Q1 As Double
K1= 432
Q1= K1
Q1 = Math.Sqrt(Q1)
K1 = CInt(Q1)
```

- فى هذه السطور تم الإعلان عن متغيرين الأول من النوع Integer والثانى من النوع Double.
- ثم تم استدعاء الدالة Sqrt لحساب الجزر التربيعى لقيمة المتغير Q1.
- ثم تم وضع قيمة المتغير Q1 فى المتغير K1 ولكن بعد وضع المتغير Q1 كمعامل للكلمة CInt(Q1) التى تقوم بتحويله أولا إلى النوع Integer.

2. استعمال الفصيلة Convert Class

توجد فصيلة Class بالاسم Convert تحتوى على عدد من دوال التحويل التى يمكنك استعمالها لتحويل نوع إلى نوع أخر مع ملاحظة أننا سوف نشرح موضوع الفصائل classes فى الفصول 10و11و12.

وتستعمل دوال هذه الفصيلة class كما فى السطور التالية:

```
Dim d1 As Double
D1 = 23.15
Dim i1 As Integer
I1 = Convert.ToInt32(d1)
```

- فى هذه السطور تم الإعلان عن المتغير d1 من النوع Double ثم تم إعطاته القيمة 23.15 ثم تم الإعلان عن المتغير I1 من النوع Integer فى السطر الأخير تم تحويل قيمة المتغير d1 إلى النوع Integer باستدعاء الدالة ToInt32 من الفصيلة Convert ووضع النتيجة فى المتغير i1 فيأخذ القيمة 23

فى لغة C#

يجرى مل يجرى فى VB.NET مع تغيير الطريقة كما يلى:

1. كلمات التحويل conversion keyword

توجد مجموعة من الدوال المجمودة بفصائل الأنواع توفرها C# للتحويل الى الأنواع المختلفة وهى تستعمل بسهولة كما فى الصورة التالية:

```
Int k;
String q='144';
K=INT32.Parse(q)
```

فى هذه السطور تم الإعلان عن متغيرين الاول من النوع Integer والثانى من النوع string ثم تم وضع قيمة المتغير q فى المتغير K ولكن بعد وضع المتغير q كمعامل للدالة Parse التى تقوم بتحويله أولا إلى النوع Integer.

2. استعمال الفصيلة Convert Class

توجد فصيلة Class بالاسم Convert تحتوى على عدد من دوال التحويل التى يمكنك استعمالها لتحويل نوع إلى نوع أخر مع ملاحظة إننا سوف نشرح موضوع الفصائل classes فى الفصول 10و11و12.

وتستعمل دوال هذه الفصيلة class كما فى السطور التالية:

```
Double d1;
d1 = 23.15;
int i1;
I1 = Convert.ToInt32(d1)
```

فى هذه السطور تم الإعلان عن المتغير d1 من النوع Double ثم تم إعطائه القيمة 23.15 ثم تم الإعلان عن المتغير I1 من النوع Integer فى السطر الأخير تم تحويل قيمة المتغير d1 الى النوع Integer باستدعاء الدالة ToInt32 من الفصيلة Convert ووضع النتيجة فى المتغير i1 فيأخذ القيمة 23

فى لغة Java

يتم استعمال طريقتين من التحويل:

1. استعمال طريقة Casting

وهو اما تحويل تلقائى كما فى السطور التالية:

```
byte a=20;
short b;
int c;
long d;
float e;
double f;
b=a;
c=b;
d=c;
E=d;
F=e;
```

أو باستعمال أوامر التحويل.

2. استعمال دوال التحويل فى الفصائل

حيث تحتوى الفصائل classes على دوال للتحويل الى أنواعها مثل الفصيلة Integer التى تحتوى على الدالة ()ParseInt و هكذا

المؤثرات Operatros

المؤثر هى رمز خاص يستخدم فى العمليات الحسابية والمنطقية التى تجرى على المتغيرات.

فى لغة Java ولغات C-Like

أنواع المؤثرات Operators Types

توجد عدة أنواع من المؤثرات فى لغة جافا Java وهم

1. المؤثرات الحسابية Arithemetic Operator
2. المؤثرات المنطقية Logical Operator
3. المؤثرات العلاقية Relational Operator
4. المؤثرات على مستوى البت Bit-Wise Operator

أ- المؤثرات الحسابية

المؤثرات الحسابية وهى (*,+,/,%,++,--)

أمثلة على المؤثرات الحسابية

المؤثر (+)

```
int a = 10;
int b =20;
int Sum;
Sum = a + b; // Sum =30
```

المؤثر (-)

```
int salary = 500;
int deduction = 85;
int netsalary;
netasalary = salary – deduction;
```

المؤثر (*)

```
int Salary = 500;
int Bonus = 2* Salary;
```

المؤثر (/)

```
int Salary = 500;
int halfsal =500/2;
```

المؤثر (%) وهو باقى القسمة Modulo

```
int x = 10% 3 ;                    // x = 1
```

المؤثر (++) مؤثر الزيادة

وهو يعمل على زيادة المتغير بقيمة 1 فمثلا كى نزيد المتغير Count بقيمة 1 كنا نقوم بعمل الآتى:

```
int Count;
Count = Count +1;
```

ونستطيع فعل ذلك بطريقة مختصرة عن طريق (++)

```
Count ++;      // doing the Same Count = Count t+1
```

وتنطبق هذه الطريقة أيضا على المؤثر (- -) الذى يعمل على نقصان المتغير بقيمة 1

أمثلة مختلفة على اختصارات المؤثرات

يقوم هذا المثال بالإعلان عن متغير من نوع int ثم يقوم بإضافة 5

```
int x=10;
x=x+5;
```

يمكننا فعل ذلك بطريقة مختصرة كالآتي

```
x+=5;
```

وينطبق ذلك على بقية المتغيرات

```
x=x*5;
x*=5;
x=x/3;
x/=3;
```

أسبقية التعامل مع المؤثرات Operator Precedence

والآن بعد أن تعرفنا على المؤثرات الحسابية نعالج معى نتعرف على أسبقية معالجة المؤثرات من خلال هذا المثال.

```
int answer;
answer = 3+5*3;
```

إذا كانت النتيجة 24 فأنت مخطئ.

لأن عزيزي القارئ فى هذا المثال ما هو ترتيب تنفيذ العمليات السابقة هل الجمع أم الضرب؟ الحل بسيط ويخضع للجدول التالى الذى يرتب تنفيذ معالجة العمليات.

دالة الإدخال INPUTBOX

تستخدم هذه الدالة لاستقبال قيمة من المستخدم و ذلك في مربع الرسالة حيث تأخذ الصورة العامة التالية :

```
M = INPUTBOX (MSG , DEFAULT , TIT )
InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])
```

في هذه الصورة تأخذ الدالة ثلاثة معاملات .

- الأول MSG و يعرض الرسالة المطلوبة.
- الثاني DEFAULT و يعرض قيمة افتراضية في مربع الاستقبال، إذا قبلها المستخدم تختار OK وإلا يغيرها أولاً.
- الثالث TIT و يعرض عنوان مربع الرسالة فى شريط العنوان.

فإذا وافق المستخدم على هذه القيمة ينقر زر OK و إلا يكتب ما شاء ثم OK.

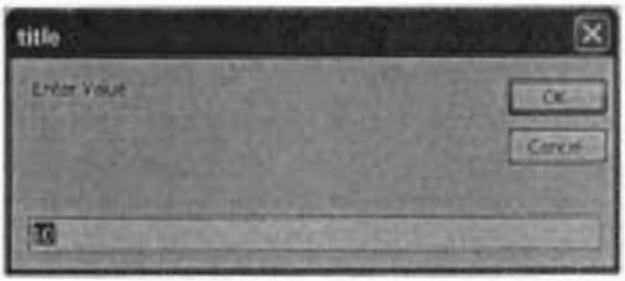
جرب استعمال هذه الدالة كما يلي :

1- ابدأ برنامج جديد وقع زر أوامر و غير عنوانه إلى CLICK.

2- اضغط زر الأمر مرتين و أكتب فيه ما يلي:

```
Dim M
M=INPUTBOX ( "Enter Value","title" ,10)
MSGBOX " YOU ENTERD                " &M
```

- في هذا السطر ثم أستدعاء الدالة INPUTBOX بمعاملاتها مع وضع القيمة التي أدخلها المستخدم في المتغير M ويظهر مربع الإدخال كما بالشكل التالى ثم عرض هذه القيمة بالدالة MSGBOX.



هذا بالاضافة للطريقة الطبيعية والمشهورة فى جميع اللغات الحديثة مثل .....VB.Net,C#,Java

وهى استعمال مربع النص Text Box فى ادخال المدخلات وكذلك إظهار النتائج.

فى لغة Java

هى بالطبع الجمل الأولية التى تستخدم للتدريب ولكن واقعيا يتم الإدخال والإخراج من برامج النوافذ Windows Applications أو صفحات مواقع الإنترنت.
ولكن هذه الدوال تستعمل فى بيئة نظام التشغيل DOS .

أوامر المخرجات

تستعمل الدالة println() فى طباعة المخرجات على الشاشة وبالطبع هى دالة Method بداخل فصيلة class بداخل حزمة Package وهذا ما سوف نتناوله فيما بعد بالتفصيل وبالطبع هى تطبع العبارات الحرفية String وتضيف اليها أى قيمة رقمية وتطبع الدالة print().

أوامر الادخال

تستخدم الدالة readLine() وتستخدم لاستقبال قيمة حرفية String ، ولاستعمالها لاستقبال قيم رقمية يتم استقبالها كقيم حرفية ثم تحويلها

بلوك الأوامر Block Statements

البلوك هو طريقة جمع مجموعة من سطور الأوامر معا للتعامل معها كوحدة وتسمى block.

فى لغات C-Like

يتم وضع مجموعه من الجمل او الأوامر محصورة بين الأقواس { } كما بالشكل:

```
{
    :
    :
    :
Statement 1
Statement 2
    :
    :
}
```

المؤثر	Description
( )	
+ +	Increment
- -	Decrement
*	Multi Placation
/	Division
%	Modulus
+	Addition
-	Subtraction
=	Assignment

ونتيجة تنفيذ المثال السابق هى 18 لأنه كما فى الجدول السابق فإن الضرب لـه أسبقية فى التنفيذ عن الجمع أي أن عمليات الضرب تتم معالجتها أولا.

وإذا أردنا المترجم أن يقوم بعملية الجمع أولا فى حالة المثال السابق نقوم بعمل الآتي

```
answer = (3+5) *3
```

وهنا لأن الأقواس لها أسبقية عن الضرب.

دوال الإدخال و الإخراج I/O Functions

كما هو مشهور في جميع لغات البرمجة توجد جمل (دوال) الإدخال و الإخراج I/O Functions التى تستعمل في استقبال قيم من المستخدم و كذلك عرض الرسائل و القيم له .

و بالرغم من اختلاف ذلك شيئا ما مع برمجة النوافذ إلا أنه توجد دوال كما يلي:

فى لغة VB.NET

دوال الإخراج (عرض الرسائل)

الدالة MSGBOX

و تأخذ الصورتين

```
1-MSGBOX MSG
2-MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

في الصورة الأولي :تم استعمال الدالة MSGBOX مع الرسالة المطلوب عرضها و هي محتوي المتغير MSG و هذه الصورة التي استعملناها في البرنامج السابق وهو صورة بسيطة يحدد فيها الرسالة المطلوب عرضها فقط مع الدالة MsgBox().

في الصورة الثانية :

يتم استعمال الدالة مع إرسال ثلاثة معاملات فا و هي :

- الأول : MSG و هو الرسالة المطلوب عرضها كما سبق .
- الثاني : رقم أو ثابت يعبر عن شكل مربع الرسالة و عدد الأزرار به هل يعرض مربع الرسالة زر أمر Button واحد بالعنوان Ok أم يعرض زرى أمر Ok,Cancle.
- الثالث : TITEL و يعبر عن العنوان الذي تريد أعطاءه لمربع الرسالة.
- المتغير K : عبارة عن قيمة أو ثابت يعبر عن اختيار المستخدم هل اختار زر الامر Ok أم اختار زر الامر Yes وهكذا.
- ويظهر ذلك من النص التالي:

```
1:Dim v1, tit, r
2:v1 = TextBox1.Text
3: TIT = TextBox2.Text
4:r = MsgBox("this Msg WITH NO ", Val(v1), tit)
5:MsgBox(" YOU SELECT NO " & R)
```

ف هذه السطور :

- في السطر رقم 1 و رقم 2 تم وضع قيم أدوات الكتابة في المتغيران TIT , V1وقد تم الإعلان عنها ف السطر الأول.
- في السطر رقم 3 يتم استدعاء الدالة MSGBOX مع إرسال معاملات هي الأول الرسالة "this Msg WITH NO " و الثاني القيمة V1 التي أدخلها المستخدم ف مربع النص TextBox1 والثالث المتغير TIT و هو عنوان مربع الرسالة الذى ادخله المستخدم ف مربع النص الثانى TextBox2.
- في السطر رقم 4 يتم عرض رسالة توضح الزر الذي اختاره المستخدم من مربع الرسالة و ذلك بعرض قيمة المتغير R الذي أخذ هذه القيمة في السطر رقم 3 من نتيجة استدعاء الدالة.

فى هذا الشكل يتم حصر جمل Statement 1 و Statement 2 بين الأقواس ليتعامل كبلوك.

ولكن السؤال ما الفائدة وراء إنشاء بلوك جمل ؟

الفائدة هى إمكانية الإعلان عن متغيرات داخل البلوك ليس لها علاقة بما قبل أو بعد البلوك مثل تعريف متغير داخل بلوك داخل دالة فهو خاص بها.

وكذلك إمكانية تنفيذ عملية مثل التكرار looping وعملية الشرط Condation على هذه المجموعة من الأوامر معا.

فى لغة VB.NET

يتم استعمال كلمات لإنهاء البلوكات ولكن حسب نوع البلوك.

فمثلا لإنهاء بلوك التكرار for يتم استعمال كلمة Next.

ولإنهاء بلوك الشرط If يتم استعمال الكلمة End If وسوف يتضح ذلك فى الفصول التالية.

الفصل الرابع

جمل التكرار والتحكم فى مسار البـرنامج

Loops &Control Statements

Programming Concepts 4 فى هذا الفصل

هي من أشهر الجمل في جميع لغات البرمجة حيث تسمح بتكرار عملية أو مجموعة سطور أكثر من مرة وكذلك جمل التي تحدد مسار البرنامج و تحقق رغبات المستخدم و هي جمل التحكم في مسار البرنامج حيث نشرح النقاط التالية

- جمل التكرار..... , FOR
- جمل التكرارFor Each
- الأشكال المختلفة للجملة FOR
- الجملة WHILE ...WEND
- جملة while
- جملة do ..... while

التكرار Looping

من العمليات المسموح بها في جميع اللغات عملية التكرار وهي تكرار تنفيذ عملية معينه أو مجموع سطور أكثر من مرة وهي مفيدة بدرجة انها لا يمكن الاستغناء عنها في كثير من البرامج.

وينقسم التكرار Looping إلى نوعين:

- تكرار معلوم العدد: وفيه معروف عدد مرات التكرار وتستعمل له الجملة for بأشكالها المختلفة.
- تكرار غير معلوم: واحياناً يسمى تكرار مشروط لان التكرار يتوقف على شرط معين وفيما يلي شرح لكلاً النوعين والجمل المختلفة :-

في لغة VB.NET

الجملة FOR

وتأخذ الصورة العامة التالية

```
For counter = start To end [Step step]
[statements]
[Exit For]
[statements]
Next [counter]
```

و هي تستعمل في تكرار تنفيذ عملية معينة عدد من المرات .

فمثلا لطباعة كلمة VISUAL BASIC 20 مرة سوف نكتب 20 سطر لتحقيق ذلك بدون استعمال جمل التكرار في حين تستعمل جملة تكرار مع سطر واحد لتنفيذ ذلك.

و يتضح ذلك من السطر التالي :

```
Dim i
For i = 0 To 10

   MsgBox(" Visual Basic.Net")
Next (i)
```

في هذه السطور تم استعمال جملة التكرار FOR .

لتنفيذ السطر رقم 2 عدد 10 مرات و هذا ما يوفر علينا كتابة سطر 2 عدد 10مرات ولتوضيح فيم بإنشاء برنامج جديد ووقع زر أمر Button1 وكتابة السطور السابقة به ثم نفذ البرنامج وأنقر الزر تلاحظ ظهور مربع الرسالة Msgbox بالعبارة 10 مرات.

أنواع الجمل

توجد أكثر من جملة للتكرار هي :

```
1. For ..Next
2. Do...      Loop
3. While.. End While
```

جملة FOR

نستخدم جملة FOR لتكرار تنفيذ عملية عدد من المرات معلومة العدد و تأخذ أكثر من صورة نعرضها فيما يلي:

الصورة الأولى :

```
1: Dim i
2:       For i = 0 To 20
3:         MsgBox("I=" & i)
4:    Next i
```

هذه هي أبسط صور التكرار حيث تبدأ في السطر رقم 2 بكلمة FOR ثم متغير I (أي متغير) و هو العدد يبدأ بالقيمة 0 ثم TO و القيمة 20 و بالتالي يكون معني السطر رقم 2 هو كرر ما يلي ابتداء من القيمة 0 إلي 20 أي 20 مرة .

- السطر رقم 3 يقوم بطباعة قيمة المتغير i كل مرة ولكن بالصورة I=0 وهكذا.
- السطر رقم 4 كلمة NEXT التي تحدد الجمل المطلوب تكرارها.

و بالتالي لتكرار جمل معينة نكتبها بين كلمتي FOR و كلمة NEXT .

الصورة الثانية :

```
1: Dim i
2:       For i = 0 To 20 step 2
3:         MsgBox("I=" & i)
4:    Next i
```

```
K=O
DO   WHILE     K <10
Msgbox k
K=K+1
LOOP
```

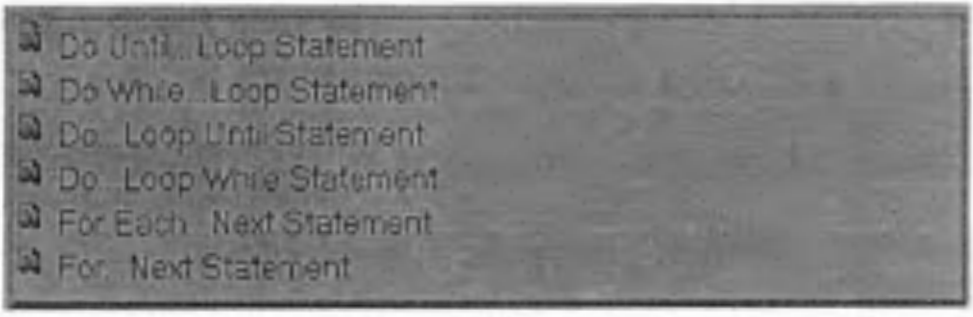
في هذه السطور يتم طباعة قيم K من 0 إلى 10 طالما أن K<10 كما في جملة الشرط.

و لكن يفضل أستعمال DO WHILE مع حالات عدم معرفة عدد المرات مثل شرط EOF الذي يعبر عن نهاية الملف الذي لا يعرف فيه عدد السجلات.

ويمكن اضافة الصيغة من قائمة Snippet التى أشرنا اليها مما يوفر عليك حفظ الصيغة

الاستفادة من الـ Snippet

توفر .NET قائمة Snippet تعرض جميع الصور المتاحة للتكرار Loops كما في الشكل (4-1)



الشكل(1-4)

وعند اختيار هذه الصور تحصل على تركيبها كما في السطور التالية

```
Do Until False

Loop
```

```
Do While True

Loop
```

```
Do

Loop Until True
```

```
Do

Loop While True
```

```
For Each Item As String In CollectionObject
Next
```

```
For index As Integer = 1 To 10
Next
```

```
While True
End While
```

فى لغات C-Like

تلتقى لغات C-Like مثل لغة Java و C# و C++ فى الأوامر وكيفية تنفيذها وأن كانت الأوامر التالية تنفذ باستعمال لغة Java مباشرة كما فى يل

التكرار باستعمال for Loop

تستعمل هذه الجملة لتكرار تنفيذ عملية اكثر من مرة وهى ابسط وأشهر أنواع جمل التكرار وتأخذ الصورة التالية:

وتأخذ الصورة العامة التالية :

```
for ([initializers]; [expression]; [iterators])
  statement
```

فى هذه الصورة تأخذ جملة for الأجزاء التالية:-

for: الأمر نفسه المستعمل فى التكرار

initializers: القيمة الابتدائية التى يبدأ بها التكرار.

expression: شرط التكرار الذى يستمر التكرار طالما هذا الشرط صحيح.

ويتم استعمال الأقواس { } لتكوين البلوك الذى ينفذ داخل التكرار وإذا كانت جملة واحدة لا تحتاج إلى أقواس.

iterators : مقدار الزيادة

statement: هى الجملة المطلوب تنفيذها داخل التكرار.

مع ملاحظه فصل إجزاء جملة for بالفاصلة المنقوطة ;

الجديد في هذه الصورة :

هو كتابة STEP 2 في السطر رقم 1 و هو تغيير خطوة العد فبدلا من أن يكون العد هو 0 و 1 و 2 و 3 يصبح 0 و 2 و 4 و 6 و 8 و هكذا و يمكنك تغيير قيمة الخطوة حسب العملية المطلوبة

الصورة الثالثة:

```
1: Dim i
2:       For i = 10 To 0 step  -1
3:         MsgBox("I=" & i)
4:   Next
```

•في السطر رقم 1 تم ابتداء العداد من القيمة 10 إلي القيمة 0 مع تحديد الخطوة بمقدار 1- (يمكن تنفيذ الخطوة) و بالتالي يصير العد بالطريقة 10 و 9 و 8 و7....

أي عد تنازلي و باقي السطور كما سبق.

الصورة الرابعة :

وتسمى هذه الصورة Nested Loops أو التكرار المتداخل حيث يتم استعمال تكرار داخل تكرار.

```
Dim i, k, m, m2, m3
      For i = 0 To 5
         For k = 0 To 4
            m = "I=" & i
            m2 = " k=" & k
            m3 = m + m2
            MsgBox(m3)
         Next (k)
      Next (i)
```

- في السطر رقم 1 بداية تكرار بجملة FOR بمتغير I يبدأمن 0 إلي 5
- في السطر رقم 2 بداية تكرار آخر بجملة FOR بمتغير K يبدأ من 0 إلي 4
- في السطر رقم 3 يتم طباعة قيم I,K وهي أول مرة 0,0
- في السطر رقم 4 جملة NEXT K التي تؤدي إلي ذهاب التكرار إلي السطر رقم 2 الذي يزيد من قيمة K لتصبح 1 ثم طباعتها في سطر 3 ثم سطر 4 يعيد التكرار مرة أخري

لزيادة قيمة K و هكذا حتي تصل قيمة K إلي 4 فينتقل التنفيذ إلي سطر 5 الذي يؤدي إلي الذهاب إلي سطر رقم 1 الذي يزيد من قيمة المتغير I لتصبح 1 ثم يعاد تنفيذ التكرار الرافع و هكذا.

كل قيمة من المتغير I يقابلها 4 قيم من المتغير K و بالتالي يكون شكل القيم كما يلي:

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4

و هذه الصورة تسمي NESTED LOOPS أي التكرار المتداخل .

و يكثر استعمالها مع المصفوفات

يمكنك تجربة الصور المختلفة و ذلك بإنشاء برنامج جديد و توقيع زر أمر عليه و كتابة سطور الصورة داخل دالة زر الأمر ثم تنفيذ البرنامج و تجربة و مشاهدة نتيجة التنفيذ حتي يتكون لديك تصور عن معني الجمل.

جملة DO.....WHILE

الجملة DO WHILE تستخدم لتكرار عملية عدد من المرات غير معلومة العدد و لكن يعتمد هذا التكرر علي شرط WHILE حيث تأخذ الصورة العامة التالية :

```
Do [{While | Until} condition]
[statements]
[Exit Do]
[statements]
Loop
Or, you can use this syntax:
Do
[statements]
[Exit Do]
[statements]
Loop [{While | Until} condition]
```

في هذه الصورة :

يتم تكرار ما بين DO و الفرق الآخر LOOP طالما أن الشرط مكتوب أمام كلمة WHILE صحيح فإذا أصبح غير صحيح يتوقف التكرار .

ويمكن تنفيذ هذه الصورة علي التكرار العددي كما بالشكل التالي :

- ويتم استعمال التكرار for بكثرة مع المصفوفات لتجنب التعامل مع كل عنصر على حدة.

لا تنتهي جملة for أو جملة if بالفاصلة المنقوطة ; وإلا يسير البرنامج في خطوات خطأ، فمثلاً لا تكتب السطور التالية:

```
for (i =0; i<20; i++); //Logic Error
System. Out. Print In ("Java –");
```

في هذه الحالة لا يتم تنفيذ الدالة Print In() عشرة مرات بل يتم تنفيذ التكرار منفرد بدون التأثير على دالة Print In() ثم تنفذ الدالة مرة واحدة.

و هي تستعمل في تكرار تنفيذ عملية معينة عدد من المرات .

فمثلا لطباعة كلمة Java عدد 20 مرة سوف نكتب 20 سطر لتحقيق ذلك بدون استعمال جمل التكرار في حين تستعمل جملة تكرار مع سطر واحد لتنفيذ ذلك.

و يتضح ذلك من السطر التالي :

```
        for (int i = 1; i <= 10; i++)
System. Out. Print In ("Java –");
```

ولتوضيح ذلك قم بانشاء برنامج جديد واكتب سطوره كما في الشكل.

```
1  //import java.io.*;
2  class For1 {
3
4      public static void main(String[] args) {
5          // Create application frame.
6  int i;
7                  for ( i = 1; i <= 10; i++)
8          System.out.println("Java ");
9
10     }
11 }
12
```

- زيادة مقدار الزيادة .

وفيه يتم تغيير مقدار الزيادة ويأخذ الصورة التالية:

```
for(I=0;I<20;I+=2)
```

وفي هذه الصورة يتم زيادة قيمة المتغير I بمقدار 2 كل مرة.

ولتوضيح ذلك قم بتعديل البرنا مج السابق ليصبح كما فى الشكل.

```
1  class For1 {
2
3     public static void main(String[] args) {
4        // Create application frame.
5  int i;
6              for ( i = 0; i <= 20; i+=2)
7           System.out.println("   Java  i= "+i);
8
9     }
10 }
```

ف هذه السطور:

قم بترجمة وتنفيذ البرنامج لحصل على نتيجة التنفيذ كما فى الشكل.

- الطباعة من الأكبر إلى الأصغر ويأخذ الصورة التالية:

```
for(I=20;I>0;I--)
```

وفى هذه الصورة يبدأ التكرار من القيمة 20 ثم يتناقص بمقدار 1 طالما أكبر من 0

التكرار المتداخل Nested Loops

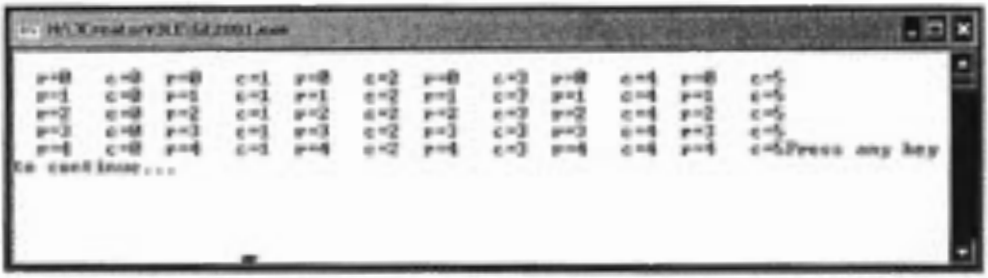
وفيه يتم استعمال تكرار for داخل تكرار for اخر كما يلى:

```
for(r=0;r<5;r++)
for(c=0;c<6;c++)
System.out.println("r="+r+"c="+c);
```

ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل

```
1  public class For4 {
2
3      public static void main(String[] args) {
4          // Create application frame
5  int r,c;
6      for(r=0;r<6;r++)
7      {
8        System.out.println("");
9
10
11 for(c=0;c<6;c++)
12 System.out.print(" r="+r+"  c="+c);
13 }
14    }
15 }
16
```

قم بترجمة وتنفيذ البرنامج تحصل على نتيجة التنفيذ كما فى الشكل.



مثال جدول الضرب

من الأمثلة الواضحة جدا على استعمال التكرار المتداخل Nested Loops هو طباعة جدول الضرب المشهور ، ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل.

```
1  public class For4 {
2
3      public static void main(String[] args) {
4          // Create application frame
5  int r,c;
6      for(r=1;r<13;r++)
7      {
8        System.out.println("");
9
10
11 for(c=r;c<13;c++)
12 System.out.print(r+"*"+c+"="+(r*c)+" ");
13
14 }
15    }
16 }
17
```

التكرار باستعمال While – Loop

الجملة while

تستعمل هذه الجملة لتكرار تنفيذ مجموعة من السطور عدد من المرات غير محدد مسبقا ولكن يتوقف على شرط ولذلك يفضل استعمالها فى حالة عدم معرفة عدد مرات التكرار وتاخذ الجملة التركيب التالى:

```
while
(expression)
statement
```

وفى هذه الصورة يتم اختبار الشرط expression إذا كان صحيح يتم تنفيذ الجمل Statements ويستمر التنفيذ حتى يصبح الشرط غير صحيح.

ولتوضيح ذلك تابع المثال التالى:

```
int n = 1;

      while (n < 6)
      {
         System.out.Println (n);
         n++;
}
```

فى هذه السطور

- فى السطر رقم 1 تم إنشاء فصيلة جديدة بالاسم usingWhile ثم الدالة الرئيسية main()
- فى السطر رقم 3 تم الإعلان عن مصفوفة قيم صحيحة int.
- فى السطر رقم 5 يتم استعمال التكرار while باختبار قيمة المتغير I طالما أقل من عدد عناصر المصفوفة نفذ ما يلى.
- فى السطر رقم 7 يتم طباعة عصر المصفوفة رقم I ثم يتم زيادة قيمة المتغير I حتى تنتهى عناصر المصفوفة.

ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل.

```
public class For4 {

   public static void main(String[] args) {
      // Create application frame
int n = 1;

      while (n <=10)
      {
        System.out.println ("     n=" +n);
        n++;
}
}
}
```

وبمراجعة أجزاء جملة الـ While تلاحظ أنها تحتوى على نفس أجزاء التكرار for

استخدام do --- While

يستخدم هذا التكرار مثل While ولكن يختلف عنه في انه يبدأ أولاً بتنفيذ الجمل ثم يختبر الشرط في While في نهاية التركيب فإذا كان صحيح يعاد التكرار مرة أخرى وإلا يتوقف وبالتالي يتم تنفيذ الجمل مرة واحدة على الأقل حتى لو كان الشرط خطأ، وتأخذ الصورة العامة التالية.

```
Do {
Statements
} While (Condation);
```

في هذه الصورة يتم تنفيذ الجمل statements أولاً ثم اختبار الشرط الموجود مع While فإذا كان صحيح يعود التكرار إلى do ويتم تنفيذ الجمل مرة أخرى وهكذا.

مثال:

في السطور التالية نعرض مثال بسيط لاستعمال do---While لطباعة مجموعة عبارات كما في الشكل.

```
public class For4 {

   public static void main(String[] args) {
      // Create application frame.


   int x = 1;
   do {
   System.out.println ("this is no "+x);
   x++;
   } while (x<=20);


}
}
```

فى هذه السطور

- يتم تعريف فصيلة جديدة بالاسم for4.
- ثم يبدأ التكرار فى السطر رقم 4 بالأمر do فيتم طباعة الرسالة this is no بالإضافة لقيمة المتغير x وهى أول مرة 1.
- ثم يتم زيادة قيمة x وفى السطر 7 يتم استعمال باقى تركيب التكرار وهو الأمر while الذى يختبر قيمة المتغير x فإذا كانت أقل من 20 يعود التكرار مرة أخرى إلى أول التكرار بالسطر 4

الخروج من جمل التكرار Breaking Out Of loops

يمكن استعمال الأمر Break للخروج من التكرار قبل إنهاء الشرط الخاص به ويظهر كما فى السطور التالية:-

```
int i =0;
While (i <50)
      {
if (i >20) break;
System.out.Println (i);
  }
```

فى هذا المثال

- يتم التكرار من 0 حتى 50 وليكن شرط لها يقطع هذا التكرار عند أول قيمة أكبر من 20 وهى 21.

فى Oracle PL/SQL

ويوجد أكثر من أمر تكرار تنفيذ العمليات وهي :

التكرار LOOP---if condition –Exit

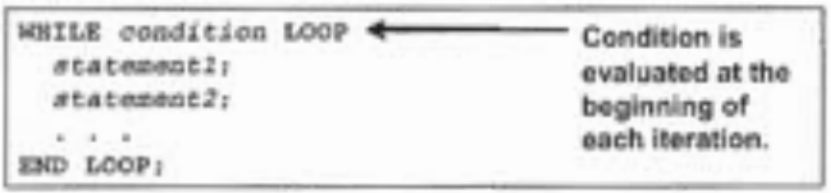
التكرار LOOP----Exit when

التكرار while----loop

التكرار For ----loop

التكرار WHILE...LOOP

فى هذه الجملة يتم إضافة شرط التكرار مع While لتكوين تكرار while بدلا من تحديد شرط باستعمال جملة IF أو EXIT WHEN وبالتالى يختلف تركيب التكرار وذلك كما بالشكل:



فى هذه الصيغة يبدأ التكرار بشرط التكرار condition مع أمر التكرار while ثم تبدأ جمل التكرار ككل جمل التكرار بالامر LOOP. ثم الجمل المطلوب تكرارها statement1,... ثم ينتهى التكرار كما هو معتاد بالامر END LOOP وبهذا الاسلوب لا تحتاج شرط للخروج لأن الشرط كتب فى أول التكرار مع الامر while ولتوضيح ذلك تابع السطور التالية:

```
DECLARE
MY_VAR NUMBER:=0;
BEGIN
 While MY_VAR <11
LOOP
DBMS_OUTPUT.PUT_LINE('MY_VAR='|| MY_VAR);
 MY_VAR := MY_VAR+1;
 END LOOP;
END;
```

فى هذا السطور:

- تم إضافة شرط التكرار وهو MY_VAR<11 مع بداية التكرار مع While ثم كتابة loop التي تبدأ طالما أن الشرط صحيح ويتم تنفيذ السطر رقم 2 ثم إعادة التكرار وهكذا حتى يصبح الشرط غير صحيح أي تزيد قيمة المتغير MY_VAR عن 11.
- ويختلف هذا التكرار عن التكرار السابق في أن التكرار WHILE لا يبدأ بالتكرار ولا مرة إلا إذا كان الشرط صحيح ، بينما التكرار السابق يقوم بتنفيذ الأوامر مرة واحدة على الأقل ثم يختبر الشرط ليعيد التكرار أو يتوقف.

جملة التكرار FOR

هو من أشهر الطرق في جميع لغات البرمجة حيث يستعمل التكرار تنفيذ عملية عدد من المرات معروفة ، ويأخذ الشكل التالى:

```
FOR counter IN [REVERSE]
    lower_bound..upper_bound LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

فى هذه الصورة:

يبدأ التكرار بالأمر FOR ثم المتغير المستعمل فى التكرار هو أي متغير ثم IN لتحدد أن قيمة التكرار لابد أن تقع فيما يلي:

- وهو 1—1 أي من القيمة 1 إلى القيمة 1 بمقدار زيادة 1 كل مرة ، ثم الأمر LOOP لتكرار تنفيذ الجمل التالية.
- ثم تكتب الجمل المطلوب تكرارها وفى نهاية الجمل يكتب END LOOP لإنهاء جمل التكرار .
- ويمكن تغير قيم التكرار بالعكس أي بدلاً من تكرارها من القيمة 1 إلى 1 ويتم تكرارها من 1 إلى 1وذلك بإضافة كلمة REVERSE كما بالشكل:

FOR MY_VAR IN REVERSE 1—1 LOOP

```
|
END LOOP;
```

ولتوضيح استعمال FOR تابع المثال التالى:

ويلاحظ ذالك فى الشكل:

```
REM NumericLoop.sql
REM Chapter 1, Oracle9i PL/SQL Programming by Scott Urman
REM This block illustrates a numeric FOR loop.


BEGIN
  FOR v_LoopCounter IN 1..50 LOOP
```

التكرار LOOP---if condition –Exit

وفيه يتم تكرار الأوامر التي بينهما بدون توقف ولإيقاف هذا التكرار لابد من إضافة شرط داخل التكرار كما يلي:

```
LOOP                                    -- delimiter
  statement1;                           -- statements
  . . .
 IF  condition THEN
  EXIT;                                 -- EXIT statement
END IF ;
END LOOP;                               -- delimiter
```

في هذه الصيغة

تبدأ هذه الصيغة بالأمر LOOP الذي يعني كرر الجمل التالية وتنتهي الصيغة بالأمر END LOOP الذي ينهي الجمل المطلوب تكرارها وبينهما الجمل.

كما يجب أن يحتوى هذا النوع من جمل التكرار على أمر شروط وخروج من التكرار وفي هذه الصيغة نستعمل الأمر IF ثم الشرط ثم الأمر EXIT إذا تحقق الشرط ولتوضيح ذلك تابع المثال التالي:

```
DECLARE
MY_VAR Number (10) :=1 ;
BEGIN
LOOP
DBMS_OUTPUT.PUT_LINE('MY_VAR ='||MY_VAR);
MY_VAR := MY_VAR +1;
IF MY_VAR = 9     THEN
EXIT;
END IF;
END LOOP;
END;
```

في هذه السطور تم استعمال التكرار LOOP—EXIT كما يلي:

- في السطر رقم 2 إعلان عن متغير MY-VAR من نوع رقمي.
- في السطر رقم 4 بداية التكرار باستعمال الأمر LOOP.
- في السطر رقم 5 يتم إضافة 1 لقيمة المتغير MY-VAR إذا كانت 9.
- ينفذ السطر 7 وهو الأمر EXIT ويتم الخروج من التكرار.

- وإلا يستمر تنفيذ الأوامر وعدم الخروج إلى السطر رقم 9 END LOOP الذى يعيد التنفيذ مرة أخرى إلى بداية التكرار وهو السطر 4 وهكذا حتى تصل قيمة المتغير إلى 9.

جملة التكرار LOOP----EXIT WHEN

يتم استعمال الامر EXIT WHEN بدلا من استعمال IF وهى أسهل وتأخذ الصورة العامة التالية:

```
LOOP                                    -- delimiter
  statement1;                           -- statements
  . . .
  EXIT [WHEN condition];                -- EXIT statement
END LOOP;                               -- delimiter
```

فى هذه الصيغة تم استبدال جملة الشرط IF وأمر الخروج EXIT بالأمر EXIT WHEN ولتوضيح ذلك تابع السطور التالية:

```
REM SimpleLoop.sql
REM This block contains a simple loop.


DECLARE
  v_LoopCounter BINARY_INTEGER := 1;
BEGIN
  LOOP
    INSERT INTO temp_table (char_col)
      VALUES (v_LoopCounter);
    v_LoopCounter := v_LoopCounter + 1;
    EXIT WHEN v_LoopCounter > 50;
  END LOOP;
END;
/
```

فى هذا السطور:

فى هذا السطر تم وضع الشرط مع الأمر WHEN بدلاً من الأمر IF وهذا الأسلوب أوقع وأوضح من استعمال IF ويؤدى إلى نفس النتيجة حيث يؤدى إلى إضافة الأرقام من 1 إلى 50 فى الجدول Temp_table

```
        INSERT INTO temp_table (char_col)
          VALUES (v_LoopCounter);
      END LOOP;
    END;
```

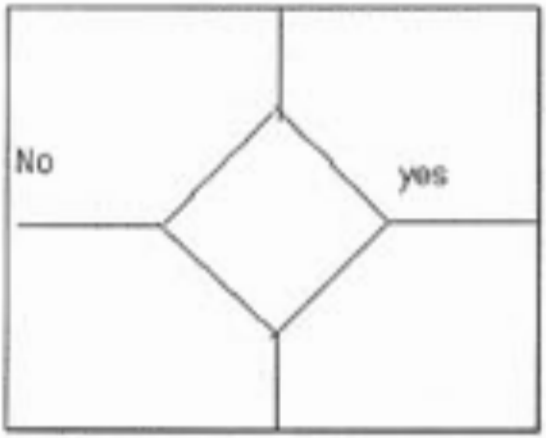
جمل التحكم في مسار البرنامج

في هذه الفقرة نتناول الجمل التالية:

- جملة ..... If
- جملة ....If.....Else
- جملة ....If ..... else if ....
- جملة select case....end select
- جملة GOTO

هذه الجمل هي التي تحدد مسار البرنامج هل يسير في الخط الطبيعي أم يتفرع إلى دالة فرعية وهي التي تحقق رغبات المستخدم حيث لا يمكن أن يكون هناك برنامج بل جمل تحكم .

و يتضح ذلك من خريطة التدفق الموضحة في الشكل (2-4):



الشكل (2-4)

في هذا الشكل نلاحظ وجود جملة شرط تحدد مسار البرنامج فإذا كان yes يتجه التنفيذ إلى اتجاه معين و إذا كان NO أتجه اتجاه آخر .

و فيما يلي نشرح جمل التحكم :

في لغة VB.NET

جملة IF

تستخدم جملة if كما هي بدون else إذا كان هناك اختبار تريد اختباره إذا تحقق ننفذ جمل معينة و إلا يسير البرنامج طبيعي و لا ينفذ هذه الجمل و يأخذ ذلك الصور العامة التالية:

```
If condition Then [statements] [Else elsestatements]
Or, you can use the block form syntax:
If condition Then
[statements]
[ElseIf condition-n Then
[elseifstatements] ...
[Else
[elsestatements]]
End If
```

وجملة if تأخذ شكلين:

الصورة الأولى:

If Condation then Statement

وهي صورة بسيطة لا تحتاج الى End If وتستعمل عندما تكون نتيجة الشرط عبارة عن جملة واحدة

الصورة الثانية

```
if ... condation ... then
Statements 1
Statements 2
End if
```

في هذه الصورة إذا كان الشرط condation صحيح ننفذ الجمل statements 1 , statements 2 و ينهي ذلك كلمة end if .

تركيب IF ...ELSE IF

يستعمل هذا التركيب عندما يكون هناك أكثر من احتمالين (شرطين) نريد اختبارهما و يأخذ الصورة :

```
IF CONDATION
ELSE IF CONDATION THEN
```

```
ELSE IF CONDATION THEN
ELSE
END IF
```

في هذه الصورة

اذا تحقق الشرط الاول يختبر الشرط الثاني وهكذا.

لاحظ المثال الموجود بالمسار EX12 بأسم NESTED_IF.

التركيب SELECT ... CASE

يستعمل هذا التركيب محل التركيب السابق و هو اختيار أكثر من حالتين و هو أسهل في التركيب و الأستعمال عن التركيب السابق .

و يأخذ الصورة العامة التالية:

```
Select Case testexpression
[Case expressionlist-n
[statements-n]] ...
[Case Else
[elsestatements]]
End Select
```

والصورة التالية هي نفس الصورة بشكل أبسط.

```
SELECT CASE VAR
CASE 1
Statement 1
CASE 2
Statement 2
CASE3
Statement 3
Else case4
Statement 4
END SELECT
```

في هذه الصورة

يتم اختبار المتغير var ثم تحديد قيمته حسب الحالات الموجودة في cases وبالتالي إذا كانت قيمة المتغير VAR 1 يتم تنفيذ الجملة State ment 1 وهكذا مع Case 2 وباقي الحالات وإذا لم تتحقق أحد الحالات تنفذ الجملة Else Case.

كما يأخذ هذا التركيب أكثر من صورة سوف نوضحها مع الأمثلة :

الصورة الاولى

```
SELECT  CASE VAR
CASE 1

CASE 2

CASE3

Else case

 END SELECT
```

وهى الصورة التقليدية والتى شرحناها فى الصورة السابقة

الصورة الثانية:

```
SELECT CASE VAR
CASE Is > 5
Statement 1
CASE Is > 10
Statement 2
CASE3 Is > 15
Statement 3
Else case
Statement 4
END SELECT
```

وفى هذه الصورة يتم اختبار قيمة المتغير هل هى أكبر أم أصغر من قيمة معينة ففى هذه الحالة إذا كانت قيمة المتغير VAR أكبر من 5 تنفذ الحالة الأولى وتنفذ الجملة Statement 1 وهكذا باقى الحالات:

الصورة الثالثة:

```
SELECT CASE VAR
CASE VAR =1 TO 5
Statement 1
CASE VAR =5 TO 10
Statement 2
CASE3 VAR =10 TO 15
Statement 3
Else case
Statement 4
 END SELECT
```

هذه الصورة جيدة جداً حيث تقوم باختيار قيمة المتغير VAR والحالات هى مدى فمثلاً.

الحالة الاولى: تقول إذا كانت قيمة المتغير تقع بين القيمة 1 أو القيمة 5 تنفذ الجملة Statement 1 وفى هذه الصورة يتم اختبار قيمة المتغير هل هى تقع بين قيمتين أم لا وهكذا باقى الحالات.

مثال :

- قم بتصميم مثال يستقبل درجة الطالب ثم يختبر هذه الدرجة
- اذا كانت أقل من 50 تخرج رسالة اعتذار أن الطالب لم ينجح
- اذا كانت تبدأ من 50 وأقل من 64.9 تخرج رسالة تفيد أن الطالب نجح بتقدير مقبول
- اذا كانت تبدأ من 65 وأقل من 74.9 تخرج رسالة تفيد أن الطالب نجح بتقدير جيد وهكذا باقى التقديرات

خطوات التصميم

1- إبدأ برنامج جديد بالخطوات المعتادة.

2- وقع أداة كتابة TextBox وأداة زر Button1.

3- اكتب فى دالة زر الامر السطور التالية:

```
Dim t
t = Val(TEXTBOX1.TEXT)
Select Case t
   Case t = 0 To 49.9
      MsgBox("SORRY....")
   Case t = 50 To 64.9
      MsgBox("OK YOU PASSED...")
   Case t = 65 To 74.9
      MsgBox("OK YOU ARE GOOD...")
   Case t = 75 To 84.9
      MsgBox("OK YOU ARE VERY GOOD...")
   Case t = 85 To 100
      MsgBox ("OK YOU ARE EXCELLENT...")
   Case Else
      MsgBox("OUT OF RANG..")
End Select
```

جملة GOTO

تستخدم GOTO لتغير مسار البرنامج وتوجه تنفيذ البرنامج إلى خطوة معينة وإن كانت غير مستحبة فى عالم البرمجة وتأخذ الشكل التالى:

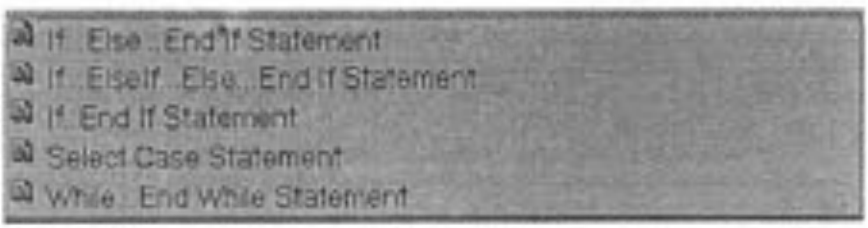
```
GoTo line
```

وتوضع فى الأمثلة كما فى الشكل التالى:

```
IF A=10 THEN GOTO R
--
--
--
--
R:
--
--
--
```

الاستفادة من الـ Snippet

تعرض قائمة Snippet جميع الصور المتاحة لجمل الشرط Condations كما فى الشكل



كما يلى:

```
    If True Then

Else

End If
```

هذه الصيغة التقليدية لجملة الشرط If Else

```
    If True Then
ElseIf False Then
    Else
    End If
```

تنفذ الجملتين STATEMENT1, STATEMENT2 و إلا (إذا كان الشرط غير صحيح) تنفذ الجمل التي بعد ELSE و هي في الصورة STATEMENT3, STATEMENT4.

و لتوضيح ذلك تابع المثال التالي:

```
1. If (x>y)
2. {
3. System.out.Println ("x is greater than y – ");
4. System.out. Println ("so you can go -- ");
5. }
6. else
7. {
8. System.Out.println ("x is less than y – ");
9. System.Out.println ("so you can not go -- ");
10. }
```

في هذه السطور:

يتم اختبار الشرط (x>y) فإذا كان صحيح يتم تنفيذ الجمل التى في السطور 3,4 وإلا (else) تنفذ الجمل في السطور 8,9.

مؤثر الشرط Conditional Operator

بالإضافة لجملة if يوجد مؤثر الشرط الذى يحقق عمل لأمر if --- else بشكل بسيط كما يلى:

```
Condition ? trueresult: false result;
```

في هذا السطر:

Condition: هو الشرط المطلوب اختباره.

True result: ينفذ إذا كان الشرط صحيح.

False result: ينفذ إذا كان الشرط خطأ.

وبالتالى يقول هذا المؤثر إذا كان الشرط Condition صحيح قم بتنفيذ الجملة المكتوبة في مكان True result وإلا قم بتنفيذ الجملة الأخرى false result ويظهر ذلك من المثال التالى:

```
k= (x > y) ? 10:20:
```

فى هذا المثال تأخذ k القيمة 10 إذا كانت x أكبر من y

وتأخذ k القيمة 20 إذا كانت x أصغر من y

وهذا المؤثر يكافئ جملة If..Else كما يلى:

```
if (x > y)
k=10;
else
k=20;
```

التركيب الشرطى Switch

هذا التركيب يستعمل بدلاً من استعمال اكثر من جملة else --- if وذلك لتحديد قيمة من عدة قيم فمثلاً إذا كانت لديك قيمة وتريدا اختبارها واحتمال ان تأخذ اكثر من نتيجة نستعمل لذلك التركيب Switch. الذى يأخذ الشكل التالى:

```
switch (expression)
{
   case constant-expression:
      statement
      jump-statement
   [default:
      statement
      jump-statement]
}
```

فى هذه الصورة

تبدأ بالكلمة switch وبداخلها المتغير expression الذى يتم اختباره ثم تبدأ الحالات المتوقعة لقيمة هذا المتغير VAR.

- الحالة الأولى Case Value One تقول فى حالة القيمة الأولى تنفذ النتيجة الأول do result one وهكذا.
- وفى النهاية default التى تنفذ فى حالة عدم تحقق أى حالة مع ملاحظة أن الكلمات المحجوزة فى هذه الصورة هى:

switch, case, break, default.

الكلمة switch : لاختبار قيمة المتغير.

الكلمة case : لتحديد الحالات المختلفة للمتغير.

فى هذه الصيغة يتم إضافة ElseIf لإضافة شرط آخر.

```
If True Then

End If
```

هذه الصيغة أبسط صور الشرط والتى تحتوى على شرط واحد.

```
Select Case VariableName
   Case 1
   Case 2
   Case Else
End Select
```

فى هذه الصيغة يتم عرض التركيبSelect Case الذى يسمح بتحديد اختيار من مجموعة اختيارات.

فى لغات C-Like

فى هذه الفقرة نتناول الجمل التالية:

جملة ..... if

جملة ...if.....Else

جملة ....if .....End if.... else if

جملة switch case....

جملة goto

ويمكن تنفيذها فى Java,

جملة IF

تستخدم جملة if كما هي بدون else إذا كان هناك اختبار نريد اختباره إذا تحقق ننفذ جمل معينة و إلا يسير البرنامج طبيعي و لا ينفذ هذه الجمل و يأخذ ذلك الصور العامة التالية:

```
if (expression)
   statement1
[else
   statement2]
```

وجملة if تأخذ شكلين:

الصورة الأولى: If Condation then Statement

وهى صورة بسيطة لا تحتاج الى End If وتستعمل عندما تكون نتيجة الشرط عبارة عن جملة واحدة.

الصورة الثانية:

```
if (condation )
{
Statements 1
Statements 2
}
```

مثال:

```
if (x>y)
System.out. Println ("x is greater than y");
```

فى هذا السطر يتم استعمال if لاختبار الشرط (x>y) فإذا كان صحيح اى ان قيمة x اكبر من قيمة y يتم تنفيذ الجملة التالية للشرط وهى طباعة العبارة الموجودة.

فى هذا المثال يتم تنفيذ جملة واحدة كنتيجة للشرط ولكن إذا أردت تنفيذ أكثر من جملة تحصرهم بين الأقواس ( ) لتصبح بلوك فينفذ كنتيجة للشرط .بالشكل التالى:

```
if (x>y)
{
System.out. Println ("x is greater than y");
System.out. Println ("x is greater than y");

}
```

جملة if ----else

يستعمل هذا التركيب لاختبار شرط معين فإذا كان صحيح ينفذ ما بعد الـif وإذا كان غير صحيح ينفذ ما بعد else ويظهر ذلك من الشكل التالي :

```
if (expression)
   statement1
[else
   statement2]
```

في هذه الصورة يتم اختيار الشرط CONDATION بجملة IF فإذا كان الشرط صحيح

الكلمة break : للخروج من التركيب الشرطى switch عند تحقق حالة.

الكلمة default : لتنفيذ مجموعة من الجمل فى حالة عدم تحقق أى حالة.

مثال:

```
public class Switchcase {

   public static void main(String[] args) {
   int v=1;
      switch (v)
{
   case 0 :
    System.out.println ( "Zero");
   break;
   case 1 :
    System.out.println ( "One");
    break;
   case 2 :
    System.out.println ( "Two");
    break;
   case 3 :
    System.out.println ( "Three");
    break;
   case 4 :
    System.out.println ( "Four");break;
    case 5 :
    System.out.println ( "Five");break;
   case 6 : System.out.println ( "Six");break;
   case 7 : System.out.println ( "Seven");break;
   case 8 : System.out.println ( "Eight");break;
   case 9 : System.out.println ( "Nine");break;
   default :System.out.println ( " ");break;
   }
   }
}
```

فى Oracle PL/SQl

التحكم في مسار البرنامج Flow of process

غالباً ما تحتاج التحكم فى سير البرنامج بتوجيهه إلى عملية معينة بشرط معين أو تكرار تنفيذ عمليات معينة أكثر من مرة.

- ويتم الاختبار وتوجيه التنفيذ باستعمال الأمر if...then...else end if.
- ويتم تكرار تنفيذ العمليات باستعمال جمل تكرار وفيها for,loop,while وغيرها.
- ونبدأها بجمل الشرط.

1. الأمر IF

ونأخذ جملة IF الشكل العام

```
IF condition THEN
  statements;
[ELSIF condition THEN
  statements;]
[ELSE
  statements;]
END IF;
```

في هذه الصورة:

يتم استعمال الأمر IF لاختبار الشرط Condition فإذا كان صحيح يتجه التنفيذ إلى ما بعد then وإلا يتم توجيه التنفيذ إلى ما بعد ELSE.

- وفي النهاية يتم إنهاء بلوك IF بالكلمة END IF.
- ويمكن تكرار الشرط بكتابة ELSE IF بدلاً من ELSE.

```
Set serveroutput on;
DECLARE
  v_TotalStudents NUMBER;
BEGIN
  -- Retreive the total number of students from the database.
  SELECT COUNT(*)
    INTO v_TotalStudents
    FROM students;
    Dbms_output.put_line('number of students='|| v_TotalStudents);
  -- Based on this value, insert the appropriate row into temp_table.
  IF v_TotalStudents = 0 THEN
    INSERT INTO temp_table (char_col)
      VALUES ('There are no students registered');
  ELSIF v_TotalStudents < 5 THEN
    INSERT INTO temp_table (char_col)
      VALUES ('There are only a few students registered');
  ELSIF v_TotalStudents < 10 THEN
    INSERT INTO temp_table (char_col)
      VALUES ('There are a little more students registered');
  ELSE
    INSERT INTO temp_table (char_col)
      VALUES ('There are many students registered');
  END IF;
END;
/
```

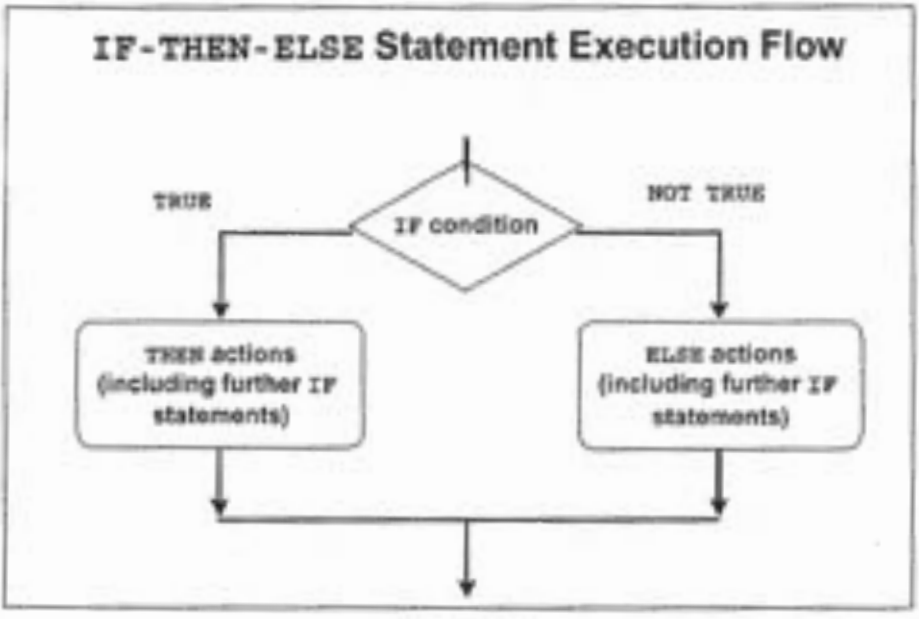
فى هذه السطور:

يتم استعمال الأمر IF لاختبار الشرط Condition فإذا كان صحيح يتجه التنفيذ إلي ما بعد then وإلا يتم توجيه التنفيذ إلي ما بعد ELSE ويبدأ ذلك باختبار عدد الطلاب فى الجدول students باستعمال الأمر SELECT مع الدالة COUNT ووضع النتيجة فى المتغير v_TotalStudents ثم استعمال الأمر IF لاختباره فاذا كان العدد أقل من 0 يتم استعمال الامر INSERT لاضافة العبارة 'There are no students registered' فى الجدول TABLE_TEMP ثم يتم استعمال الأمر ELSIF شرط آخر وهو اختبار قيمة المتغير v_TotalStudents إذا كانت أقل من 5 وهكذا .

وفى النهاية يتم إنهاء بلوك IF بالكلمة END IF ويمكن تكرار الشرط بكتابة ELSE IF بدلاً من ELSE.

والشكل التالى يعرض مخطط يوضح منطق تنفيذ جملة IF فى حالة ما إذا كان الشرط صحيح أو غير صحيح ويوضح ما تنفذ نتيجة IF ومتى تنفذ نتيجة ELSE.

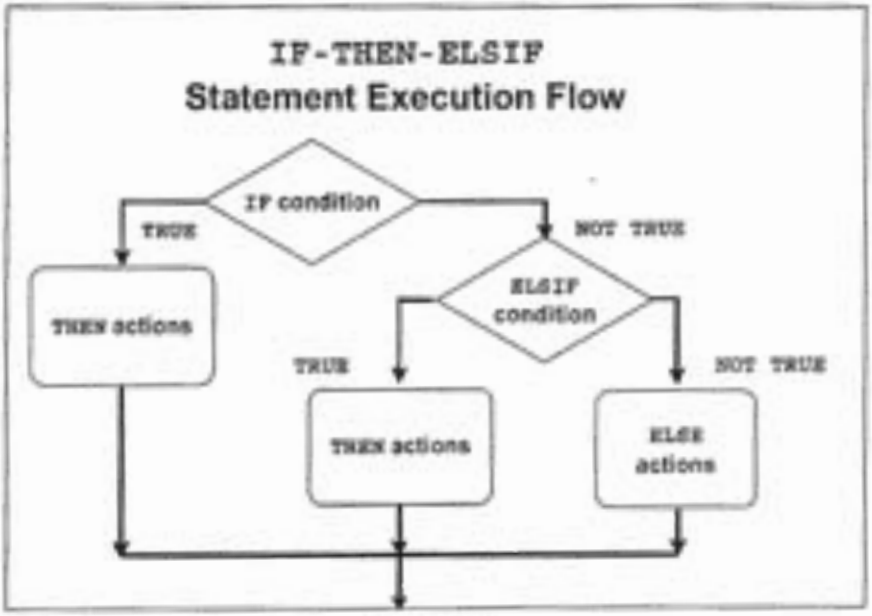
كما فى الشكل (3-4)



الشكل(3_4)

الأمر IF ELSIF

يستعمل هذا الأمر في حالة تعدد الشروط بمعنى أنك لديك أكثر من اختبار ويتضح ذلك من الشكل (4-4).



شكل (4_4)

مثال:

```
SET SERVEROUTPUT ON ESCAPE OFF

DECLARE
  v_price books.price%TYPE;
  v_isbn books.isbn%TYPE := '12345678';
BEGIN
  SELECT price
  INTO v_price
  FROM books
  WHERE isbn = v_isbn;

  DBMS_OUTPUT.PUT_LINE('Starting price: '||v_price);

  IF v_price < 40
  THEN
    DBMS_OUTPUT.PUT_LINE('This book is already discounted');
  ELSIF v_price BETWEEN 40 AND 50
```

```
  THEN
    v_price := v_price - (v_price * .10);

    UPDATE books
    SET price = v_price
    WHERE isbn = v_isbn;
  ELSIF v_price > 50
  THEN
    v_price := v_price - (v_price * .10);
    UPDATE books
    SET price = v_price
    WHERE isbn = v_isbn;
  END IF;

  DBMS_OUTPUT.PUT_LINE('Ending price: '||v_price);

  ROLLBACK;
EXCEPTION
  WHEN OTHERS
  THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
    ROLLBACK;
END;
/
```

الشكل(4_4)

في هذه السطور:

- تم وضع قيمة الحقل price أي سعر الكتاب لكتاب محدد في المتغير v_price ثم تم اختباره.
- باستعمال الأمر if ثم أجريت عليه عدة اختبارات باستعمال ELSIF وفي كل مرة يتم خصم قيمة من هذا السعر حسب السعر.

وبالتالي يكون الغرض من هذه السطور هي تحديد سعر الكتاب بعد الخصم.

استعمال Case

بالإضافة لأوامر IF المختلفة لاختبار المتغير وتحديد القيمة وعلى أساسها التصرف يوجد الأمر CASE الذي يسمح لك باختيار أكثر من قيمة للمتغير وهو أبسط من استعمال IF ELSEIF.

ويأخذ هذا الأمر الصيغة العامة التالية:

```
CASE selector
   WHEN expression1 THEN result1
   WHEN expression2 THEN result2
   ...
   WHEN expressionN THEN resultN
  [ELSE resultN+1;]
END;
```

في هذه الصيغة:

- يتم اختبار قيمة المتغير selector باستعمال الأمر CASE ثم تحديد حالة من حالاته.
- ويتم ذلك باستعمال الامر WHEN ثم اختبار الشرط ثم الأمر THEN ثم نتيجة تحقق الشرط result1.
- وهكذا حتى تنتهي الحالات وفي النهاية ELSE.
- التي تنفذ حالة عدم تحقق جميع الحالات السابقة.
- ثم END لإنهاء بلوك .

ولتوضيح ذلك تابع المثال التالي:

```
SET SERVEROUTPUT ON;
DEFINE P_GRADE=A;
DECLARE
V_GRADE CHAR(10) := UPPER('&P_GRADE');
V_APPRAISAL VARCHAR2(20);
BEGIN
V_APPRAISAL := CASE V_GRADE
WHEN 'A' THEN 'EXCELLENT'
WHEN 'B' THEN 'VERY GOOD'
WHEN 'C' THEN ' GOOD'
ELSE 'NO SUCH GRADE'
END;
DBMS_OUTPUT.PUT_LINE('GRADE:'|| V_GRADE ||' APPRAISAL '|| V_APPRAISAL);
END;
```

فى هذه السطور:

تم تعريف متغير بالاسم p_grade ووضع الحرف A فيه وذلك فى بيئة SQL PLUS ثم ابتداء بلوك PL/SQL ثم وضع قيمة هذا المتغير فى متغير البلوك v_grade بعد تحويله الى حرف كبير باستعمال الدالة UPPER() .

- ثم تم تعريف متغير من نوع varchar2.
- ثم تم استعمال الامر CASE لاختبار قيمة المتغير v_grade.
- ثم تم استعمال أكثر من WHEN لتحديد أحد الحالات.

وفى النهاية يتم طباعة النتيجة وهى EXCELENT حيث أن قيمة المتغير هى A.

الأمر GoTo

هذا الأمر يؤدى إلى انتقال التنفيذ إلى سطر محدد ويفيد عند استعمال الأمر If فإذا تحقق شرط معين يتم الانتقال بالتنفيذ إلى سطر معين ولا يتم تنفيذ سطور أخرى .

ولتوضيح ذلك تابع السطور التالية:

```
SET SERVEROUTPUT ON
BEGIN
   GOTO l_Last_Line;
DBMS_OUTPUT.PUT_LINE('GOTO didn"t work!');
   RETURN;
   <<l_Last_Line>>
DBMS_OUTPUT.PUT_LINE('Last Line');
END;
/
```

فى هذه السطور:

يتم استدعاء الأمر GOTO الى يؤدى إلى الانتقال إلى آخر سطر وبالتالى يطبع العبارة Last Line.

الفصل الخامس

الدوال
Functions, Procedures

Programming Concepts 5

فى هذا الفصل

في هذا الفصل نشرح موضوع من أساسيات البرمجة التركيبية لأى لغة وهو إنشاء دوال فرعية FUNCTIONS أو Procedures و ذلك من خلال النقاط التالية :

- معني الدالة و البرنامج الفرعي (Function & Procedure)
- الفرق بين الدالة Functionو البرنامج الفرعي Procedure (الأجراء)
- كيفية إنشاء وأستعمال برنامج فرعي Procedure (الأجراء)
- كيفية انشاء و استعمال دالة Function
- ملفات الدوال و البرامج الفرعية Modules
- برنامج من النوع Console Application

معني الدالة و البرنامج الفرعي FUNCTION & SUB

إذا سبق لك دراسة لغة من لغات البرمجة الأخري مثل لغة C و غيرها فأنت تعرف أن أي برنامج يتكون من برنامج رئيسي و برامج فرعية و دوال يتم استدعائها من البرنامج الرئيسي أي أن الدالة و البرنامج الفرعي هما مجموعة من الأوامر التي تكتب داخل اطار اسم برنامج فرعي يسمي دالة FUNCTION أو SUB يتم إنشائه مرة واحدة و يمكن استدعائة أكثر من مرة مما يوفر اعادة كتابة هذه الأوامر أكثر من مرة.

الفرق بين الدالة FUNCTION و البرنامج الفرعي Procedure

الفرق الأساسي بين ال FUNCTION و Procedure (الإجراء) هو أن الـ FUNCTION لابد أن تعيد قيمة عند استدعائها مثل دالة حساب AVGأو SIN أو أي دالة من دوال اللغة.

بينما يقوم البرنامج الفرعي Procedure (الإجراء) بتنفيذ مجموعة من السطور دون أن تعيد قيمة بينما ينفذ الجمل لتحقيق حدث و سوف يتضح ذلك عند التعامل معهما .

ما هي الدوال Methods

الدالة Method هي مجموعة من سطور البرمجة التي تتحد معا لتحقيق وظيفة معينة ، ويمكنك أن تقول أنها برنامج فرعي يؤدي وظيفة محددة

وهي تتصف بما يلي:

1- برنامج فرعي يمكن استدعئه أكثر من مرة
2- هي بلوك من الكود لتحقيق وظيفة محددة
3- وهي غالبا ما تحتوى على مدخلات Input ومخرجات output (ولكن ليس بالضروري)
4- أحيانا تسمى "subroutines" or "functions"

لماذا نحتاج إلى الدوال

1. Modularity جعل البرنامج مركب

بالطبع يصبح البرنامج كبير ويحتوى على عدد كبير من السطور ..، وبالتالى من

المفضل تقسيم هذا البرنامج إلى برامج صغيرة Methods واستدعائها ، وبالتالى يتم تقسيم وظائف البرنامج إلى دوال Methods

2. Code Re-use اعادة الاستخدام

من أشهر قواعد البرمجة والتى حققتها الدوال methods نظرية كتابة الكود مرة واحدة ثم اعادة استخدامه أكثر من مرة. وهى من القواعد المشهورة جدا والمهمة جدا ، وهو اتجاه تطوير البرمجة من أول إنشائها وحتى الآن وفى المستقبل ، ومن أول ما حقق هذا المفهوم وجود الدوال Methods

3. Abstraction إخفاء الأوامر

من فوائد الدوال Methods هو إنشاء دوال تحقق وظائف معينة وعند الاحتياج إلى وظائف هذه الدوال يتم استدعائها دون الرجوع إلى سطور هذه الدوال ، وسوف تتضح هذه النقطة أكثر مع درس Objects and Classes.

التعامل مع البرنامج الفرعي Procedure (الإجراء)

يتم التعامل مع البرنامج الفرعي Procedure أو الدالة خلال مرحلتين هما :

إنشاء البرنامج الفرعي (الإجراء)

استدعاء البرنامج الفرعي (الإجراء)

فى لغة VB.NET

(أ) إنشاء البرنامج الفرعي SUB

لتوضيح ذلك تابع الخطوات التالية :

من القائمة الرئيسية اختر PROJECT و منها اختر ADD MODULE كما بالشكل (1-5)

و بالتالي تكون خطوات التعامل مع البرنامج الفرعي هي :

1- إضافة ملف أوامر أول مرة (MODULE).

2- كتابة اسم و سطور البرنامج الفرعي S1.

3- استدعاء البرنامج الفرعي بكتابة اسمه من المكان المطلوب أستدعائه منه إرسال معاملات .

في المثال السابق تم أنشاء برنامج فرعي Procedure (الإجراء) بسيط لا يحقق شيء أكثر من إظهار رسالة ثابتة و لكن يمكن تغيير هذه الرسالة بإضافة معاملات للبرنامج الفرعي و لتحقيق ذلك تابع ما يلي :

1- أعد كتابة البرنامج الفرعي S1 ليصبح كما يلي :

```
Sub s1(ByVal st As String)
    MsgBox(st)
End Sub
```

- الجديد في هذه السطور هو إضافة المتغير ST من نوع STRING و هذا يعني أن S1 لها معامل من نوع STRING يتم ارساله إليها عند الاستدعاء، ولابد من إرسال المعامل وألا تحصل على رسالة خطأ.
- في السطر رقم 2 تم أستعمال الدالة MSGBOX لعرض رسالة و لكن هذه الرسالة هي محتوي المتغير ST الذي يأخذ قيمة عند استدعاء S1 من المعامل.

2- عدل دالة زر أمر استدعاء S1 ليصبح كما يلي :

```
S1 (" THIS MSG PASSED TO S1 AS A PARAMETER")
```

- في هذا السطر تم استدعاء S1 كما سبق و لكن الجديد هو ارسال معامل لها هو الرسالة المطلوب اظهار ما بين " " و ذلك لأن المعامل من النوع STRING و بالتالي عند الاستدعاء ترسل الرسالة إلي المعامل ST في البرنامج الفرعي (الإجراء) الذي يأخذ هذا المعامل (الرسالة) ويضعه أمام الدالة MSGBOX فتظهر الرسالة .

و بنفس الأسلوب يمكن زيادة أو تغيير المعاملات من حيث النوع و العدد و لتوضيح ذلك تابع المثال التالي :

```
1: sub s2 (no as integer , msg  as  string )
2: for I=0 to no
3: Msgbox msg
4: next I
5: end sub
```

في هذه السطور

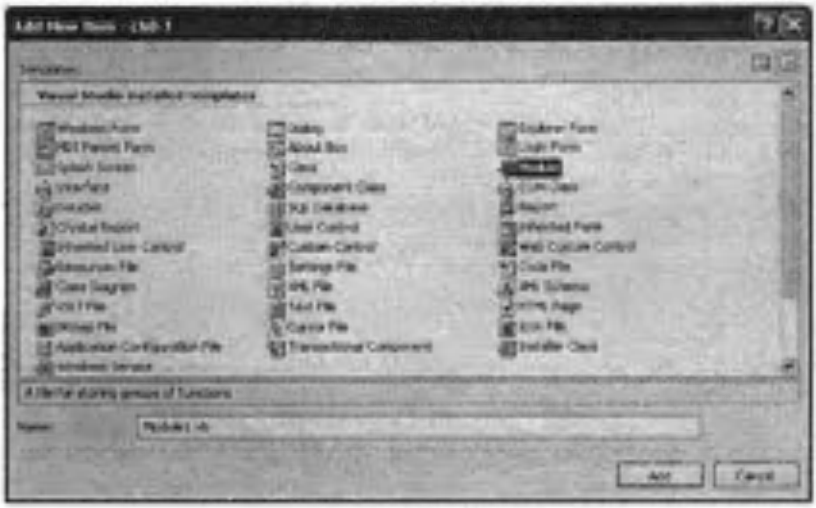
- في السطر رقم 1 يبدأ البرنامج الفرعي بأسم s2 مع تعريف معاملين هما no من نوع رقم integer و msg من نوع حرفي string.
- في السطر رقم 2 يبدأ التكرار باستعمال for و لكن يبدأ من 0 إلى قيمة المتغير no التي يرسلها برنامج الاستدعاء الفرعي .
- في السطر رقم 3 يتم عرض رسالة هي قيمة المتغير msg و هو المعامل التالي القادم من استدعاء البرنامج الفرعي (الإجراء).
- و باقي السطور كما سبق.
- وقع زر أمر علي خلفية البرنامج (From 1) و أكتب في دالتة السطر التالي:
- S2 10 , " VISUAL BASIC.Net ..."
- في هذا السطر تم استدعاء البرنامج الفرعي S2 مع أرسال معاملين له هما القيمة 10 و الرسالة ... VISUAL BASIC.Net تذهب إلي المعامل الأول في البرنامج الفرعي وهو NO القيمة 10.
- و المعامل الثاني هو الرسالة و بالتالي يتم طباعة هذه الرسالة عدد NO من المرات كما في نص البرنامج الفرعي S2.

مما سبق نلاحظ إمكانية تصميم (إنشاء) برنامج فرعي من نوع SUB (الأجراء) و ذلك بالأشكال التالية :

- بدون معاملات.

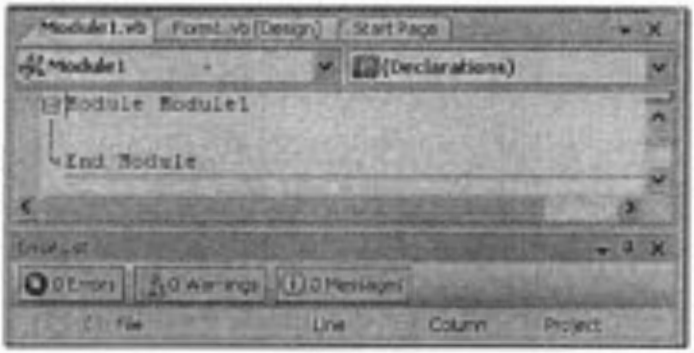
الشكل (5-1)

1- تحصل على شكل به مجموعة من الـ Modules وتحصل على صفحة بيضاء لكتابة سطور كما في الشكل (5-2) .



الشكل (5-2)

2- اضغط زر OPEN يتم إضافة ملف أوامر يظهر في مربع المشروع كما يتم فتحه. كما في الشكل (5-3) .



الشكل (3-5)

3- في مربع المشروع اضغط MODULE1 مرتين تحصل في الجانب الأيسر على خلفية بيضاء

4- في الصفحة البيضاء أكتب SUB S1 ثم أضغط مفتاح ENTER تلاحظ كتابة END SUB كما بالشكل التالي:

```
Module Module1
   Sub s1()

   End Sub
End Module
```

بهذا تم إنشاء برنامج فرعي Procedure (الإجراء) بالاسم S1 و لكنة لا يؤدي وظيفة لأنه لا يحتوي على اي أوامر و لتحقيق وظيفة ما أكتب بين SubS1() و END SUB السطر التالي :

```
MSGBOX " THIS LINE CALLED FROM SUB S1 "
```

و لاختبار هذا البرنامج الفرعي Procedure (الإجراء) تابع ما يلي :

- استدعاء البرنامج الفرعي Procedure CALLING.
- وقع زر أمر على خلفية البرنامج Form ثم أكتب في دالته السطر التالي:

```
S1()
```

- في هذا السطر تم استدعاء البرنامج الفرعي بكتابة اسمه S1.
- نفذ البرنامج ثم اضغط الزر CALL S1 تلاحظ ظهور الرسالة التي تم كتابتها في البرنامج الفرعي S1 وهي الرسالة THIS LINE CALLED FROM SUB S1.

- بمعامل واحد.
- أكثر من معامل.

كما تلاحظ أنه بعد إنشاء البرنامج الفرعي يمكن استدعائه من أي مكان بكتابة اسمه فقط إذا لم يكن له معاملات ، و كتابة اسمه و أمامه المعاملات بترتيبها إذا كان له معاملات .
و أيضا تلاحظ إمكانية أنشاء البرنامج الفرعي مرة واحدة و أستدعائة أكثر من مرة وأيضا البرنامج الفرعي من نوع sub (الإجراء) لا يعيد قيمة مثل الدوال كما سيلي.

القيمة الافتراضية للمعامل Optional Argumen

يمكن تحديد معامل أو أكثر على أنه Optional وفى هذه الحالة عند استدعاء البرنامج الفرعى Sub يمكن ارسال قيمة المعامل أو عدم إرسالها.
بشرط اعطاء هذا المعامل قيمة افتراضية عند الانشاء كما يظهر من السطور التالية:

```
Sub s1(Optional ByVal st As String = "hi")
      MsgBox(st)
End Sub
```

فى هذه السطور فى السطر الاول تم تعريف البرنامج الفرعى S1 مع اضافة معامل له بالاسم st كما سبق ولكن الجديد هو اضافة كلمة Optional التى تعنى ان ارسال قيمة هذا المعامل عند الاستدعاء ليس إجبارى مع تحديد قيمة افتراضية لهذا المعامل وهى "hi"
فى هذه الحالة يمكن استدعاء هذا الإجراء Sub بصورتين.
الأولى :عدم إرسال معامل بكتابة اسم الإجراء ()S1 فقط كما فى السطر التالى:

```
S1()
```

الثانية: إرسال معامل بكتابة اسم الإجراء ()S1 مع ارسال المعامل كما فى السطر التالى:

```
S1("new Value Sent")
```

من فضلك

- جرب اضافة زرى أمر Button وكتابة الأمر الأول فى دالة الزر الأول Button1
- وكتابة الأمر الثانى فى دالة الزر الثانى Button2.

ثم نفذ البرنامج وجرب استعماله وشاهد النتيجة

الفرق بين ByVal وByRef

فى الامثلة لاحظت كتابة الكلمة ByVal تلقائيا أمام المعاملات كما فى المثال السابق حيث كتبت أمام المعامل st وهذه الكلمة اختصار للعبارة By Value أى بالقيمة.

وهى تعنى أن استدعاء هذا المعامل يتم بإرسال قيمته إلى البرنامج الفرعى الذى يأخذ هذه القيمة ويجرى عليها العمليات المحددة فى سطوره.

فكما سبق فى المثال السابق يتم ارسال قيمة المعامل st وهى عبارة وفى البرنامج الفرعى S1 يتم اظهر هذه العبارة باستعمال الدالة Msgbox.

ولكن السؤال هو هل يمكن استعمال كلمة أخرى غير ByVal ؟

نعم توجد الكلمة ByRef وهى اختصار للعبارة By Reference وهى لا تؤدى إلى إرسال قيمة المعامل بل تؤدى إلى ارسال عنوان المتغير للتعامل معه بدلا من القيمة.

ولتوضيح ذلك تابع المثال التالى:

1- قم بكتابة البرنامجين الفرعين Inc1(),Inc2() داخل Module1 كما فى السطور التالية:

```
Sub inc1(ByVal v1 As Integer)
    v1 += 1
End Sub
Sub inc2(ByRef v1 As Integer)
    v1 += 1
End Sub
```

فى هذه السطور فى البرنامج الفرعى الأول تم تحديد المعامل مع تحديد كيفية التعامل معه وهى التعامل بالقيمة.

برنامج فرعي من نوع function (الدالة)

يأخذ البرنامج الفرعي من نوع function نفس خطوات الإنشاء و لكنه يختلف عن النوع sub في أنها لابد أن تعيد قيمة مثل الدوال الجاهزة في اللغة مثل دالة abs و دالة sin و غيرها من الدوال .

و لتوضيح ذلك تابع ما يلي :

مثال :

1- ابدأ برنامج جديد من النوع Windows Application.

2- أضف ملف أوامر MODULE.

3- أفتح ملف الأوامر لحصل علي صفحة بيضاء لكتابة أوامر.

4- أكتب السطور التالية :

```
Function f1(ByVal N1 As Integer, ByVal N2 As Integer)
    Dim s1 As Integer
    s1 = N1 + N2
    Return s1
End Function
```

في هذه السطور

- في السطر رقم 1 تم أنشاء دالة (FUNCTION) بالاسم F1 و لها معاملين الأول N1 و الثاني N2 و كلاهما من النوع Integer يتم استقبالهما من استدعاء الدالة (FUNCTION) التى تأخذ الاسم F1.
- في السطر رقم 2 يتم جمع المعاملين N2 , N1 ووضع النتيجة في المتغير S1.
- في السطر رقم 3 يتم اعادة قيمة S1 باستعمال الأمر Return وهو نتيجة استدعاء الدالة .

5- وقع زر أمر Button على خلفية البرنامج Form و أكتب في دالته السطور التالية :

```
R=F1 (5,7)
MSG BOX " RETURENED VALUE IS : " & R
```

- في السطر رقم 1 تم استدعاء الدالة F1 مع إرسال القيمة 7.5 كمعاملات للدالة فيتم أرسالها إلي سطور الدالة التي تستقبلها في المعاملين N1,N2 و تجمعهما و تعيد النتيجة التي تستقبل في المتغير R .
- في السطر رقم 2 يتم عرض القيمة المرتجعة و هي القيمة R مع رسالة تفيد ذلك.

- نفذ البرنامج و جرب النقر على زر الأمر تلاحظ ظهور الرسالة راجع الخطوات تجد أنها :

1. إضافة ملف أوامر Modale.
2. كتابة سطور الدالة FUNCTION بمعاملتها و عملياتها و أعادة النتيجة باستعمال الامر Return (في هذا المثال اسم الدالة F1).

1- استدعاء الدالة من أي مكان بكتابة أسم الدالة و معاملاتها ووضع النتيجة في متغير و الفرق الجوهري بين الدالة FUNCTION و البرنامج الفرعي SUB هو أن الدالة لابد أن تعيد قيمة بينما البرنامج الفرعي SUB (الإجراء) لا تعيد قيمة.

2- كما تكمن أهمية البرامج الفرعية Procedures و الدوال FUNCTION في إمكانية إنشائها مرة واحدة و استدعائها أكثر من مرة مما يوفر أعادة كتابة السطور مرة أخرى .

برنامج من النوع Console Application

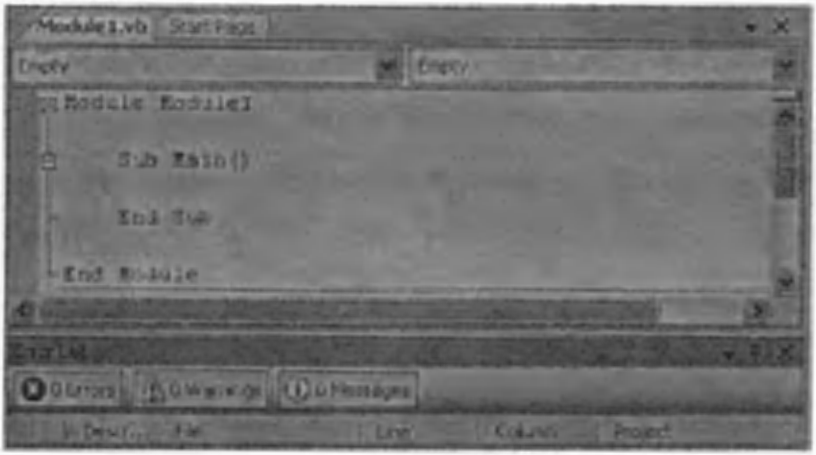
يوفر Visual Basic.Net أنواع اخرى من التطبيقات غير النوع التقليدى الذى نتعامل معه حتى الآن وهو Windows Application ومن هذه الأنواع برنامج من النوع Console Application وهو عبارة عن برنامج بدون واجهة (Form) وبالتالى عند تنفيذه تحصل على النتيجة فى شاشة خارجية مثل شاشة نظام التشغيل Dos ويفيد ذلك فى حالة عدم احتياجك لواجهة البرامج Form فقط تريد تحقيق عمليات فى الخلفية بدون خلفية أو تريد اختبار عمليات

ولتوضيح ذلك تابع الخطوات التالية:

1- قم بانشاء تطبيق جديد ولكن هذه المرة حدد النوع ConsoleApplication كما فى الشكل (4-5) .

الشكل (5-4)

2-اضغط الزر OK تحصل على تطبيق من هذا النوع كما في الشكل (5-5) .



الشكل (5-5)

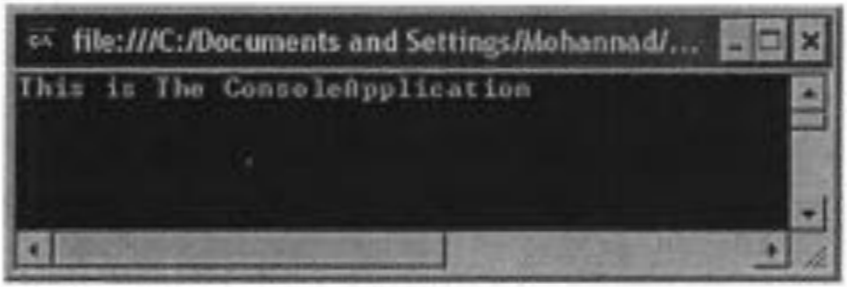
- في هذا الشكل نلاحظ وجود إجراء Sub بالاسم ()Main وهو الذي ينفذ عند تشغيل البرنامج اكتب السطر التالي:

```
Console.WriteLine("This is The ConsoleApplication ")
```

في هذا السطر يتم استدعاء الدالة WriteLine من الفصيلة Console

لطباعة العبارة This is The ConsoleApplication

3- شغل البرنامج بالضغط على Ctrl+F5 تحصل على نتيجة التنفيذ كما في الشكل (6-5)



الشكل (6-5)

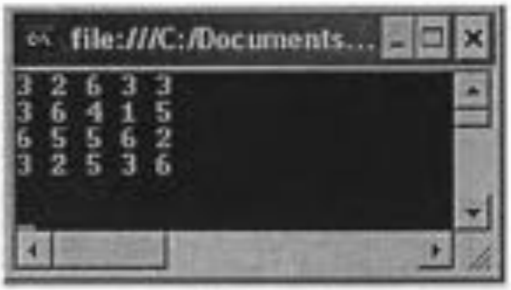
4- اضغط اى مفتاح تعود إلى بيئة التطوير.

ويمكن استعمال النوع من البرامج لاختبار قواعد البرمجة بسهولة حيث أنه أسرع حيث لا يحتاج إلى تحميل واجهة البرنامج Form والأدوات ولتوضيح ذلك تابع الخطوات التالية:

1- عدل سطور المثال السابق ConsoleApplication لتصبح كما في السطور التالية:

```
1:Dim rObject As Random = New Random()
2:Dim rNumber As Integer
3:Dim Resurt As String = ""
4:Dim i As Integer
5:For i = 1 To 20
6:rNumber = rObject.Next(1, 7)
7:Resurt &= rNumber & " "
8:If i Mod 5 = 0 Then
9:Resurt &= vbCrLf
10:End If
11:Next
12:Console.WriteLine(Resurt)
```

2- نفذ هذا البرنامج تحصل على النتيجة كما في الشكل التالي (7-5):



الشكل (7-5)

فى لغات C-Like

فى لغة Java

لا توجد Procedures توجد دوال فقط ويطلق عليها methods أو functions

وفى هذا المثال يتم انشاء دالة Method بسيطة لتوضيح فقط كيفية انشاء واستدعاء الدالة Method ولتحقيق ذلك اكتب السطور التالية:

```
1  public class Method1 {
2      public void Hello()
3      {
4          System.out.println("Hello from Method 1 ");
5          }
6      public static void main(String[] args)
7      {
8      Method1 Obj1=new Method1();
9      Obj1.Hello();
10     }
11 }
12
```

فى هذه السطور

فى السطر رقم 2 تم إنشاء دالة Method جديدة بالاسم ()Hello.

فى السطر رقم 4 وداخل هذه الدالة .. تقوم الدالة بطباعة العبارة "Hello from Method1".

فى السطر رقم 8 يتم تعريف متغير هدف من الفصيلة وهذا ما سوف نتناوله فى فصول البرمجة باستعمال الاهداف OOP.

فى السطر رقم 9 يتم استدعاء الدالة الجديدة ()Hello.

1.Types of methods أنواع الدوال

بعد إنشاء دالة بسيطة وتناول كيفية الانشاء وكيفية الاستدعاء .. تعالى معنا نستمل معا باقي عناصر هذا الموضوع وذلك بتناول ... أنواع الدوال.

وفيها يحدد نوع الدالة بنوع القيم المرتجعة من الدالة فاذا كانت الدالة تعيد قيمة من نوع int كانت الدالة من النوع int واذا كانت الدالة تعيد قيمة من النوع char كانت الدالة من النوع char وهكذا ، ولتوضيح ذلك تابع معنا الأمثلة التالية:

مثال :

المثال التالى يوضح كيفية انشاء دالة Method من النوع int كما فى السطور التالية :

```
1  public class Method1 {
2      public int Sum(int a ,int b )
3      {
4  int s;
5  s=a+b;
6  return s;
7          }
8      public static void main(String[] args)
9      {
10       int r;
11     Method1 Obj1=new Method1();
12     r=Obj1.Sum(10,15);
13
14       System.out.println("the sum is :"+r);
15     }
16 }
```

فى هذه السطور

فى السطر رقم 2 تم انشاء الدالة ()Sum من النوع int لانها تعيد قيمة int وهما معاملان من النوع int .

فى السطر رقم 5 يتم جمع قيم المعاملان ووضع النتيجة فى المتغير s .

فى السطر رقم 6 يتم استعمال الامر return لاعادة قيمة المتغير s وبالتالى عند استدعاء هذه الدالة تعود قيمة المتغير s .

فى السطر رقم 12 تم استدعاء الدالة ()Sum مع ارسال القيمتين 10و15 كمعاملات للدالة وبالتالى يتم إرسال هذان القيمتين إلى الدالة التى تقوم بجمعهما واعادة النتيجة التى توضع فى المتغير r .

وفى النهاية يتم طباعة قيمة المتغير r وهى القيمة المرتجعة من استدعاء الدالة .

مثال : دالة تعيد قيمة حرفية

وهذا المثال يستعمل التركيب Switch لاختبار الأرقام (0,1,2,...) وتحويلها إلى العبارة المقابلة فمثلاً يحول الرقم 1 إلى one وهكذا.وهو يشابه فكرة تقفيط الشيكات.

ولتحقيق ذلك قم بكتابة السطور التالية:

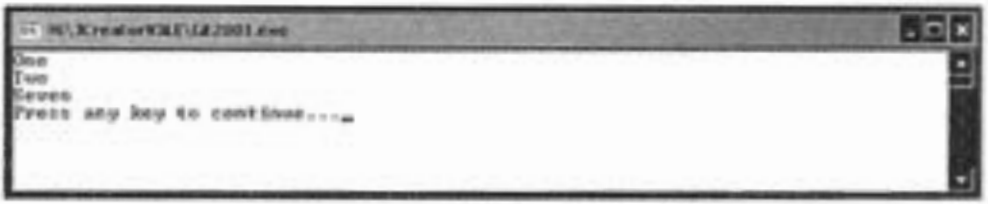
```
1  public class Method2 {
2
3      String Convfunc (int v) {
4   switch (v)
5     {
6   case 0 : return "Zero";
7
8   case 1 : return "One";
9   case 2 : return "Two";
10  case 3 : return "Three";
11  case 4 : return "Four";
12  case 5 : return "Five";
13  case 6 : return "Six";
14  case 7 : return "Seven";
15  case 8 : return "Eight";
16  case 9 : return "Nine";
17    default : return "";
18  }
19    }
20  public static void main(String[] args)
21    {
22    Method2 CN= new Method2();
23   String wordNum = CN. Convfunc (1);
24    System.out.println (wordNum);
25
26     wordNum = CN. Convfunc (2);
27    System.out.println (wordNum);
28
29     wordNum = CN. Convfunc (7);
30    System.out.println (wordNum);
31    }
32  }
```

فى هذه السطور

- فى السطر رقم 1 يتم إنشاء فصيلة بالاسم Method2.
- وفى السطر رقم 2 يتم إنشاء دالة عضوه بالفصيلة بالاسم Convertfunc.
- وفى السطور من 5 إلى 17 تم اختبار قيمة المتغير v الذى تستقبله الدالة وإعادة أحد الكلمات المقابلة حيث ان الدالة تعيد كلمة (string) حسب نوعها وبالتالى إذا كان الرقم 0 تعيد الدالة كلمة Zero وإذا كان 1 تعيد one وهكذا.

- وفى السطر 17 الكلمة default تعتبر مسافة " " إذا لم تتحقق أى حالة.
- فى السطر 20 تبدأ الدالة ()main التى تتولى سير البرنامج.
- فى السطر 22 يتم تعريف المتغير CN من نوع الفصيلة Convert Numb كعنصر object
- فى السطر 23 يتم تعريف المتغير Word Num من نوع string مع تخزين فيها نتيجة استدعاء الدالة Convert func مع إرسال القيمة 1 كمعامل فتقدم الدالة بتحويل 1 إلى بالمثل 3,2 وتصبح النتيجة one two three توضع فى المتغير word Num.
- فى السطر رقم 24 يتم طباعة محتوى المتغير word Num وهو one two three.

قم بكتابة البرنامج كما فى الشكل ثم قم بترجمة وتنفيذ البرنامج
تحصل على نتيجة التنفيذ كما فى الشكل (8-5)



الشكل (8-5)

دوال من غير قيم مرتجعة Methods with no returning values

كما اتفقنا يحدد نوع الدالة حسب القيمة المرتجعة ، فماذا إذا كانت الدالة لا تعيد قيمة... فى هذه الحالة تسمى الدالة بلا عائد أو void ، وهى الدوال التى تقوم بتنفيذ عملية محددة بدون اعادة نتيجة مثل دوال الرسم ودوال الصوت ...

وهذا ما تم فى المثال الاول حيث تم إنشاء دالة لا تعيد قيمة .. فقط تطبع رسالة وبالتالى كان نوعها void.

وهذا ما يقابل برنامج فرعى من نوع procedure فى لغة VB.NET حيث لا تحتوى لغات C Like برامج فرعية من نوع procedure ولكن فى المقابل تحتوى على دالة من نوع void بنفس الوظيفة

فى Oracle PL/SQL

إعداد الدوال والبرامج الفرعية Building procedures & functions

وفى هذه الفترة سوف نتناول خطوات ومتطلبات إعداد الدوال والبرامج الفرعية procedures & functions وذلك من خلال النقاط التالية:

- خطوات إنشاء Procedures.
- إرسال معاملات للبرامج الفرعية Parameter passing.
- إنشاء دوال Functions.

إنشاء Procedure

```
    PROCEDURE DEPT_SEARCH IS
    BEGIN
      SELECT DNAME INTO :TEXT_ITEM6  FROM DEPT WHERE
DEPTNO=:TEXT_ITEM5;
    END;
```

فى هذه السطور تعدل بما يناسب الكود

- فى السطر الأول تم انشاء PROCEDURE بالاسم DEPT_SEARCH مع تحديد معامل بالاسم P_DEPTNO من النوع NUMBER.
- وهذا يعنى أنه عند استدعاء هذه الدالة لابد من ارسال معامل من النوع NUMBER.
- فى السطر الثانى وبعد IS تم الاعلان عن متغير بالاسم V_DNAME من نوع حقل DNAME الموجود فى جدول DEPT.
- فى السطر الثالث يبدأ البرنامج الفرعى بالأمر BEGIN.
- فى السطر رقم 4 تم استدعاء أمر SELECT لعرض اسم الادارة DNAME فى المتغير V_DNAME.
- فى السطر رقم 5 يتم وضع قيمة المتغير V_DNAME فى مربع النص TEXT_ITEM5: حتى يظهر على الشاشة وبالتالى أصبح لدينا برنامج فرعى PROCEDURE تأخذ رقم الإدارة و تعرض اسم الإدارة.

تنفيذ البرنامج الفرعى PROCEDURE CALLING

يكتب سطر استدعاء البرنامج الفرعى DEPT_SEARCH كما فى السطر.

```
DEPT_SEARCH();
```

إرسال المعاملات Passing parameters

من الضرورى جداً الإعلان عن متغيرات كمعاملات داخل أقواس الدالة function أو البرنامج الفرعى procedure وذلك لأن المعاملات توسع استعمالها فمثلاً الدالة sin(x) الموجودة باللغة يمكن استدعائها عدد كبير من المرات مع إرسال أى قيمة مكان المتغير كمعامل للدالة وبالتالى يمكن للدالة حساب (SIN ((جا)) أي قيمة ، فى حين لو كانت هذه الدالة بدون معاملات فبالتأكيد فهى تحسب قيمة جا (SIN لقيمه واحدة فقط ولابد من إنشاء دالة (SIN لكل قيمه وهذا هو المستحيل، والمعاملات تأخذ ثلاثة أنواعها هي:

النوعIN : وهو الذى يستقبل قيمة عند استدعاء الدالة ولا يمكن إعادة قيمة فيه لمكان الاستدعاء.

النوع OUT : وهو الذى يأخذ قيمة تعود لمستدعى الدالة فقط ولا يستعمل لاستقبال قيم عند الاستدعاء.

النوع INOUT: وهو النوع الذى يسمح باستقبال القيم عند الاستدعاء فيه وكذلك إعادة قيم للخروج فيه.

ويتضح ذلك عند استدعاء البرنامج الفرعى الذى أنشأناه من قبل حيث يتم استدعائه بدون معاملات ولكن يتم تعريف المعاملات عند الانشاء ثم استعمالها عند الاستدعاء ولتوضيح ذلك تابع الخطوات التالية:

فى هذا السطر تم استدعاء البرنامج MYPROC1 مع إرسال القيمة 398 5 كمعامل له وبالتالى يتم استقبال القيمة في المتغير DEL-EMPID المعرف كمعامل من نوع IN ثم استعماله داخل السطور لمسح الموظف الذي كوده 398 5 وبالتالى يمكن استدعاء البرنامج MYPROC1 اكثر من مرة مع تغيير كود الموظف المطلوب حذفة

5- اضغط الزر OK تظهر صفحة كتابة الاوامر CODE EDITOR اكتب بها سطور البرنامج الفرعى التالى:

```
PROCEDURE DEPT_SEARCH(P_DEPTNO NUMBER) IS
V_DNAME DEPT.DNAME%TYPE;
BEGIN
  SELECT DNAME INTO V_DNAME FROM DEPT WHERE DEPTNO=P_DEPTNO;
  :TEXT_ITEM5:=V_DNAME;
END;
```

فى هذه السطور:

- فى السطر الأول تم انشاء PROCEDURE بالاسم DEPT_SEARCH مع تحديد معامل بالاسم P_DEPTNO من النوع NUMBER.
- وهذا يعنى أنه عند استدعاء هذه الدالة لابد من إرسال معامل من النوع NUMBER.
- فى السطر الثانى وبعد IS تم الاعلان عن متغير بالاسم V_DNAME من نوع حقل DNAME الموجود فى جدول DEPT.
- فى السطر الثالث يبدأ البرنامج الفرعى بالامر BEGIN.
- فى السطر رقم 4 تم استدعاء أمر SELECT لعرض اسم الإدارة DNAME فى المتغير V_DNAME.
- فى السطر رقم 5 يتم وضع قيمة المتغير V_DNAME فى مربع النص TEXT_ITEM5: حتى يظهر على الشاشة .

وبالتالى أصبح لدينا برنامج فرعى PROCEDURE تأخذ رقم الإدارة و تعرض اسم الإدارة.

استدعاء البرنامج الفرعى Procedure

ويتم استدعاء البرنامج الفرعى DEPT_SEARCH بنفس الطريقة ولكن مع تحديد معامل ثانى وهو معامل اسم الإدارة وذلك كما فى الشكل :

```
DEPT_SEARCH(:TEXT_ITEM4,:TEXT_ITEM5);
```

- فى هذه السطور تم استدعاء البرنامج الفرعى DEPT_SEARCH بمعامل IN وهو TEXT_ITEM4 الذى يكتب المستخدم به كود الإدارة
- والمعامل الثانى TEXT_ITEM5 المعامل من النوع OUT الذى يظهر به اسم الإدارة.

- وبهذا اصبح البرنامج الفرعي DEPT_SEARCH غير مرتبط بعناصر معينة بل يصلح للاستعمال في اى مكان

انشاء معامل من نوع IN OUT

بالإضافة لإمكانية إنشاء معامل من IN للادخال و معامل من النوع OUT للاخراج يمكن إنشاء معامل من النوع IN OUT لاستعماله للإدخال والإخراج في نفس الوقت ولتوضيح ذلك تابع الخطوات التالية:

1- قم بتعديل سطور البرنامج لتصبح كما في الشكل (9-5).

```
PROCEDURE DEPT_SEARCH(P_INOUT  IN OUT NUMBER) IS
BEGIN
  SELECT SAL INTO P_INOUT FROM EMP WHERE EMPNO=P_INOUT;

END;
```

الشكل(9-5)

في هذه السطور:

2- اكتب سطور استدعاء البرنامج الفرعي لتصبح كما في الشكل :

```
DEPT_SEARCH(:TEXT_ITEM4);
```

في هذه السطور

إنشاء دالة CREATING PL/SQL FUNCTION

الدالة Function في جميع اللغات بما فيها PL/SQL هي برنامج فرعي مثل PROCEDURE ولكن يختلف عنه في أن الدالة لا يمكن أن تعيد أكثر من قيمة فمثلاً الدالة SQRT(X) الموجودة باللغة تعيد قيمة واحدة فقط وهي الجذر التربيعي للقيمة X وهكذا أي دالة أخرى.

ويتم إنشاء الدالة باستعمال الأمر Create function في كلا من بيئة البرنامج SQL Pluse والبرنامج Procdure Builder كما في السطور التالية:

```
CREATE [OR REPLACE] FUNCTION function_name
    [ (argument1 [mode1] datatype1 [,argument2 [mode2] datatype2, ...] ) ]
RETURN datatype
IS/AS
    [ Local Declaration ]
Begin
      Executable Statements
[Exception
      exception handlers ]
END [function_name];
```

ولتوضيح ذلك تابع المثال التالي:

```
1: Create function doub_salary (EID IN NUMBER (1 ))
2: RETURN NUMBER
3: IS
4: OLDSA NUMBER;
5: BEGIN
6: Select salary into OLDSA FROM EMP
7: WHERE EMPNO = EID;
8: OLDSA = OLDSA * OLDSA;
9: RETURN OLDSA;
1 : END;
```

ملحوظة 1:

البرنامج PROCEDURE قد يستقبل معامل أو اكثر أوقد لا يستقبل وكذلك يعيد نتيجة أو أكثر أو لا يعيد نهائياً .

الدالة function قد تستقبل معامل أو أكثر أوقد لا تستقبل ولكن لا تستطيع إلا أن تعيد قيمة واحدة.

ملحوظة 2:

في حالة الدالة function لا يصلح أن تحدد نوع المعامل IN أو OUT أو IN OUT لأن الدالة لا تعيد إلا قيمة واحدة وتعيدها كما في المثال السابق باستعمال الأمر RETURN.

ملحوظة 3

فى حالة الدوال Function لابد أن تكون نوع المتغيرات و المعاملات من الأنواع الأساسية فى Oracle ،الدالة لابد أن تعمل على قيمة واحدة single row function وليس group لا ينفذ لك return واحدة فى الدالة لكن يمكن تواجد أكثر من جملة return لا تنفذ إلا واحدة.

القيم الافتراضية لمعاملات الدوال default parameters

من الامكانيات المتاحة عند التعامل مع الدوال إمكانية إعطاء قيم ابتدائية لمعاملات الدوال default parameters هذه القيم تفيد فى حالة استدعاء الدالة بدون قيم معاملات تأخذ المعاملات القيم الابتدائية وفى حالة إرسال قيم للمعاملات تأخذ معاملات الدالة القيم الجديدة ولتوضيح ذلك تابع المثال التالى:

مثال قيم المعاملات الافتراضية default parameters

فى هذا المثال نعرض كيفية إنشاء اجراءات بقيم افتراضية للمعاملات:

```
/*
 * AddNewBook.sql
 * This procedure will insert a new book into the books table.
 * It also demonstrates default parameters.
 */

CREATE OR REPLACE PROCEDURE AddNewBook(
  p_ISBN IN books.ISBN%TYPE,
  p_Category IN books.category%TYPE := 'Oracle Server',
  p_Title IN books.title%TYPE,
  p_NumPages IN books.num_pages%TYPE,
  p_Price IN books.price%TYPE,
  p_Copyright IN books.copyright%TYPE DEFAULT
    TO_NUMBER(TO_CHAR(SYSDATE, 'YYYY')),
  p_Author1 IN books.author1%TYPE,
  p_Author2 IN books.author2%TYPE := NULL,
  p_Author3 IN books.author3%TYPE := NULL) AS
```

```
BEGIN
  -- Insert a new row into the table using the supplied
  -- parameters.
  INSERT INTO books (isbn, category, title, num_pages, price,
                     copyright, author1, author2, author3)
  VALUES (p_ISBN, p_Category, p_Title, p_NumPages, p_Price,
          p_Copyright, p_Author1, p_Author2, p_Author3);
END AddNewBook;
/
```

في هذه السطور:

تم إنشاء procedure لها معاملات وتم اعطاء المعامل p_Category قيمة ابتدائية وهي القيمة 'Oracle Server' وبالتالي عند الاستدعاء إذا لم يتم إرسال قيمة لهذا المعامل يتم استعمال القيمة الافتراضية وإذا تم إرسال قيم يتم استعمال القيم الجديدة.

وفيما يلي أكثر من حالة لاستدعاء هذا الإجراء لتجربة المعاملات ذات القيم الافتراضية.

الاستدعاء الأول

```
-- We can avoid passing author2 and author3 since they have
-- default values:
BEGIN
  AddNewBook('00000000', 'Oracle Basics', 'A Really Nifty Book',
             500, 34.99, 2004, 1);
END;
/
ROLLBACK;
```

في هذه السطور يتم استدعاء الإجراء AddNewBook مع إرسال قيم لجميع المعاملات وتم تجاهل القيم الافتراضية التي فرضت عند تعريف الإجراء Procdure في حين كان من الممكن تجاهل هذه المعاملات وبالتالي تخزين القيم الافتراضية.

الاستدعاء الثاني

```
-- The same example, using named notation:
BEGIN
  AddNewBook(p_ISBN => '0000000',
             p_Category => 'Oracle Basics',
             p_Title => 'A Really Nifty Book',
```

```
        p_NumPages => 500,
        p_Price => 34.99,
        p_Copyright => 2004,
        p_Author1 => 1);
END;
/
ROLLBACK;
```

في هذه السطور:

تم الاستدعاء بقيم مختلفة .

الاستدعاء الثالث

```
-- Using all the default values:
BEGIN
  AddNewBook(p_ISBN => '00000000',
        p_Title => 'A Really Nifty Book',
        p_NumPages => 500,
        p_Price => 34.99,
        p_Author1 => 1);
END;
/

ROLLBACK;
```

في هذه السطور تم الاستفادة بجميع القيم الافتراضية Default Paramters

الفصل السادس

المصفوفات ARRAYS

في هذا الفصل Programming Concepts

6

في هذا الدرس نشرح موضوع المصفوفات Arrays بأنواعها و كيفية تمثيلها و استخدامها و ذلك من خلال النقاط التالية :

- معني المصفوفات و استخدامها .
- أنواع المصفوفات Arrays types.
- مصفوفة بعد واحد One Dimension.
- مصفوفة بعدين Two Dimension .
- خصائص المصفوفات

معني المصفوفة

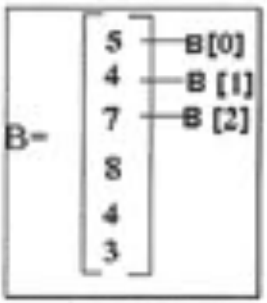
كما تعلم من الرياضيات أن المصفوفة Arrayعبارة عن مجموعة من القيم (العناصر) من نفس النوع و تأخذ أسم و تأخذ عدد للعناصر.

والمطلوب كيفية تمثيلها و استعمالها وكيفية التعامل مع عناصرها و للمصفوفة تطبيقات كثيرة لا تصلح إلا بها مثل عمليات البحث عن قيمة ضمن مجموعة قيم Search ومثل عمليات ترتيب مجموعة من القيم سواء تصاعدى أو تنازلى Sorting ومثل إيجاد أكبر قيمة ضمن مجموعة قيم Max وايجاد أقل قيمة Min والكثير من التطبيقات وخاصة فى مجال الدرسات العليا والأبحاث.

أنواع المصفوفات :

(أ) مصفوفة بعد واحد One Dimension كما بالشكل (6.1)

A= [5 4 7 8 4 3]



الشكل (1-6)

- المصفوفة A مصفوفة صف One Row وهي ذات بعد واحد حيث لا يوجد صفوف و أعمدة بل هي صف واحد فقط .
- المصفوفة B مصفوفة عمود واحد One Column ، لذلك يسمي كلا منهما مصفوفة البعد الواحد و يتم ترتيب العناصر بأسم المصفوفة و ترتيب العنصر مثل ، B[2] A[1] وهكذا.

(ب) مصفوفة متعددة الأبعاد :

و فيها يوجد أكثر من صف Row وأكثر من عمود Column كما بالشكل (2-6) .

C=

5	70	10
20	40	5
30	40	7
5	30	5
9	40	6
30	10	7

الشكل (2-6)

و في هذه المصفوفة يوجد 6 صفوف Rows و 3 أعمدة Columns لذلك تسمي 6 ⊕ 3 ويتم ترتيب العناصر بالصف Row ثم العمود Column كما بالشكل التالي :

- C (0, 0) العنصر الأول الذي يقع في الصف الاول رقم 0 و العمود الاول رقم 0
- C (0, 1) العنصر الثاني الذي يقع في الصف الاول رقم 0 و العمود الثانى رقم 1
- C (0, 2) العنصر الثالث الذي يقع في الصف 0 و العمود 2
- C (0,3) العنصر الرابع الذي يقع في الصف 0 و العمود 3
- C (1,0) العنصر الخامس الذي يقع في الصف 1 و العمود 0
- C (1,1) العنصر السادس الذي يقع في الصف 1 و العمود 1

و هكذا حتى باقي العناصر

مصفوفة البعد الواحد One Dimension :

و يتم التعامل مع هذه المصفوفة بالخطوات التالية :

1- الإعلان عن المصفوفة.

2- استعمال عناصر المصفوفة بملئها بالقيم أو استعمال القيم أو طباعة القيم.

فى لغة VB.NET

الإعلان عن المصفوفة

- يتم الإعلان عن المصفوفة كما يتم الإعلان عن المتغيرات و بنفس درجات الإعلان فيمكن الإعلان عنها و كأنها محلية Local داخل برنامج فرعي Sub أو على مستوي form أو على مستوي التطبيق كما يحدث مع المتغيرات و ذلك حسب المكان و بالصورة التالية :

```
Dim a(10) As Integer
```

أو يتم استعمال الصورة الحديثة التالية:

```
Dim Arr As Integer()=New Integer(10) {}
```

في هذه الصورة استعملت كلمة DIM لإعلان عن المصفوفة و المتغير A هو أسم المصفوفة و العدد 10 هو عدد عناصر المصفوفة و النوع Integer هو نوع عناصر المصفوفة و يمكن هذا النوع أن يتغير ليصبح من أي نوع (STRING,.....) حسب استعمال المصفوفة وفى الصورة الثانية تم الإعلان عن مصفوفة بالاسم Arr من النوع Integer ثم حجز هذه الأماكن فى الذاكرة باستعمال الكلمة New والإشارة إلى هذه الاماكن باسم المصفوفة.
و لتوضيح ذلك تابع المثال التالي :

مثال :

1- قم بانشاء تطبيق جديد ولكن من النوع ConsoleApplication كما سبق.

2- داخل الدالة الرئيسية Sub Main() اكتب السطور التالية:

```
1.  Dim a(10) As Integer
2.  Dim i As Integer
3.  For i = 0 To 10
4.  a(i) = i
5.  Next
6.  For i = 0 To 10
7.  Console.WriteLine(a(i))
8.  Next
```

فى هذه السطور :

- ق السطر رقم 1 تم الإعلان عن مصفوفة بالاسم a عدد عناصرها 10 من النوع Integer وق السطر رقم 2 تم الإعلان عن متغير بالاسم I من النوع Integer.
- ق السطر رقم 3 بداية التكرار For من القيمة 0 إلى القيمة 10.
- ق السطر رقم 4 يتم وضع قيمة المتغير I ق عنصر المصفوفة رقم I.
- فمثلا عندما تكون قيمة المتغير I هى 0 يتم وضع هذه القيمة ق العنصر رقم 0 ق المصفوفة أى a(0) و عندما تكون قيمة المتغير I هى 1 يتم وضع هذه القيمة ق العنصر رقم 1 ق المصفوفة أى a(1) وهكذا حتى تملئ المصفوفة بالقيم من 0 إلى 10.
- ق السطر رقم 6 يبدأ التكرار For مرة أخرى من القيمة 0 إلى القيمة 10.
- ق السطر رقم يتم استعمال الدالة ()Writeln لطباعة قيم المصفوفة a(I) مع تغيير قيمة I بالتكرار For.

تمرين (1) :

صمم برنامج يستقبل من المستخدم مجموعة من الحروف ثم يطبعها له بالعكس أي إذا استقبل الحروف a,b,c,d يطبعها لك d,c,b,a.

تمرين (2) :

صمم برنامج يستقبل مجموعة من الأسماء ثم يضبعها للمستخدم بعكس الترتيب.

فى لغة C#

يتم التعامل مع هذه المصفوفة بالخطوات التالية :

1- الإعلان عن المصفوفة.

2- استعمال عناصر المصفوفة بملئها بالقيم أو استعمال القيم أو طباعة القيم.

الإعلان عن المصفوفة

يتم الإعلان عن المصفوفة كما يتم الإعلان عن المتغيرات و بنفس درجات الإعلان بالصورة التالية :

```
int[] myArray = new int [5];
```

في هذه الصورة المتغير A هو أسم المصفوفة و العدد 10 هو عدد عناصر المصفوفة

والنوع int هو نوع عناصر المصفوفة و يمكن هذا النوع أن يتغير ليصبح من أي نوع (STRING,.....) حسب استعمال المصفوفة وفى الصورة الثانية تم الاعلان عن مصفوفة بالاسم Arr من النوع Integer ثم حجز هذه الأماكن فى الذاكرة باستعمال الكلمة New والإشارة إلى هذه الأماكن باسم المصفوفة.

ويمكن اعطاء قيم ابتدائية لعناصر المصفوفة كما فى الصور التالية:

```
int[] myArray = new int[] {1, 3, 5, 7, 9};

string[] weekDays = new string[]
                {"Sun","Sat","Mon","Tue","Wed","Thu","Fri"};

int[] myArray = {1, 3, 5, 7, 9};
string[] weekDays =
{"Sun","Sat","Mon","Tue","Wed","Thu","Fri"};

int[] myArray;
myArray = new int[] {1, 3, 5, 7, 9};   // OK
myArray = {1, 3, 5, 7, 9};   // Error
```

و لتوضيح ذلك تابع المثال التالي :

مثال :

1- قم بإنشاء تطبيق جديد ولكن من النوع ConsoleApplication كما سبق

2- داخل الدالة الرئيسية Main() اكتب السطور التالية:

```
public static void Main()
{
// Declare and initialize an array:
string[] WeekDays = new string []
{"Sun","Sat","Mon","Tue","Wed","Thu","Fri"};

// Pass the array as a parameter:
PrintArray(WeekDays);
}
```

فى هذه السطور:

تم الاعلان عن مصفوفة بالاسم WeekDays من النوع string .

قم بإنشاء دالة جديدة بالاسم PrintArray() كما فى السطور التالية :

```
static void PrintArray(string[] w)
{
for (int i = 0 ; i < w.Length ; i++)
Console. WriteLine (w[i] + "{0}", i < w.Length - 1 ? " "
: "");
Console.WriteLine();
}
}
```

في هذه السطور:

تم استعمال التكرار For مؤة أخرى من القيمة 0 إلى عدد عناصر المصفوفة.

تم استعمال الدالة ()Writeln لطباعة قيم المصفوفة a(i) مع تغيير قيمة I بالتكرار For.

وبالتالي عند تشغيل هذا البرنامج تحصل على قيم عناصر المصفوفة مطبوعة.

تمرين (1) :

صمم برنامج يستقبل من المستخدم مجموعة من الحروف ثم يطبعها له بالعكس أي إذا استقبل الحروف a,b,c,d يطبعها لك d,c,b,a.

تمرين (2) :

صمم برنامج يستقبل مجموعة من الأسماء ثم يطبعها للمستخدم بعكس الترتيب.

في لغة Java

لإنشاء مصفوفة نستعمل ثلاثة خطوات هي:

1- الإعلان عن متغير المصفوفة Array Variables .

2- تعريف (إنشاء) عنصر المصفوفة array object .

3- تخزين القيم أو البيانات داخل المصفوفة والتعامل معها .

1. الإعلان عن متغير مصفوفة Array Variable

ويتم ذلك بطريقتين هما:

الطريقة الأولى:

```
String ourteam [];
int Codes [];
```

فى هذه الطريقة يتم كتابة نوع عناصر المصفوفة فى الأول (String , int) ثم كتابة اسم المصفوفة [our team, codes] ثم وضع الأقواس [ ] وبالتالى يتم الإعلان عن مصفوفة بالاسم ourteam من النوع string والذى يفرق بين المصفوفة والمتغير العادى هو الأقواس [ ] التى تدل على أن اسم المتغير هو مصفوفة.

الطريقة الثانية:

```
String [ ] ourteam;
int [ ] codes;
```

فى هذه الطريقة يتم كتابة نوع المصفوفة ثم الأقواس بالشكل [ ] string ثم اسم المصفوفة وهى our team فى السطر الأول.

بالمثل المثال الثانى هو [ ] int ثم اسم المصفوفة وهو Codes.

وفى الحالتين تم الإعلان عن متغير من نوع هذه المصفوفة.

تعريف عنصر المصفوفة Array Object

بعد الإعلان عن متغير مصفوفة يتم تعريف (إنشاء) عنصر المصفوفة ويتم بطريقتين:

- استعمال كلمة new لإنشاء العنصر.
- تعريف عنصر object المصفوفة مع إعطائها قيم ابتدائية بدون استعمال new.

استعمال new

ويتم استعمال new لتعريف object عنصر كما يلى:

```
String jobs= new string [10]
```

وفى هذا السطر يتم تعريف متغير بالاسم jobs من نوع مصفوفات حرفيات [ ] string كما سبق حجز او تعريف مصفوفة عناصر جديدة باستعمال new فى الطرف الثانى من النوع string وعدد العناصر 10 وبالطبع يمكن إجراء هذا السطر على سطرين كما يلى:

```
String [ ] jobs;
jobs = new string [10];
```

وعند استعمال new مع المصفوفة لابد من تحديد عدد العناصر مثل 10.

تعريف عنصر المصفوفة وإعطائها قيم ابتدائية

يمكن تعريف عنصر مصفوفة بدون استعمال new وذلك بإعطاء المصفوفة قيم ابتدائية وبالتالى لا نحتاج لتحديد عدد العناصر لأننا نضع العناصر نفسها كما يلى:

```
String [ ] jobs = {"Eng", Doc', "Tech");
```

فى هذا السطر تم تعريف مصفوفة حرفيات بالاسم jobs ثم إعطائها قيم ابتدائية وهى الكلمات ...Eng وفى هذه الحالة لا نحتاج لتحديد عدد العناصر لأن العناصر موجودة بالفعل.

فى حالة استعمال New وعند إعطاء المصفوفة قيم ابتدائية يتم تسجيل قيم افتراضية حسب نوع المصفوفة فمثلاً مصفوفة القيم الرقمية تأخذ العناصر0 ومصفوفة الحرفيات تأخذ 0/ ومصفوفة القيم المنطقية Boolean تأخذ القيمFalse ومصفوفة العناصر Objects تأخذ null.

التعامل مع عناصر المصفوفة Accessing Array Elements

ويتم التعامل مع عناصر المصفوفة بكتابة رقم عنصر المصفوفة بين أقواس المصفوفة [ ]، كما بالشكل.

```
Array Name [ No]
```

وفى هذه الصورة Array Name اسم المصفوفة وno ترتيب العنصر فى المصفوفة مع ملاحظة أن ترتيب عناصر المصفوفة تبدأ من القيمة 0 ، وتقوم لغة Java باختبار قيمة ترتيب العنصر قبل التعامل معه للتأكد من وجود هذا العنصر ضمن عناصر المصفوفة وليس خارجها ولا يمكن تسجيل قيمة فى عنصر غير موجود ضمن عناصر المصفوفة (كما يحدث فى لغة C/C++) فلا يحدث خطأ.

ولتوضيح ذلك تابع المثال التالى:

```
String [ ] My Array = new String [20];
          My Array [20] = "Java lang";
```

فى السطر الأول يتم الإعلان وتعريف مصفوفة بالاسم My Array من نوع حرفيات string وحجز 20 عنصر باستعمال new.

فى السطر الثانى يتم تسجيل العبارة .Java lang فى المصفوفة My Array فى العنصر رقم 20 وهذا السطر عند ترجمة البرنامج يعطى خطأ وذلك لان هذه المصفوفة تبدأ من العنصر رقم 0 وتنتهى عند العنصر 19 وعدد العناصر 20 عنصر لذلك لا يوجد عنصر رقمه 20.

وللتعامل مع عدد عناصر المصفوفة يمكن استعمال الدالة LENGTH كما فى السطر التالى:

```
int L= MyArray. Length;
```

فى هذا السطر يتم حساب عدد عناصر المصفوفة باستعمال الدالة Length مع المصفوفة وحفظ هذه العدد فى المتغير L.

مثال

فى هذا المثال نحاول تجميع النقاط السابقة عن المصفوفة وتوضيحها كما فى السطور التالية:

```
1  public class Myfriends {
2
3      String [ ] Names = { "Khaled", "Alaa", "Amr", "Omr"};
4  void printNames ( )
5  {
6    int I = 1;
7    System.out.println (Names [I]);
8    I++;
9    System.out.println (Names [I]);
10   I++;
11 System.out.println (Names [I]);
12   I++;
13   System.out.println (Names [I]);
14   }
15   public static void main (String args [ ])
16   {
17
18   Myfriends FD= new Myfriends ( );
19   FD. printNames ();
20   }
21   }
```

فى هذه السطور:

- فى السطر رقم 1 يتم تعريف فصيلة البرنامج (Class) بالاسم Myfriends .
- فى السطر رقم 3 يتم تعريف مصفوفة حرفيات (string) بالاسم Names مع إعطائها قيم ابتدائية عبارة عن مجموعة أسماء.

- فى السطر رقم 4 يتم تعريف دالة عضوه دالة فى الفصيلة بالاسم ()printNames.
- فى السطر رقم 5 يتم الإعلان عن متغير i بقية ابتدائية 0.
- فى السطر رقم 6 يتم طباعة عنصر المصفوفة رقم i اى رقم 0 أى الاسم Khaled المخزن فى العنصر.
- فى السطر رقم 7 يتم زيادة قيمة المتغير i بمقدار 1 لتصبح i تساوى 1.
- فى السطر رقم 8 يتم طباعة عنصر المصفوفة رقم i أى رقم 1 أى Alaa وهكذا السطور حتى السطر رقم 12.
- وفى السطر رقم 13 تنتهى الدالة ()Print Names التى تقوم بطباعة الأسماء.
- فى السطر رقم 15 تبدأ الدالة الرئيسية ()main التى يبدأ التنفيذ منها وإذا كان هناك اكثر من فصلة class يتم تنفيذ الفصيلة التى بها الدالة main كما أشرنا من قبل.
- فى السطر رقم 18 يتم تعريف عنصر (object) بالاسم FD فى الفصيلة My friends وذلك لتتمكن من التعامل مع أعضاء الفصيلة My friends فى دوال وبيانات.
- فى السطر 19 تم استدعاء الدالة ()PrintNames مع هدف الفصيلة FD وبالتالى تقوم الدالة بطباعة الأسماء المعرفة داخل المصفوفة Names وتحصل على قائمة بالأسماء كما فى نتيجة التنفيذ.

استعمال أوامر التكرار for مع المصفوفة

لاحظنا فى المثال السابق كيف تم التعامل مع عناصر المصفوفة بتحديد ترتيب عنصر المصوفة كما فى السطر.

```
Names[3]
```

أى العنصر رقم 4 ، ولكن أشرنا إلى أنه يصعب أو يستحيل استعمال هذه الطريقة مع المصفوفات ولابد من استعمال جملة لتكرار for بدلا من ذلك.

وبالتالى يتم تعديل البرنامج السابق ليصبح كما فى السطور التالية:

```
1  public class MyFriends {
2
3      String [ ] Names = { "Khaled", "Alaa", "Am", "Omr"};
4  void printNames ( )
5    {
6     int i ;
7     for (i=0;i<4;i++)
8     System.out.println (Names [i]);
9       }
10   public static void main (String args [ ])
11    {
12
13    MyFriends FD= new MyFriends ( );
14    FD. printNames ();
15    }
16    }
```

الجديد فى هذه السطور عن برنامج المصفوفات التقليدى هو استعمال التكرار for Loop .

فى السطر رقم 4 للتكرار من 0 حتى Names. Length وهو عدد عناصر المصفوفة:

- فى السطر رقم 5 يتم طباعة عناصر المصفوفة Names[I]> أى العناصر المخزنة وبالتالى يتم طباعة الأسماء، وذلك يوفر إعادة كتابة أمر الطباعة وذلك باستعمال تكراره مع for بعدد عناصر المصفوفة.

وعند تنفيذ البرنامج تحصل على نفس النتيجة السابقة.

كما يتضح لنا أهمية استعمال جملة التكرار for مع المصفوفات .

نسخ المصفوفات

يمكن استعمال متغير مصفوفة للإشارة إلى مصفوفة أخرى كما فى الشكل.

```
double[] a = new double[3];
double[] b = a;
```

فى هذه السطور يتم الإعلان عن مصفوفة a ثم الإعلان عن مصفوفة b واستخدام متغيرها للإشارة إلى نفس المصفوفة a.

طول المصفوفة

```
for (int i = 0; i < a.length; i++)
    {
        b[i] = a[i];
       }
```

فى هذه السطور:

يتم الاستفادة من الخاصية length الموجودة فى المصفوفة لتحديد طولها أى عدد عناصرها بدلا من تحديد هذا الطول برقم ، ثم القيام بنسخ عناصر المصفوفة a إلى المصفوفة b.

ايجاد أكبر قيمة

يمكن استعمال المصفوفات لإيجاد أكبر قيمة من مجموعة قيم وذلك كما فى السطور التالية :

```
double max = A[0];
    for (int i = 1; i < A.length; i++) {
      if (A[i] > max)
        max = A[i];
    }
```

فى هذه السطور:

يتم وضع قيمة أول عنصر من المصفوفة فى المتغير mx بافتراض أنه أكبر عنصر .

ثم استعمال تركيب التكرار for فى المرور على جميع عناصر المصفوفة وفى كل مرة يتم مقارنة قيمة المتغير max بعنصر المصفوفة الحالى .

إذا كانت القيمة الموجودة فى المصفوفة أكبر من قيمة المتغير mx يتم التبديل بحيث ينتهى التكرار وقد تم وضع أكبر عنصر فى المتغير mx.

البحث عن قيمة داخل المصفوفة

من العمليات المشهورة للمصفوفات القيام بالبحث عن قيمة داخل مجموعة قيم.

والمثال التالى يوضح ذلك:

```
static int find(int[] A, int N)
    {

        for (int index = 0; index < A.length; index++)
        {
           if ( A[index] == N )
              return index;  // N has been found at this index!
        }

        return -1;
    }
```

في هذه السطور:

يتم إنشاء دالة تستقبل اسم المصفوفة والقيمة المطلوب البحث عنها ثم يتم استعمال تركيب التكرار for للمرور على العناصر ومقارنتها بالقيمة حتى نجدها.

فاذا وجدت يتم اعادة ترتيب هذه القيمة في المصفوفة وإلا يتم إعادة القيمة -1.

ترتيب عناصر المصفوفة

من العمليات المشهورة أيضا استعمال المصفوفات في ترتيب مجموعة قيم ولتوضيح ذلك تابع السطور التالية :

```
static void selectionSort(int[] A)
{
    for (int lastPlace = A.length-1; lastPlace > 0; lastPlace--)
    {
 position lastPlace.
        int maxLoc = 0;  // Location of largest item seen so far.
        for (int j = 1; j <= lastPlace; j++)
      {
        if (A[j] > A[maxLoc])
        {
             maxLoc = j;
        }
      }
```

```
        int temp = A[maxLoc];  // Swap largest item with A[lastPlace].
        A[maxLoc] = A[lastPlace];
        A[lastPlace] = temp;
    } // end of for loop

}
```

المصفوفة ذات البعدين :

فى VB.NET

يتم الإعلان عنها بالشكل التالي :

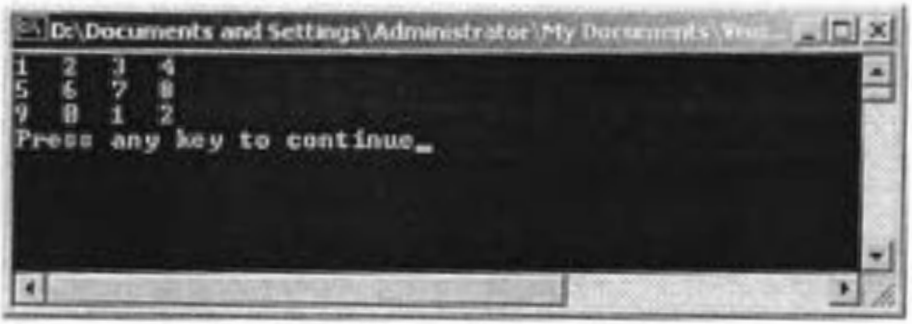
```
Dim RC(3, 4) As Integer
```

و فيها يتم الإعلان عن مصفوفة بالاسم RC بعدد صفوف 3 و عدد أعمدة 4 من النوع INTEGER أي الأرقام الصحيحة .

و بنفس القاعدة يمكن تغيير عدد العناصر حسب الطلب و كذلك تغيير نوع المصفوفة و لتوضيح ذلك تابع المثال التالي :

مثال :

عند تنفيذ هذا المثال تحصل علي نتيجة التنفيذ عبارة عن برنامج يطلب منك إدخال قيم عناصر مصفوفة 3*4 بعد إدخال العناصر يقوم البرنامج بطباعة العناصر بشكل مصفوفة كما بالشكل (2-6) .



الشكل (2-6)

جرب استعمال البرنامج ثم تابع معنا خطوات التصميم .

خطوات التصميم :

إبدأ برنامج جديد من النوع ConsoleApplication كما سبق .

- اكتب السطور التالية داخل الدالة الرئيسية (Sub Main():

```
Dim c, r
   Dim AR(3, 4) As Integer
   For r = 1 To 3
      For c = 1 To 4
         AR(r, c) = InputBox("ENTER NO:")
      Next c
   Next r
   For r = 1 To 3

      For c = 1 To 4
         Console.Write(AR(r, c))
         Console.Write("  ")
      Next c
      Console.WriteLine()

   Next r
```

في هذه السطور:

- يتم استعمال جملتي تكرار FOR لاستقبال عناصر المصفوفة وكذلك لطباعة صفوف وأعمدة المصفوفة كما يلي:

- في السطرين 1 يتم الاعلان عن مصفوفة بالاسم AR عدد عناصر 3*4 من النوع Integer.
- في السطر رقم 2 تبدأ التكرار الخارجي بلكمة For بالمتغير R.
- في السطر رقم 3 تبدأ التكرار الداخلي بالمتغير 2.
- في السطر رقم 4 يتم استقبال قيم المصفوفة من المستخدم باستعمال INPUTBOX مع وضع القيمة في عناصر المصفوفة AR (R,C) حسب قيمة R,C

الحالية. وبهذه يستمر استقبالنا عناصر المصفوفة حتى ينتهي التكرار الداخلي C والتكرار الخارجي R.

- من السطر 7 حتى السطر 12 يتم طباعة عناصر المصفوفة بشكل مصفوفة.
- وعند تنفيذ هذا البرنامج تحصل نتيجة التنفيذ كما في الشكل (3-6)

الشكل (3-6)

ملحوظة هامة:

يتصاحب استعمال المصفوفات Arrays باستعمال التكرار For لأنه بدون استعمال التكرار For يتم التعامل مع عناصر المصفوفة عنصر بعنصر ولذلك عند التعامل مع مصفوفة متعددة الابعاد يتم استعمال أكثر من For.

خصائص المصفوفات

بالاضافة لما شرحناها من طرق التعامل مع المصفوفات توجد بعض الخصائص للمصفوفات تفيد عند التعامل معها منها ما يلي:

الدالة GetUpperBound()

تقوم هذه الدالة بحساب ترتيب العنصر الأخير في المصفوفة ولتوضيح ذلك تابع المثال التالي:

مثال :

1- أبدأ برنامج جديد من النوع WindowsApplication كما سبق .

2- قم باضافة زر أمر Button وبداخل الحدث Button1_Click اكتب السطور التالية:

```
1:    Dim output As String
2:    Dim i As Integer
3:    Dim array1, array2 As Integer()
4:    array1 = New Integer() {10, 20, 30, 40, 50, _
   60, 70, 90, 70, 60, 50}
5:    array2 = New Integer(array1.GetUpperBound(0)) {}
6:    For i = 0 To array2.GetUpperBound(0)
7:    array2(i) = 2 + 2 * i
8:    Next
9:    output = "Index " & vbTab & "First " & vbTab & _
   "Secand" & vbCrLf
10:     For i = 0 To array1.GetUpperBound(0)
11:  output &= i & vbTab & array1(i) & vbTab & array2(i)&vbCrLf
12:     Next
13:       MessageBox.Show(output, " Values", _
14:       MessageBoxButtons.OK)
```

في هذه السطور:

- في السطر رقم 3 يتم الاعلان عن array1,array2 كمصفوفات من نوع القيم الصحيحة Integers.
- في السطر رقم 4 يتم إعطاء قيم لعناصر المصفوفة array1 مجموعة من القيم.
- في السطر رقم 6 يتم ابتداء التكرار for ابتداء من 0 إلى نهاية المصفوفة array2.
- GetUpperBound(0).
- في السطر رقم 7 يتم وضع قيم في المصفوفة array2 بمعلومية متغير التكرار I .
- في السطر رقم 9 يتم تكوين المتغير output بكلمة Index ثم مسافة ثم مسافة tab ثم كلمة First ثم مسافة كلمة Secand .
- في السطر رقم 10 يتم دمج عناصر المصفوفات array1,array2 مع ما سبق وذلك من خلال التكرار for.

في النهاية تم MessageBox.Show() لعرض النتائج.

نفذ البرنامج تحصل على النتيجة التالية كما في الشكل (4-6).

الشكل (4-6)

فى لغة C#

المصفوفة ذات البعدين :

يتم الإعلان عنها بالشكل التالي :

```
int[,] myArray = new int[4,2];
```

و فيها يتم الإعلان عن مصفوفة بالاسم myArray بعدد صفوف 4 و عدد أعمدة 2 من النوع int أي الأرقام الصحيحة .

و بنفس القاعدة يمكن تغيير عدد العناصر حسب الطلب و كذلك تغيير نوع المصفوفة و لتوضيح ذلك تابع الأمثلة التالية :

```
int[,] myArray = new int[4,2];
int[,,] myArray = new int [4,2,3];

int[,] myArray = new  int[,] {{1,2}, {3,4}, {5,6}, {7,8}};

int[,] myArray = {{1,2}, {3,4}, {5,6}, {7,8}};

int[,] myArray;

myArray = new int[,] {{1,2}, {3,4}, {5,6}, {7,8}};   // OK

myArray = {{1,2}, {3,4}, {5,6}, {7,8}};   // Error

myArray[2,1] = 25;
```

فى لغة Java

المصفوفات متعددة الأبعاد Multi dimensional Array

الإعلان عن مصفوفة متعددة الأبعاد.

ويتم ذلك باسم المصفوفة ونوعها وتحديد عدد الصفوف وعدد الأعمدة كما يلى:

```
int XY [ ] [ ] = New int [5] [6];
```

فى هذا السطر تم الإعلان عن مصفوفة بالاسم XY ذات بعدين من النوع int ثم استعمال new لتحديد عدد الصفوف بالعدد 5 والأعمدة بالعدد 6.

ويتم تسجيل أو التعامل مع أى عنصر بتحديد الصف والعمود فمثلاً لوضع قيمة فى العنصر الموجود فى الصف الثانى (2) والعمود الثالث (3) نكتب السطر التالى:

```
XY [2] [3] = 90;
```

وللتعامل مع المصفوفة ذات البعدين تستعمل اثنين تكرار متداخلة باستعمال for loop والذى يسمى Nested كما سيل فى فقرة التكرار Loops.

بلوك الأوامر Block Statements

كما يوجد فى لغة C ما يسمى بلوك كذلك يوجد فى لغة Java وهو عبارة عن أى مجموعه من الجمل أو الأوامر محصورة بين الأقواس { } كما بالشكل التالى:

```
{
    :
    :
    :
Statement 1
Statement 2
    :
    :
}
```

فى هذا الشكل يتم حصر جمل Statement 1 و Statement 2 بين الأقواس ليتعامل كبلوك.

ولكن السؤال ما الفائدة وراء إنشاء بلوك جمل ؟

الفائدة هى إمكانية الإعلان عن متغيرات داخل البلوك ليس لها علاقة بما قبل او بعد البلوك مثل تعريف متغير داخل بلوك داخل دالة فهو خاص بها.

مثال :

فى هذا المثال نوضح كيفية استعمال التكرار المتداخل مع المصفوفات وكيفية التعامل مع عناصر المصفوفات بدقة اكتب سطور هذا البرنامج كما فى الشكل.

```
1  import java.io.*;
2  public class MyFriends {
3
4      public static void main (String args [ ]) throws IOException
5
6      {
7        int x,y;
8          InputStreamReader inStream = new InputStreamReader( System.in );
9        BufferedReader stdin = new BufferedReader( inStream );
10
11       String inData;
12   char  [ ][ ]  arr=new char [40][40];
13     int r=0,c=0;
14
15     for (r=0;r<40;r++)
16     for (c=0;c<40;c++)
17     arr[r][c]=' ';
18   do
19   {
20
21   for (r=0;r<40;r++)
22   {
23     System.out.println ("");
24
25     for (c=0;c<40;c++)
26     System.out.print (arr[r][c]);
27     }
28        System.out.print ("X=");
29        System.out.print("Enter the u r Name :");
30     inData = stdin.readLine();
31     x = Integer.parseInt( inData );
32
33        System.out.print(" Y=");
34        inData = stdin.readLine();
35      y = Integer.parseInt( inData );
36      arr[x][y]='X';
37    }while (x>=0);
38    }
39 }
```

فى هذه السطور:

فى السطر رقم 12 تم الاعلان عن مصفوفة من نوع حروف عدد صفوفها وأعمدتها 40

فى السطور 15و16و17 يتم ملئ عناصر هذه المصفوفة بالحرف . وذلك باستخدام التكرار المتداخل مع المصفوفة.

فى السطر رقم 18 يبدأ تكرار do.

فى السطور من 21 إلى 27 يتم طباعة عناصر المصفوفة وهى طباعة 40x40 من النقط مما يعطى شكل ورقة رسم بيانى.

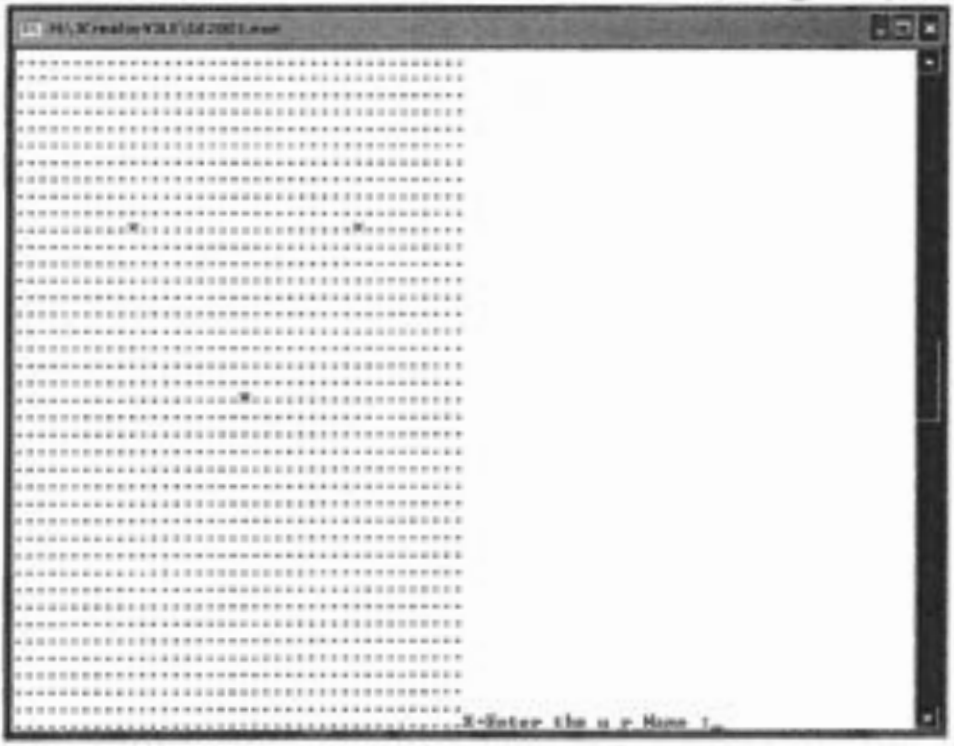
فى السطر رقم 30 يتم استقبال قيمة x.

فى السطر رقم 35 يتم استقبال قيمة y.

فى السطر رقم 36 يتم وضع الحرف x فى عنصر المصفوفة x,y .

ثم يعاد طباعة المصفوفة بالشكل الجديد مما يؤدى إلى طباعة ورقة رسم بيانى موقع عليها هذه النقطة، ويستمر ذلك حتى يدخل المستخدم قيمة سالبة.

نفذ البرنامج لتحصل على برنامج رسم بيانى يستقبل منك النقطة (r,c) المطلوب توقيعها ثم يقوم البرنامج برسم علامة x فيها وهكذا حتى تدخل قيمة سالبة كما فى الشكل.



الفصل السابع

مفاهيم البرمجة بواسطة الأهداف OOP

فى هذا الفصل Programming Concepts

7

تعتبر مفاهيم البرمجة بواسطة الأهداف OOP من أهم مفاهيم البرمجة الحديثة حيث أصبحت معظم لغات البرمجة تطبقه وبناء الفصائل classes هو أساس البرمجة بواسطة الأهداف OOP وفى هذا الفصل نشرح الخطوات العملية لبناء الفصائل وذلك من خلال النقاط التالية :

- مفاهيم البرمجة بواسطة الأهداف OOP
- انشـاء واستعمال فصائل Classes
- أضافة دوال أعضاء adding methods to class
- فصائل بدون دوال بناء classes without constructors
- دوال البناء في الفصيلة constructors
- استعمال أداة designer في التعامل مع الفصائل classes
- إنشاء خصائص للفصيلة properties
- إضافة خصائص قراءة أو كتابة فقط read-only and write- only

مفاهيم البرمجة بواسطة الأهداف Object Oriented Programming

تعتبر البرمجة بواسطة الأهداف (object oriented programming)

من الاتجاهات الحديثة للبرمجة ومن أول اللغات التي طبقت هذا المفهوم لغة c++ وكذلك من اللغات التي تلتزم بمفهوم OOP لغة JAVA ولغة VB.NET زادت من اهتمامها بمفهوم OOP بل وتعتبر ذلك من التعديلات المهمة والتي زادت من قوة VB.NET لذلك قبل ان نشرح كيفية تحقيق ذلك في VB.NET نشرح أولاً مفاهيم البرمجة بواسطة الأهداف (OOP)بشكل مختصر لذلك إذا كنت على علم بها انتقل إلي الفقرة التالية وهي استعمال VB.NET في إنشاء الفصائل والا تابع معنا شرح هذه الفقرة حيث نتناول النقاط التالية :

- معنى البرمجة بواسطة الأهداف Object Orient Programming
- معنى الفصيلة class
- ما هي دالة البناء و دوال الهدم constructors & destructors
- ما هي خاصية التوريث inheritence
- ما هي خاصية over loading

معنى البرمجة بواسطة الأهداف OOP(Object oriented programming

تبني فكرة البرمجة بواسطة الأهداف (OOP) على استعمال الهدف (Object) كوحدة برمجة بمعنى انه بدلاً من استعمال الدوال والأوامر لبناء البرنامج مما يضطر المبرمج لإعادة كتابة الأوامر كل مرة لتحقيق فكرة معينة وهذه كانت فكرة البرمجة التقليدية .

ولكن أتت البرمجة بواسطة الأهداف (OOP)لتجعل وحدة بناء البرمجة كبيرة وهي هدف Object أو فصيلة class وبالتالي يتم إعداد مجموعة من الفصائل العامة classes التي تلبي معظم متطلبات إعداد برنامج والتي يحتاجها المبرمج حتى ان بعض المبرمجين يشبه البرمجة بواسطة الأهداف بالبناء باستعمال المباني الجاهزة والبرمجة بالطريقة التقليدية القديمة تشابه البناء باستعمال الأدوات الأولية وبالتالي الفرق بينهما في السرعة كبيرة جدًا.

معنى الفصيلة class

الفصيلة class هي أساس البرمجة بواسطة الأهداف (OOP)وهي التي يبنى عليها

البرنامج وأخذت فكرة الفصيلة CLASS من الواقع فكل عنصر من عناصر الحياة عبارة عن فصيلة class ،فأنت تستطيع أن تطلق علي جميع السيارات إنها من فصيلة CAR مع بعض الاختلافات ،ويمكنك ان تطلق على الطيور فصيلة Birdأى طائر وهكذا تنتمى جميع العناصر الى فصائل classes ،وكل فصيلة تستطيع تمثيلها بعنصرين هما البيانات والدوال (methods , data)، فمثلاً فصيلة الموظف Employee بياناتها هي بيانات الموظف العامة مثل كود الموظف ، اسم الموظف ، عنوان الموظف ،تليفون الموظف ، وباقي بياناته ، وكذلك الدوال (methods) هي دوال تحقيق العمليات التي يمكن أن تتم على الموظف مثل :عملية إضافة موظف جديد ، وحذف موظف موجود ، تعديل بيانات موظف وجميع العناصر يمكن تمثيلها بهذه الطريقة .

إذن الفصيلة هي مجموعة من السطور التي تمثل عنصراً تمثيل تاماً من حيث بيانات العنصر والتي تسمى خصائص proprties، وكذلك دوال العنصر التي تسمى methods وبتقسيم البرنامج إلي فصائل يصبح أكثر نظاماً وأسرع في الإعداد ، حيث قامت ميكروسوفت في منتجها vb.net بإعداد مجموعة كبيرة من الفصائل التي تلبي متطلبات المبرمج،وما عليك عند إعداد البرامج إلا أن تدرس مكتبة الفصائل class Lib الموجودة لتعرف المتوفر منها، وكذلك تتعرف على بيانات (خصائص) proprtie ودوال الفصيلة Methods وما شاهدناه في الفصول الأولي من الأوامر التي يكتبها vb.net عند إنشائك لبرنامج جديد هو استعمال فصائل كل عنصر تستعمله ، فمثلاً عند إضافتك لنموذج (form) تلاحظ انه يستورث الفصيلة form وبالمثل عند إضافتك لزر الأمر button1 يتم استوراث الفصيلة button وهكذا .

وهذا الأسلوب لم يكن واضحاً في الإصدار السابق vb6 حيث كنت تستعمل ذلك بدون الشعور بالفصائل ولا التعامل معها ولا التعديل فيها ولكن أتاح لك vb.net التعامل مع الفصائل للاستفادة من مفهوم OOP .

والسؤال :كيف أستطيع إنشاء فصيلة CLASS والتعامل معها وتطبيق مفاهيم OOP ؟

الإجابة: هذا ما سوف نوضحه في هذا الفصل بعد الانتهاء.

أولاً : توضيح المفاهيم الأساسية لمفهوم OOP

دوال تنفذ عند البداية Construction

دوال البناء (constructors): هي دوال تنفذ تلقائياً عند استعمالك للفصيلة class.
(عند تعريف عنصر object) وهي تشبه حدث form-load الموجود في الـ form ،تستعمل في تسجيل أي قيم ابتدائية أو تعريف أي متغيرات أو أي شروط ابتدائية.

دوال تنفذ عند النهاية destructors : هي دالة أو دوال تنفذ تلقائياً عند الانتهاء من استعمال الفصيلة class ويصبح هدف الفصيلة يشير إلى nothing .
وهي تشبه الحدث form-unload الموجود في الform وتستعمل للإنهاء أو التخلص من تعريف متغيرات أو إجراء أي عمليات قبل الخروج من البرنامج.

ما هي خاصية التوريث inheritance

معنى خاصية التوريث (inheritance) هو توريث فصيلة Base class قديمة موجودة بالفعل لفصيلة class جديدة عند إنشائها بحيث يضاف تركيب الفصيلة Base class القديمة من بيانات و دوال إلى الفصيلة الجديدة new class ثم تكمل عليها في الفصيلة الجديدة

وهذه الخاصية من أهم خصائص مفهوم OOP فعلى أساسه تبنى مكتبات الفصائل Class Lib حيث يتم بناء فصيلة أساس Base class ثم تستورثها الفصيلة الثانية والثالثة وتأخذ الفصائل من بعضها البعض حتى تكون مكتبة فصائل عبارة عن شجرة فصائل Class Tree وهذا هو الحال في مكتبة الفصائل الشهيرة MFC الخاصة بميكروسوفت في لغة Visual C++ وكذلك مكتبة فصائل لغة Java المعروفة بالاسم JFC.

ما هي الخاصية Overloading

معنى خاصية **overloading** : هو إمكانية إنشاء اكثر من دالة Method بنفس الاسم مع تغيير عدد المعاملات Parameters مثلاً ، وهذا يفيد بإنشاء أكثر من دالة لنفس الوظيفة

بمعاملات مختلفة بنفس الاسم ، مثل إنشاء دالتين بالاسم ( )find cust إحداهما تأخذ معامل رقمي هو كود الموظف ، والثانية تأخذ عبارة حرفية هي اسم الموظف وفي الحالتين تبحث الدالة ، و بالتالي يشعر المبرمج كأنها دالة واحدة ولكنها دالتين يتم استدعاء كل واحدة تلقائياً حسب المعامل المرسل لها .

فى VB.NET

إنشاء واستعمال الفصائل creating & Use class

بعد شرح بعض المفاهيم المهمة لمفهوم OOP تعال معنا نشرح عملياً كيفية إنشاء الفصائل CLASSES وتحقيق هذه المفاهيم :

الفصيلة class هي أساس مفهوم OOP لذلك أول ما تعلمه فى هذا المفهوم هو كيفية إنشاء واستعمال المصفوفة class ولتحقيق ذلك تابع معى الخطوات التالية:

1- قم بإنشاء تطبيق جديد من النوع ConsoleApplication بالخطوات المعتادة كما سبق.

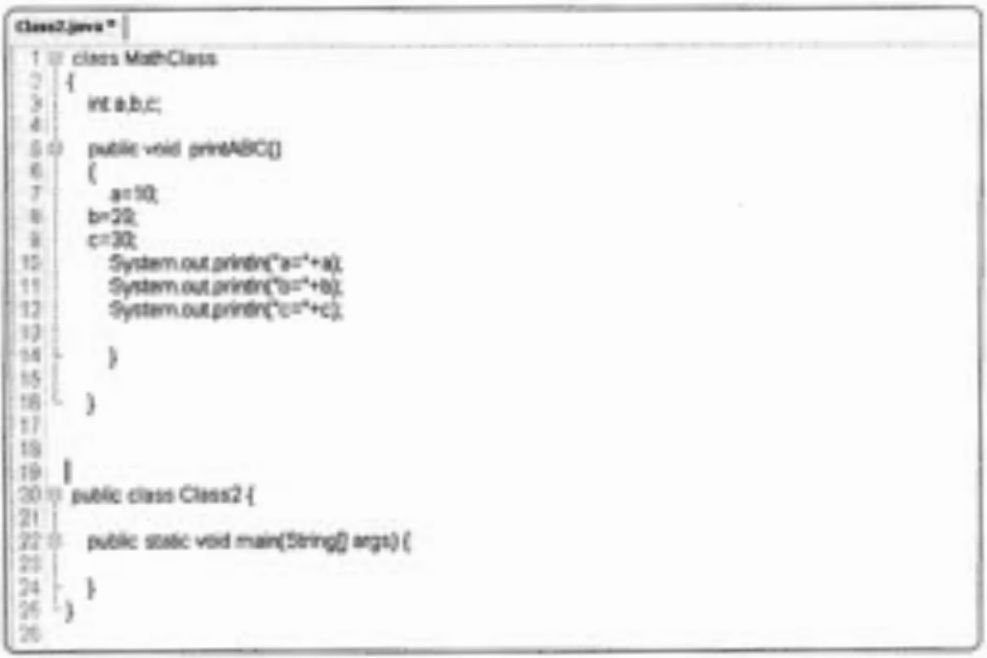
2- أسفل الدالة الرئيسية اكتب سطور إنشاء فصيلة جديدة بالاسم class1 كما فى الشكل (7-1).

الشكل (7-1)

فى لغة C#

```
using System;
namespace ConsoleApplication3
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {

        }
    }
    public class MyClass
    {
    public  int a,b,c;
    public  void printabc()
        {
            a=10;
            b=20;
            c=30;
            Console.WriteLine ("a="+a);
            Console.WriteLine ("b="+b);
            Console.WriteLine ("c="+c);

        }

    }
}
```

فى لغة Java



```
class MathClass
{
  int a,b,c;

  public void printABC()
  {
    a=10;
  b=20;
  c=30;
    System.out.println("a="+a);
    System.out.println("b="+b);
    System.out.println("c="+c);

    }

  }


public class Class2 {

  public static void main(String[] args) {

  }
}
```

1- في هذا الشكل في السطر رقم 7 تم إنشاء فصيلة بالاسم class1 وذلك باستعمال الكلمة المحجوزة class مع وضع كلمة publi قبلها حتى يمكن التعامل مع الفصيلة class1.

2- في السطر رقم 8 تم الاعلان عن المتغيرات a,b,c من النوع Integer مع وضع كلمةpublic قبلها حتى يمكن التعامل مع هذه المتغيرات مباشرة من خارج الفصيلة class1 وسوف يتم توضيح هذه النقطة.

3- في السطر رقم 9 تم إنشاء Method من النوع Sub إجراء بالاسم ()Printabc.

- مع وضع كلمة public قبلها حتى يمكن التعامل مع هذا الاجراء مباشرة من خارج الفصيلة class1 وسوف يتم توضيح هذه النقطة.
- وداخل سطور هذا الاجراء تم استعمال الدالة ()WriteIn ثلاث مرات لطباعة قيم المتغيرات a,b,c كما في السطور 10,11,12.
- بهذا تم إنشاء فصيلة جديدة Class بالاسم class1 مع تعريف متغيرات data (a,b,c) ودالة ()Printabc كأعضاء لهذة الفصيلة class1.
- من فضلك قم بمراجعة سطور إنشاء الفصيلة class1 جيدا وحاول استيعاب كيفية إنشاء وكتابة سطور الفصيلة class.

والخطوة التالية هي كيفية استعمال الفصيلة الجديدة class1 لتوضيح ذلك تابع الخطوات التالية:

1- عد الى البرامج وداخل سطور الدالة الرئيسية()Sub Main اكتب سطور استعمال الفصيلة class1 كما في الشكل (2-7).

الشكل (2-7)

فى لغة C#

```
using System;
namespace ConsoleApplication1
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
Myclass MyObj=new Myclass();
MyObj.printabc ();
        }
    }
    public class Myclass
    {
    public  int a,b,c;
    public  void printabc()
        {
            a=10;
            b=20;
            c=30;
            Console.WriteLine ("a="+a);
            Console.WriteLine ("b="+b);
            Console.WriteLine ("c="+c);

        }

    }
}
```

فى لغة Java

```
Class2.java *
 1  class MathClass
 2  {
 3     int a,b,c;
 4
 5     public void printABC()
 6     {
 7        a=10;
 8     b=20;
 9     c=30;
10        System.out.println("a="+a);
11        System.out.println("b="+b);
12        System.out.println("c="+c);
13
14        }
15
16     }
17
18
19  public class Class2 {
20
21     public static void main(String[] args) {
22        MathClass obj1=new MathClass();
23        obj1.printABC();
24
25     }
26  }
```

فى هذا الشكل تم استعمال المصفوفة كما يلى:

- فى السطر رقم 4 تم الاعلان عن متغير بالاسم nc من نوع الفصيلة class1 فهو فى هذه الحالة لا يسمى متغير بل يسمى هدف Object مع استعمال الكلمة new لإنشاء هدف object فعلى بالذاكرة
- وبالتالى تكون أول خطوة لاستعمال الفصيلة class هى تعريف هدف object من الفصيلةclass كما فى السطر التالى:

```
Dim nc As New class1()
```

- فى السطر رقم 5 تم وضع القيمة 20 فى المتغير a التابع للفصلة class1 ولكن مع الهدف nc.
- بالمثل السطر 6 و7 تم وضع قيم فى المتغيرات.

- وبالتالى أى تعامل مع الفصيلة لا يتم مع اسمها بل يتم مع اسم الهدف المعرف منها مع ملاحظة امكانية تعريف أكثر من هدف وبالتالى تغيير القيم فى كل هدف .
- فى السطر رقم 8 تم استدعاء الدالة Printabc() ولكن مع اسم الهدف من الفصيلة وهو nc حيث لا يصلح التعامل مع أى عضو داخل الفصيلة الا عن طريق متغير الهدف المعرف من الفصيلة.

3- قم بتنفيذ البرنامج تلاحظ استدعاء السطور المكتوبة داخل الدالة الرئيسية وهى تعريف هدف من الفصيلة واعطاء قيم لمتغيرات الفصيلة ثم طباعة هذه القيم باستدعاء الدالة Printabc() كما فى الشكل (3-7) .

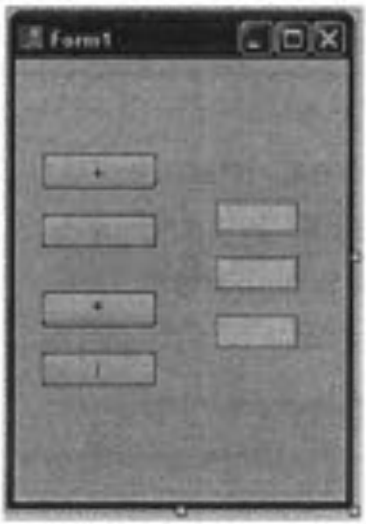
الشكل (3-7)

صديقى حاول التدرب كثيرا على أعداد الفصائل classes بما تحتويه من بيانات Data Members ودوال اعضاء Methods وكذلك التدريب على استعمال هذه الفصائل classes بتعريف متغير object ثم استدعاء الدوال الاعضاء Methods

ولزيادة التوضيح تابع معنا المثال التالى:

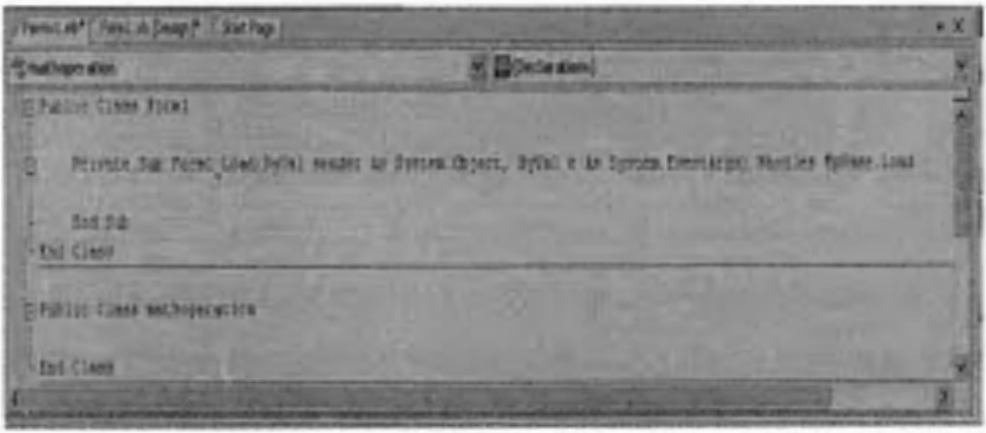
1- قم بإنشاء تطبيق جديد من النوع Windows Application بالخطوات المعتادة كما سبق

2- قم بتصميم برنامج العمليات الحسابية الذى تم اعداده فى الفصل الأول كما فى الشكل (4-7) .



الشكل (4-7)

3- اضغط بالماوس مرتين فوق خلفية البرنامج Form1 للحصول على صفحة الأوامر في نهاية الصفحة قم بكتابة سطور إنشاء فصيلة class جديدة كما في الشكل (5-7) .



الشكل (5-7)

في لغة C#

```
150         static void Main()
151         {
152             Application.Run(new Form1());
153         }
154
155         private void Form1_Load(object sender, System.EventArgs e)
156         {
157
158         }
159     }
160
161     public class mathoperations
162     {
163
164     }
165
166 }
167
```

فى لغة Java

Mathoperations.java *

```
1  class operations
2  {
3
4  }
5
6
7  public class Mathoperations {
8
9      public static void main(String[] args) {
10
11     }
12 }
```

- فى هذا الشكل فى السطر رقم 122 تم إنشاء فصيلة جديدة بالاسم mathoperations وذلك بالاستعمال الكلمة المحجوزة class وتلقائيا تم إغلاق هذه الفصيلة بالعبارة End class.

4- داخل سطور الفصيلة الجديدة mathoperations قم بتعريف متغيرات جديدة كما فى السطر التالى Public a, b, c As Double.

5- داخل سطور الفصيلة الجديدة mathoperations قم بإنشاء دوال العمليات الحسابية فقم بتعريف دالة لعملية الجمع واخرى لعملية الطرح وعملية الضرب وعملية القسمة ، لتصبح الفصيلة الجديدة mathoperations كما فى السطور التالية:

```
Public Class mathoperations
    Public a, b, c As Double
    Public Function sum(ByVal v1 As Double, ByVal v2 As Double)
        Dim res As Double
        res = v1 + v2
        Return res
    End Function

    Public Function subs(ByVal v1 As Double, ByVal v2 As Double)
        Dim res As Double
        res = v1 - v2
        Return res
    End Function
```

```
    Public Function mul(ByVal v1 As Double, ByVal v2 As Double)
        Dim res As Double
        res = v1 * v2
        Return res
    End Function

    Public Function divs(ByVal v1 As Double, ByVal v2 As Double)
        Dim res As Double
        res = v1 / v2
        Return res
    End Function

End Class
```

فى لغة C#

```
    public class mathoperations
    {
    public double a, b, c;
  public double sum( double v1 ,  double v2  )
      {
    double res;
    res = v1 + v2;
        return res;
        }
public double sub( double v1 ,  double v2  )
      {
    double res;
    res = v1 - v2;
        return res;
        }

        public double mul( double v1 ,  double v2  )
        {
            double res;
            res = v1 * v2;
            return res;
        }

public double div( double v1 ,  double v2  )
      {
    double res;
    res = v1 / v2;
        return res;
        }

    }
```

فى لغة Java

```
class operations
{
  public double a,b,c;

  public double sum(double v1,double v2)
  {
    double res;
    res=v1+v2;
    return res;
      }
  public double sub(double v1,double v2)
  {
    double res;
    res=v1-v2;
    return res;
      }
  public double mul(double v1,double v2)
  {
    double res;
    res=v1*v2;
    return res;
      }
   public double div(double v1,double v2)
  {
    double res;
    res=v1/v2;
    return res;
      }
    }


public class Mathoperations {

  public static void main(String[] args) {

  }
}
```

راجع هذه السطور تلاحظ أننا أنشأنا فصيلة جديدة وبها مجموعة دوال العمليات الحسابية التي أشرنا إليها.

6- قم باستعمال الفصية الجديدة كما يلي:

- اضغط زر أمر الجمع واكتب به السطور التالية:

```
1. Dim mathobj As New mathoperations()
2. Dim a, b, c As Double
3. a = Val(TextBox1.Text)
4. b = Val(TextBox2.Text)
5. c = mathobj.sum(a, b)
6. TextBox3.Text = c
```

فى لغة C#

```
161 private void button1_Click(object sender, System.EventArgs e)
162         {
163 mathoperations mathobj=new mathoperations ();
164 double a, b, c ;
165 a = double.Parse (textBox1.Text);
166 b = double.Parse(textBox2.Text);
167 c = mathobj.sum(a, b);
168 textBox3.Text = c+"";
169         }
```

ق هذه السطور:

- ق السطر رقم 1 تم تعريف هدف object بالاسم mathobj من الفصيلة الجديدة mathoperations وذلك باستعمال كلمتى dim وNew كما سبق وشرحنا.
- ق السطر رقم 2 تم الاعلان عن ثلاثة متغيرات a,b,c من النوع Double.
- ق السطر رقم 3 و4 تم وضع قيمة محتوى مربعى النص TextBox1,TextBox2 ق المتغيرين a,b وذلك باستعمال الدالة ()val.
- ق السطر رقم 5 تم استدعاء الدالة ()sum مع هدف الفصيلة mathobj مع إرسال معاملين هما a,b لجمع المتغيرين a,b ووضع النتيجة ق المتغير c.
- ق السطر رقم 6 تم وضع قيمة المتغير c وهو نتيجة الجمع ق مربع النص TextBox3 بهذا تم استعمال الدالة ()sum المعرفة داخل الفصيلة mathoperations
- اضغط زر أمر الطرح واكتب به السطور التالية:

```
7.  Dim mathobj As New mathoperations()
8.  Dim a, b, c As Double
9.  a = Val(TextBox1.Text)
10. b = Val(TextBox2.Text)
11. c = mathobj.subs(a, b)
12. TextBox3.Text = c
13.
```

فى لغة C#

```
161 private void button1_Click(object sender, System.EventArgs e)
162         {
163 mathoperations mathobj=new mathoperations ();
164 double a, b, c ;
165 a = double.Parse (textBox1.Text);
166 b = double.Parse(textBox2.Text);
167 c = mathobj.sub(a, b);
168 textBox3.Text = c+"";
169         }
```

فى هذه السطور:

تلاحظ أنها نفس السطور السابقة مع استدعاء دالة الطرح.

- اضغط زر أمر الضرب واكتب به السطور التالية:

```
14. Dim mathobj As New mathoperations()
15. Dim a, b, c As Double
16. a = Val(TextBox1.Text)
17. b = Val(TextBox2.Text)
18. c = mathobj.mul(a, b)
19. TextBox3.Text = c
```

فى لغة C#

```
161 private void button1_Click(object sender, System.EventArgs e)
162         {
163 mathoperations mathobj=new mathoperations ();
164 double a, b, c ;
165 a = double.Parse (textBox1.Text);
166 b = double.Parse(textBox2.Text);
167 c = mathobj.mul(a, b);
168 textBox3.Text = c+"";
169         }
```

فى هذه السطور:

تلاحظ أنها نفس السطور مع استدعاء دالة الضرب.

- اضغط زر أمر القسمة واكتب به السطور التالية:

```
20. Dim mathobj As New mathoperations()
21. Dim a, b, c As Double
22. a = Val(TextBox1.Text)
23. b = Val(TextBox2.Text)
24. c = mathobj.divs(a, b)
25. TextBox3.Text = c
```

فى لغة C#

```
161 private void button1_Click(object sender, System.EventArgs e)
162         {
163 mathoperations mathobj=new mathoperations ();
164 double a, b, c ;
165 a = double.Parse (textBox1.Text);
166 b = double.Parse(textBox2.Text);
167 c = mathobj. div (a, b);
168 textBox3.Text = c+"";
169         }
```

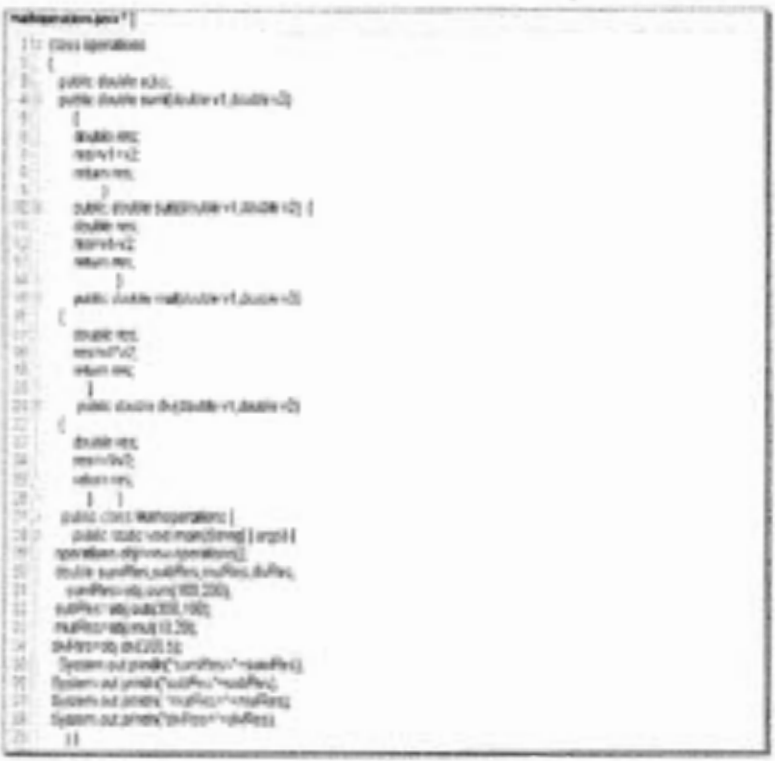
فى هذه السطور:

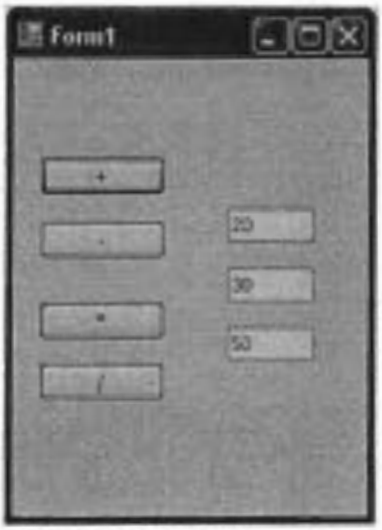
تلاحظ أنها نفس السطور مع استدعاء دالة القسمة.

7- قم بتشغيل البرنامج لتحصل على نتيجة التنفيذ جرب واكتب رقمين واضغط زر أمر الجمع و الطرح وغيره ولاحظ النتيجة كما فى الشكل (6-7).

فى لغة Java

جميع العمليات كما فى السطور.





الشكل (6-7)

الجديد فى هذا البرنامج:

لم يكن الغرض من هذا البرنامج هو تحقيق عمليات الجمع والطرح والضرب وإلا فقد حققناها فى أول الكتاب ببساطة وبدون استعمال فصائل وإنما الغرض من البرنامج هوتوضيح كيفية إنشاء فصيلة واضافة بيانات اعضاء ودوال أعضاء لها فأرجو أن تكون الصورة واضحة وكما لفت نظرك من قبل حاول التدرب على هذا الأسلوب الجديد وهو إنشاء الفصائل وإضافة دوال وبيانات استعمال هذة الفصائل.

اضافة فصيلة class على مستوى التطبيق:

فى المثال السابق تم إنشاء الفصيلة class داخل Form1 ولكن هناك أكثر من طريقة لإنشاء الفصيلة class على مستوى التطبيق كما يلى:

(1) اضافة ملف Module

يمكن اضافة ملف Module ثم إنشاء الفصيلة class داخلها وبالتالى تصبح هذه الفصيلةclass معرفة على مستوى التطبيق ويمكن اضافة ملف Module لاختيار Project من القائمة الرئيسية ثم اختيار Add Module كما سبق فتحصل على صفحة بيضاء يمكنك

الكتابة بها ويمكنك كتابة الفصيلة الجديدة داخلها كما فى الشكل (7-7).

```
Module Module1

    Public Class Ourclass

    End Class

End Module
```

الشكل (7-7)

(2) إضافة ملف فصيلة مباشرة

يمكن اضافة فصيلة class باضافة ملف للفصيلة وذلك باختيار Add Class من الاختيار Project من القائمة الرئيسية فتحصل على مربع حوار كما فى الشكل (7-8) .

الشكل (7-8)

وبعد تحديد اسم الفصيلة class والضغط على الزر Open تحصل على الفصيلة class الجديدة كما فى الشكل (7-9) .

```
Public Class Class1


End Class
```

الشكل (7-9)

فى هذا الشكل يمكنك كتابة محتويات الفصيلة class1 من بيانات Data ودوال أعضاء Methods.

ملحوظة:

لا تستخدم الكلمة MyClass كأسم للفصيلة class لانها كلمة محجوزة من كلمات Visual Basic.Net التى لها معنى خاص

دوال Constructors

من الخصائص المتوفرة فى مفاهيم البرمجة بالاهداف OOP وتدعمها Visual Basic.Net فكرة وجود دالة البناء Constructor داخل الفصيلة class وهى دالة Method مثل أى دالة ولكن الفرق فى أنها دالة تنفذ تلقائيا بدون استدعاء أى بمجرد تعريف هدف object من الفصيلة class ، والغرض من ذلك هو استعمال هذه الدالة فى تنفيذ أى عمليات أولية للفصيلة class مثل اعطاء قيم ابتدائية للمتغيرات.

وفى لغة Visual Basic.Net يتم تحديد دالة البناء Constructor داخل الفصلة بالاسم New أى تصبح بالصورة التالية:

```
Sub New()
End Sub
```

فى لغة C#

```
 8    public class ExtClass1
 9    {
10        public int v1,v2,v3;
11
12        public ExtClass1()
13        {
14            //
15            // TODO: Add constructor logic here
16            //
17
18        }
19    }
```

فى لغة Java

```
Methoperators.java  Constructor.java*
1  public class Constructor {
2      Constructor ()
3      {
4        System.out.println("this is the Constructor >>> it run without calling ");
5        }
6
7        public static void main(String[] args) {
8        Constructor obj=new Constructor();
9          }
10 }
11
```

ولتوضيح ذلك تابع الخطوات التالية:

1- قم باضافة فصيلة class بالاسم class1 كما شرحنا فى الفقرة السابقة.

2- داخل سطور الفصيلة الجديدة class1 قم بإنشاء دالة البناء Constructor بالاسم Sub New() كما فى الشكل (10-7).

```
Public Class Class1

    Public Sub New()
        MsgBox("This is My constructor")
    End Sub

End Class
```

الشكل (10-7)

فى لغة C#

```
using System;
using System.Windows.Forms;
namespace WindowsApplication11
{
    /// <summary>
    /// Summary description for constclass1.
    /// </summary>
    public class constclass1
    {
        public constclass1()
        {
            MessageBox.Show (" this message displayed without call");
        }
    }
}
```

فى لغة Java

```
Methodoperations.java  Constructor.java*
public class Constructor {
    Constructor()
    {
      System.out.println("this is the Constructor >>> it run without calling ");
    }

    public static void main(String[] args) {
    Constructor obj=new Constructor();
      }
}
```

فى هذا الشكل (10-7) تم إنشاء دالة Sub جديدة بالاسم ()New وهى تحتوى على سطر واحد هو عرض رسالة باستعمال الدالة ()MsgBox.

3-قم بتوقيع زر أمر Button1 فوق خلفية البرنامج Form وغير عنوانه خاصية Text إلى Test Constructor.

4- اضغط زر الأمر Button1 مرتين واكتب به السطر التالى :.

```
Dim ConsObj As New Class1()
```

فى لغة C#

```
179        private void button3_Click(object sender, System.EventArgs e)
180        {
181            constclass1 obj1 =new constclass1 ();
182
183        }
```

- فى هذا السطر تم تعريف هدف object من الفصيلة class1 بالاسم ConsObj ولم يتم استدعاء أى دوال.

5- نفذ البرنامج F5 واضغط على زر الأمر تلاحظ ظهور رسالة دالة البناء Costructor بالرغم من عدم استدعائه ولكن تم الاستدعاء تلقائيا وتحصل على النتيجة كما فى الشكل (11-7).

الشكل (11-7)

إنشاء أكثر من دالة باسم واحد Method Overloading

من المفاهيم المشهورة فى البرمجة بالأهداف OOP مفهوم إنشاء أكثر من دالة باسم واحد ويطلق عليه Method Overloading حيث يسمح هذا المفهوم بإنشاء أكثر من دالة بنفس الاسم اذا دعت الحاجة لذلك مثل إنشاء مجموعة دوال رسم بالاسم draw() تستطيع رسم خط Line ومربع Box ودائرة circle وغيره ولكن يتوقف ذلك على معاملات الدالة.

ومثال أخر يمكن إنشاء أكثر من دالة بحث بالاسم Search() واحدة للبحث عن موظف بمعلومية رقم الموظف والثانية للبحث عن الموظف باسم الموظف ويتم استدعاء الأولى أو الثانية حسب المعامل الذى تم إرساله.

ولتوضيح فكرة إنشاء أكثر من دالة بنفس الاسم Mehod Overloading تابع المثال التالى:

1- قم بإنشاء تطبيق جديد من النوع ConsoleApplication بالخطوات المعتادة كما سبق.

2- عدل سطور البرنامج وقم بإنشاء فصيلة جديدة بالاسم Shapes كما فى السطور التالية:

```
26.   Public Class shapes
27.       Public Overloads Sub drawLine()
28.           Dim i As Integer
29.   Console.WriteLine("First Method")
30.           For i = 0 To 20
31.               Console.Write("*")
32.           Next
33.           Console.WriteLine()
34.
35.       End Sub
36. Public Overloads Sub drawLine(ByVal n As Integer)
37.           Dim i As Integer
38.        Console.WriteLine("Secand Method")
39.           For i = 0 To n
40.               Console.Write("*")
41.           Next
42.           Console.WriteLine()
43.       End Sub
44.
45.   Public Overloads Sub drawLine(ByVal n As Integer, ByVal ch As Char)
46.           Dim i As Integer
47.    Console.WriteLine("Thrid Method")
48.           For i = 0 To n
49.               Console.Write(ch)
50.           Next
51.           Console.WriteLine()
52.       End Sub
53.
54.   End Class
55.
```

فى لغة C#

```
24        public class shapes
25        {
26            public void drawline()
27            {
28            for (int i=0;i<10;i++)
29                Console.Write ("*");
30            Console.WriteLine  ("");
31        }
32            public void drawline(int n)
33            {
34                for (int i=0;i<n;i++)
35                    Console.Write ("*");
36                Console.WriteLine  ("");
37            }
38
39            public void drawline(int n,char ch)
40            {
41                for (int i=0;i<n;i++)
42                    Console.Write (ch+"");
43                Console.WriteLine  ("");
44            }
45        }
```

فى لغة Java

```
1  class shapes
2  {
3    public void drawline ()
4    {
5      for(int i=0;i<10;i++)
6      System.out.print("*");
7      System.out.println("");
8      }
9
10   public void drawline (int n)
11   {
12     for(int i=0;i<n;i++)
13     System.out.print("*");
14     System.out.println("");
15     }
16
17     public void drawline (int n,char ch)
18   {
19     for(int i=0;i<n;i++)
20     System.out.print(ch);
21     System.out.println("");
22     }
23
24   }
25
26 public class ShapesApp {
27
28   public static void main(String[] args) {
29     shapes obj=new shapes();
30     obj.drawline();
31     obj.drawline(20);
32     obj.drawline(30,'-');
33
34   }
35 }
```

- فى هذه السطور تم إنشاء فصيلة بالاسم Shapes وداخل هذه الفصيلة Class تم تعريف ثلاث دوال بالاسم ()drawLine كلها باسم واحد ولكن الملاحظ أن الفرق بينها هو المعاملات الاولى بدون معاملات وتقوم بطباعة الحرف *عدد 20 مرة الثانية بمعامل واحد هو n من نوع Integer ويستخدم لتحديد عدد مرات طباعة الحرف *
- الثالثة لها معاملين الأول هو n من نوع Integer ويستخدم لتحديد عدد مرات طباعة الحرف * والثانى ch لتحديد الحرف المطلوب طباعته.
- بعد إنشاء الدوال الثلاثة داخل الفصيلة Shapes قم بكتابة أوامر استعمال الفصيلة Shapes ودوالها الثلاثة داخل الدالة الرئيسية للبرنامج ()main sub كما فى السطور التالية:

```
1. Sub Main()
2. Dim shapobj As New shapes()
3. shapobj.drawLine()
4. shapobj.drawLine(10)
5. shapobj.drawLine(15, "-")
6. 5-End Sub
```

فى لغة C#

```
1  using System;
2
3  namespace ConsoleApplication4
4  {
5          class Class1
6      {
7              [STAThread]
8          static void Main(string[] args)
9          {
10             shapes shobj =new shapes();
11             shobj.drawLine();
12             shobj.drawLine(15);
13             shobj.drawLine(12,'-');
14         }
15 }
```

فى هذه السطور:

- فى السطر رقم 1 تم تعريف هدف Object من الفصيلة Shapes كما سبق.
- فى السطر رقم 2 تم استدعاء الدالة ()drawLine مع هدف الفصيلة ولكن الملاحظ

إننا لم نكتب معاملات للدالة لذلك فأننا نستدعى الدالة الأولى التى لا تأخذ معاملات.

- فى السطر رقم 3 تم استدعاء الدالة drawLine(10) مع هدف الفصيلة ولكن الملاحظ إننا كتبنا معامل للدالة لذلك فأننا نستدعى الدالة الثانية التى تأخذ معامل واحد.
- فى السطر رقم 4 تم استدعاء الدالة drawLine(15, "-") مع هدف الفصيلة ولكن الملاحظ إننا كتبنا معاملين للدالة لذلك فأننا نستدعى الدالة الثالثة التى تأخذ معاملين.
- قم بتنفيذ البرنامج تحصل على نتيجة التنفيذ التى تعبر عن البرامج كما فى الشكل التالى (12-7):

الشكل (12-7)

الدالة Finalize

من الأفكار المتاحة أيضا فى البرمجة بالأهداف OOP فكرة وجود دالة Method داخل الفصيلة class هذه الدالة تنفذ عند الانتهاء من استعمال أهداف الفصيلة مثل إنهاء البرنامج أى عكس عمل دالة البناء Constructor.

وتستخدم هذه الدالة فى كثير من الاستخدمات أشهرها هو التخلص من أى متغيرات تم الاعلان عنها فى أول الفصيلة حتى لا تزدحم الذاكرة بمتغيرات غير مرغوب فيها فكما ذكرنا أن دالة الهدم هذه تنفذ تلقائيا عند الانتهاء من استعمال أهداف الفصيلة وتحقق Visual Basic.Net هذه الفكرة عن طريق إنشاء دالة بالاسم Finalize.

ولتوضيح ذلك قم بإضافة سطور هذه الدالة إلى سطور الفصيلة shapes فى المثال السابق كما فى السطور التالية:

```
Protected Overrides Sub Finalize()
        MsgBox("go go  go  go  go  go  go")
    End Sub
```

فى لغة C#

```
16          public class shapes
19          {
20              ~shapes()
21              {
22  Console.WriteLine  ("this is the Destructor");
23              }
```

- نفذ البرنامج وعند الخروج تلاحظ تنفيذ هذه الدالة وظهور الرسالة تلقائيا كما فى الشكل (13-7).

الشكل (13-7)

قواعد يفضل الالتزام بها عند كتابة البرامج:

الشكل التالى يعرض التركيب المثال للبرنامج من حيث اسم البرنامج والتعليقات وصاحب البرنامج وتاريخ كتابة البرنامج ومكتبات البرنامج وبيئة التطوير ومتطلبات التطوير .

من فضلك حاول مراجعة هذه القواعد وحاول الاسترشاد بها عند كتابة البرامج.

```
//*************************************************************
//*************************************************************
                        //  Program name
                    //  Copyright (c) 200? by
                     //  ALL RIGHTS RESERVED
//*************************************************************
//*************************************************************
          // Date:                              Coded by:
         // Filename:                          Module name:
         //                                    Source file:
                    // Program description:
                              //
//*************************************************************
               // Libraries and software support:
                              //
//*************************************************************
                    // Development environment:
                              //
//*************************************************************
                     // System requirements:
                              //
//*************************************************************
                           // Start date:
                        // Update history:
          //              DATE              MODIFICATION
                              //
//*************************************************************
                        // Test history:
     //  TEST PROTOCOL              DATE        TEST RESULTS
                              //
//*************************************************************
                     // Programmer comments:
                              //
                              //
//*************************************************************
                                 ***
```

ملاحظات يجب مراعاتها عند تصميم الفصائل:

1- حاول أن تحافظ على أن تكون البيانات من النوع الـ private.

2- حاول ان تعطى قيم ابتدائية للمتغيرات initialize data.

3- لا تستعمل متغيرات منفردة كثيرا فبدلا من ذلك حاول تجميعها فى فصيلة class.

فمثلا بالنظر إلى المتغيرات التالية :

```
private String street;
private String city;
private String state;
private int zip;
```

يمكن تجميعا فى فصيلة بالاسم Address.

4- حاول استخدام الصورة القياسية لتعريف الفصيلة class كما فى التركيب التالى:

```
public features

package scope features

private features

Within each section, we list:

instance methods

static methods

instance fields

static fields
```

5- حاول تقسيم الفصائل حسب الوظائف .. وتلاشى إنشاء فصيلة تتناول أكثر من موضوع ولتوضيح ذلك راجع هذا التصميم.

```
public class CardDeck // bad design
{
  public CardDeck() { . . . }
  public void shuffle() { . . . }
  public int getTopValue() { . . . }
  public int getTopSuit() { . . . }
  public void draw() { . . . }

  private int[] value;
  private int[] suit;
}
```

فى هذا التصميم تلاحظ احتوى على بيانات أكثر من موضوع وهذا التصميم سيئ بل يجب فصل الموضوعات بحيث لا تحتوى الفصيلة إلا على موضوع واحد .

وبالتالى يصبح التصميم الجيد كما فى السطور التالية:

```
public class CardDeck
{
   public CardDeck() { . . . }
   public void shuffle() { . . . }
   public Card getTop() { . . . }
   public void draw() { . . . }

   private Card[] cards;
}

public class Card
{
   public Card(int aValue, int aSuit) { . . . }
   public int getValue() { . . . }
   public int getSuit() { . . . }

   private int value;
   private int suit;
}
```

6- حاول اعطاء أسماء للفصائل والمتغيرات معبرة عن الغرض منها كمات يجب أن تلاحظ انه من المشهور إنشاء الدوال METHODS بالأسماء ()_SET و ()GET حيث أن مجموعة دوال ()Set_ تقوم باعطاء قيم لمتغيرات الفصيلة ومجموعة دوال ()Get_ تقوم باعادة قيم متغيرات الفصيلة.

الفصل الثامن

خاصية التوريث INHERITANCE

فى هذا الفصل Programming Concepts

8

في هذا الفصل نتناول أشهر خصائص مفهوم البرمجة بواسطة الأهداف OOP وهي خاصية التوريث Inheritance وكيفية تحقيقها وما يرتبط بها من أوامر وخصائص وذلك من خلال النقاط التالية :

- ما معنى خاصية التوريث Inheritance
- كيف كان يتم تطبيق خاصية التوريث في الإصدار السابق vb6
  كيف يتم تطبيق خاصية التوريث في vb.net
- مثال لتطبيق خاصية التوريث inheritance
- الكلمات المستعملة مع خاصية التوريث Inheritance
- تغيير الخصائص و الدوال في الفصيلة المشتقة overriding
- ما هي خاصية polymorphism

معنى خاصية التوريث Inheritance

هي خاصية قوية جداً في مفهوم البرمجة باستعمال الأهداف OOP وتستعمل لتقليل إعادة كتابة الأوامر، وهي أنه باستعمال خاصية التوريث تستطيع إنشاء فصيلة (CLASS) تحتوى على خصائص Proprities و دوال methods ثم استعمالها كأساس لفصائل أخرى وبالتالي لا نحتاج لكتابة ما كتبناه في تعريف الفصيلة الأولي (والتي تسمى basic class) مع كل فصيلة جديدة .

وتسمى الفصائل الجديدة التي تستورث فصيلة قديمة فصائل مشتقة derived class وغالباً يتم التعامل مع الفصائل classes بأن نستورث فصيلة ونكمل عليها .

ولتوضيح ذلك نضرب لك المثال التالي:

نفرض أنك تريد تعريف فصيلتان الأولي لتعريف بيانات الطالب و الثانية لتعريف بيانات المدرس فبدون استعمال خاصية التوريث سنضطر لإنشاء الفصيلتان برغم من تشابه الفصيلتان واحتوائهما على بيانات متشابه فمثلاً فصيلة الطالب تحتوى على البيانات التالية :

- الكود ، الاسم ، العنوان ، التليفون ، المؤهل ، تاريخ الميلاد ، وكثير من البيانات

وتحتوى على الدوال التالية :

- دالة تسجيل بيانات الطالب
- دالة حذف بيانات الطالب
- دالة تعديل بيانات الطالب (ودوال أخرى)

وعند النظر إلي فصيلة المدرس سوف نجد أنها تشتمل علي كثير من البيانات و الدوال الأعضاء في فصيلة الطالب بالاضافة لبعض الأعضاء الجديدة .

وبالتالي باستعمال خاصية التوريث علينا توريث فصيلة الطالب لفصيلة المدرس وإضافة الجديد إليها مما يوفر علينا إعادة كل شيء ويمكن تكرار هذه الفكرة مع فصائل جديدة أخرى مما يوفر الكثير من الوقت بالاضافة ان هذا الاسلوب يسمح لك باستعمال فصائل classes تم اعدادها وتم اختبارها.

التوريث في VB.Net

أما VB.NET فقد وفر إمكانية التوريث الحقيقية ، حيث لا يسمح لك بإنشاء فصائل كاملة بكل دوالها (تعتبر كفصائل أساس Base class) ثم توريث هذه الفصائل بكل دوالها ومتغيراتها لفصائل جديدة وذلك يوفر عليك الكثير وهذا يعني أنه يمكنك أن تنشأ form جديدة بناء علي form قديمة لأن form في حد ذاتها فصيلة.

وفي نفس الوقت يمكنك تنفيذ الفكرة القديمة (Interface) في VB.NET ولتوضيح كيفية تحقيق عملية التوريث في VB.NET تابع المثال التالي:

1- ابدأ تطبيق وحدد نوع ConsoleApplication كما سبق

2- اكتب سطور تعريف فصيلة class بالاسم Person كما في السطور التالية:

```
1.  Public Class person
2.  Public Pname, Paddress As String
3.  Public Sub SetData(ByVal name_v As String, ByVal    address_v As String)
4.  Pname = name_v
5.  Paddress = address_v
6.  End Sub
7.  Public Function getName()
8.  Return Pname
9.  End Function
10. Public Function getAddress()
11. Return Paddress
12. End Function
13. End class
```

في لغة C#

```
public class person
{
    public string Pname, Paddress;
    public void  SetData( string name_v ,   string address_v )
{
Pname = name_v;
Paddress = address_v;
}
    public string  getName()
    {
    return Pname;
    }
    public string  getAddress()
    {
        return Paddress;
    }
}
```

فى لغة Java

```
class Person
{
    public String Pname,Paddress;
    public void Set_Data(String Name_V,String address_V)
      {

        Pname=Name_V;
        Paddress=address_V;
    }

    public String Get_Name ( )
      {
       return Pname;
       }

    public String Get_Address ( )
      {
          return Paddress;
       }
  }
public class InheritApp1 {

    public static void main(String[] args) {
    } }
```

- في هذه السطور تم إنشاء فصيلة بالاسم person وتم تعريف متغيرات.
- وتم كتابة فهي فصيلة متكاملة تحتوى على المتغيرات Pname, Paddress.
- وتحتوى على الدالة () SetData التى تقوم بتسجيل بيانات الفصيلة.
- وتحتوى على الدالة ()getName التى تعيد قيمة المتغير pname الخاص بالفصيلة والخاص باسم الشخص وكذلك الدالة ()getAddress التى تعيد قيمة المتغير paddress .
- وعند إنشاء فصيلة جديدة وتوريثها هذه الفصيلة Person يتم استعمال متغيراتها ودوالها مباشرة دون الحاجة إلى إنشائها مرة أخرى .

وتوريث هذه الفصيلة لفصيلة جديده يتم كما يلي:

3- أكتب سطور فصيلة جديدة بالاسم student

```
Public Class student
Inherits person
Dim degree As Integer
Public Sub setdegree(ByVal degree_v As Integer)
degree = degree_v
End Sub
Public Function retdegree()
Return degree
End Function
End Class
```

فى لغة C#

```
57      public class student: person
58      {
59      int degree;
60          public void  setdegree(int  degree_v)
61      {
62          degree = degree_v;
63       }
64
65          public int  retdegree()
66          {
67              return degree;
68          }
69
70      }
```

فى لغة Java

```
25  class Student extends Person
26  {
27  int degree;
28      public void Set_Degre(int degree_V)
29      {
30
31       degree=degree_V;
32
33  }
34  public int RetDegree()
35  {
36    return degree;
37  }
38
39  }
```

في هذه السطور:

- تم إنشاء فصيلة جديدة بالاسم student.
- وتم استعمال العبارة inherits person ومعناها قم بتوريث الفصيلة person بكل ما فيها للفصيلة الجديدة (student) و بالتالي أصبحت نفس إعضاء الفصيلة القديمة person اعضاء فى الفصيلة الجديدة student و هذا يسمح لك باستعمالها مباشرة وذلك كما يلي :

فى الدالة الرئيسية (Sub Main قم بكتابة سطور استعمال كلا من الفصيلتين person,student كما فى السطور التالية:

```
1. Dim vname, vaddrs As String
2. Dim no As New person()
3. no.SetData("azab", "Cairo")
4. vname = no.getName()
5. vaddrs = no.getAddress()
6. Console.WriteLine(vname)
7. Console.WriteLine(vaddrs)
8. Dim st As New student()
9. St.setdegree(100)
10. Dim vd As Integer
11. dim vname2,vaddress2 as string
12. vd = st.retdegree()
13. st. SetData("omr", "Giza")
14. vname = st.getName()
15. vaddress2 = st.getAddress()
16. Console.WriteLine(vd)
17. Console.WriteLine(vname)
18. Console.WriteLine(vaddress2)
```

فى لغة C#

```
12        static void Main(string[] args)
13        {
14
15   string vname, vaddrs ;
16  person no = new person ();
17      no.SetData("azab", "Cairo");
18  vname = no.getName();
19  vaddrs = no.getAddress();
20  Console.WriteLine(vname);
21  Console.WriteLine(vaddrs);
22      student st =new student();
23
24      st.setdegree(100);
25      int  vd ;
26  string vaddress2;
27      vd = st.retdegree();
28      st. SetData("omr", "Giza");
29      vname = st.getName();
30      vaddress2 = st.getAddress();
31      Console.WriteLine(vd);
32      Console.WriteLine(vname);
33      Console.WriteLine(vaddress2);
34
35  }
```

فى لغة Java

```
Mathoperations.java | ShapesApp.java | InheritApp1.java
42  public class InheritApp1 {
43
44      public static void main(String[] args) {
45          String vname,vaddress;
46          Person no=new Person();
47          no.Set_Data("Azat","Cairo");
48          vname=no.Get_Name ( );
49          vaddress=no.Get_Address ( );
50          System.out.println(vname);
51          System.out.println(vaddress);
52
53  Student st=new Student();
54  st.Set_Degree(100);
55  int vd;
56  String vaddress2;
57  vd=st.RetDegree();
58  st.Set_Data("Omar","Giza");
59
60      vname=st.Get_Name ( );
61          vaddress2=st.Get_Address ( );
62
63          System.out.println("vd:"+vd);
64       System.out.println("vname:"+vname);
65          System.out.println("vaddress:"+vaddress);
66
67          }
68  }
```

في هذه السطور

- في السطر رقم 1 تم تعريف المتغيرات vname, vaddrs.
- في السطر رقم 2 تم تعريف المتغير no(هدف) في الفصيلة person.
- فى السطور 3 و 4 و 5 و 6 و 7 تم التعامل مع هدف الفصيلة person باستدعاء الدوال والتعامل معها كما سبق.
- في السطر رقم 8 تم تعريف المتغير st من الفصيلة student.
- في السطر رقم 9 تم استدعاء الدالة الجديدة المضافة للفصيلة بالاسم setdegree.
- في السطر رقم 10 و 11 تم الاعلان عن متغيرات.
- في السطر رقم 13 تم استدعاء الدالة SetData() المعرفة داخل الفصيلة الأساسية Person فبالرغم من عدم وجود الدالة داخل الفصيلة الجديدة إلا أننا استعملناها لأنها معرفة فى الفصيلة التى استورثناها بالاستعمال الأمر Inherits.

- في السطر رقم 14 و 15 تم استدعاء دوال الفصيلة الأساسية Person فبالرغم من عدم وجود هذه الدوال داخل الفصيلة الجديدة ألا إننا استعملناها لانها معرفة فى الفصيلة التى استورثناها بالاستعمال الأمر Inherits.

من هذا المثال نلاحظ استعمال الخصائص المستورثه من الفصيلة person مع الفصيلة student بالرغم من عدم تعريفها أو عدم الإعلان عنها والسطور التالية تعرض النص الكامل للبرنامج.

```
Module Module1
    Sub Main()
        Dim vname, vaddrs As String
        Dim no As New person()
        no.SetData("azab", "Cairo")
        vname = no.getName()
        vaddrs = no.getAddress()
        Console.WriteLine(vname)
        Console.WriteLine(vaddrs)

        Dim st As New student()
        st.setdegree(100)
        Dim vd As Integer
        Dim vname2, vaddress2 As String
        vd = st.retdegree()
        st.SetData("omr", "Giza")
        vname = st.getName()
        vaddress2 = st.getAddress()
        Console.WriteLine(vd)
        Console.WriteLine(vname)
        Console.WriteLine(vaddress2)

    End Sub
    Public Class person
        Public Pname, Paddress As String

        Public Sub SetData(ByVal name_v As String, ByVal address_v As String)
            Pname = name_v
            Paddress = address_v
        End Sub
        Public Function getName()
            Return Pname
```

```
    End Function
    Public Function getAddress()
      Return Paddress
    End Function
  End Class
  Public Class student
    Inherits person
    Dim degree As Integer
    Public Sub setdegree(ByVal degree_v As Integer)
      degree = degree_v
    End Sub
    Public Function retdegree()
      Return degree
    End Function
  End Class
End Module
```

فى لغة C#

```
using System;
namespace ConsoleApplication5
{
/// <summary>
/// Summary description for Class1.
/// </summary>
class Class1
{

[STAThread]
static void Main(string[] args)
{

string vname, vaddrs ;
person no = new person ();
no.SetData("azab", "Cairo");
vname = no.getName();
vaddrs = no.getAddress();
Console.WriteLine(vname);
Console.WriteLine(vaddrs);
student st =new student();

st.setdegree(100);
int  vd ;
string vaddress2;
```

```
vd = st.retdegree();
st. SetData("cmr", "Giza");
vname = st.getName();
vaddress2 = st.getAddress();
Console.WriteLine(vd);
Console.WriteLine(vname);
Console.WriteLine(vaddress2);

}
}
public class person
{
public string Pname, Paddress;
public void  SetData( string name_v       string
address_v  )
{
Pname = name_v;
Paddress = address_v;
}
public string  getName()
{
return Pname;
}
public string  getAddress()
{
return Paddress;
}
}

public class student: person
{
int degree;
public void  setdegree(int  degree_v)
{
degree = degree_v;
}

public int  retdegree()
{
return degree;
}

}
}
```

فى لغة Java

```
class Person
{
public String Pname,Paddress;
public void Set_Data(String Name_V,String
address_V)
{

Pname=Name_V;
Paddress=address_V;
}

public String  Get_Name ( )
{
return Pname;
}

public String  Get_Address ( )
{
return Paddress;
}
}

class Student  extends Person
{
int degree;
public void Set_Degree(int degree_V)
{

degree=degree_V;

}
public int  RetDegree()
{
return degree;
}

}
public class InhertAppl {
```

```
public static void main(String[] args) {
String vname,vaddress;
Person no=new Person();
no.Set_Data("Azab","Cairo");
vname=no.Get_Name ( );
vaddress=no.Get_Address ( );
System.out.println(vname);
System.out.println(vaddress);

Student st=new Student();
st.Set_Degree(100);
int vd;
String vaddress2;
vd=st.RetDegree();
st.Set_Data("Omar","Giza");

vname=st.Get_Name ( );
vaddress2=st.Get_Address ( );
System.out.println("vd:"+vd);
System.out.println("vname:"+vname);
System.out.println("vaddress:"+vaddress);
}
}
```

فى لغة VB.NET

كلمات التوريث Inheritance Keywords

هناك مجموعة من الكلمات التي تستعمل مع عملية توريث الفصائل Inheritance وذلك لتحقيقها والتحكم فيها ، فمثلاً يمكنك منع توريث فصيلة أو العكس ،أو يمكنك استعمال فصيلة بدون توريثها لغيرها حل ذلك من خلال الكلمات المستعملة لهذا العرض ونعرضها كما يلي :

الكلمات Mustinheritable،Notinheritable

الأصل في الفصائل التي تنشئها أنها تعمل بمفردها أي يمكن استعمالها مباشرة بتعريف متغير (هدف) من الفصيلة ، وكذلك يمكن توريثها لغيرها ،ولكن يمكن تغيير ذلك كما يلي :

الأمر Notinheritable

هذا الأمر عند ما يكتب أمام الفصائل عند إنشائها يعني أنها غير قابلة للتوريث ويتم ذلك عند إنشاء الفصيلة بالصورة التالية :

```
public Notinheritable class Employee

End class
```

في السطور السابقة تم تعريف (الإعلان عن) فصيلة جديدة (class) بالاسم Employee ولكن الجديد هو كتابة الأمر notinheritable الذى لا يسمح بتوريث هذه الفصيلة لفصيلة أخرى ، ولكن يسمح فقط بتعريف متغير (هدف) من الفصيلة للتعامل معها ، وبالتالي لا يمكنك عند إنشاء فصيلة جديدة كتابة الأمر inherits واسم هذه الفصيلة.

الأمر Mustinherit

يؤدى هذا الأمر إلى نتيجة عكس الأمر Notinheritable فهو لا يسمح باستعمال الفصيلة المحددة مباشرة بل لابد من توريثها لفصيلة أخرى لتعريف متغير واستعمالها فمثلاً عند إنشائك فصيلة بالشكل التالي :

```
public Mustinherit class student

End class
```

في هذه الحالة لا يمكنك استعمال هذه الفصيلة مباشرة بتعريف متغير.

فلا يمكنك كتابة السطر التالي :

```
Dim ST AS New student ( )
```

هذا السطر يطلب تعريف المتغير ST من نوع الفصيلة Student ونظراً لوجود الأمر Mustinherit فلابد من توريث هذه الفصيلة ثم استعمال الفصيلة الجديدة .

كما في السطور التالية :

```
public class instructor
  Inherits student

End class
```

بهذا تم تعريف فصيلة جديدة وتوريث الفصيلة student لها ، و بالتالي يمكن تعريف متغير منها كما يلي :

```
Dim NS AS  New instructor ( )
```

تغيير الدوال في الفصيلة الجديدة overriding proprities & Methods

في لغة VB.NET

من الإمكانيات المتاحة في مفهوم OOP البرمجة بواسطة الأهداف خاصية تسمى Overriding وهذه الخاصية تعني إمكانية كتابة دوال جديده في الفصائل الجديدة بنفس اسم الدوال الموجودة في الفصيلة الأساس (Base class) التي تم توريثها أي أن الدالة موجودة في الفصيلة الأساسية ومع ذلك تم إنشائها مرة أخرى بنفس الاسم في الفصيلة الجديدة و بالتالي يتم إلغاء الدالة من الفصيلة الجديدة واعتماد الدالة الجديدة وهذا يسمى تركيب دالة على أخرى overriding ولتوضيح ذلك ننشأ فصيلة بها دالة ثم نستورث هذه الفصيلة و ننشأ الدالة السابقة في الفصيلة الجديدة و يظهر ذلك من السطور التالية:

```
1: public class one
2: public overridable function say( )
3:   MSGbox "THIS Msg from  class one   "
4:  End function
5: End class
6: Public class two
7:inherits one
8:  public overrides function say( )
9:    Msgbox "this Msg from class two..."
10:  end function
11: end class
```

في لغة C#

```
public class one
{
    public virtual void say( )
        {
    Console.WriteLine ("THIS Msg from  class one   " );
        }
}
public class two:one
{
    public override void say( )
    {
        Console.WriteLine ("THIS Msg from  class two   " );
    }
}
```

فى لغة Java

Override.java

```
1  class One
2  {
3     public void Say()
4     {
5     System.out.println("This Msg from class One");
6
7     }
8  }
9  class Two extends One
10 {
11    public void Say()
12    {
13    System.out.println("This Msg from class Two");
14
15    }
16 }
17 public class Override {
18
19    public static void main(String[] args) {
20
21
22    }
23 }
```

في هذه السطور:

- في السطر رقم 1 تم تعريف فصيلة جديدة بالاسم one.
- في السطر رقم 2 تم تعريف دالة بالاسم ()say مع وضع الكلمة overiridable أمام الدالة وهذا يعني إمكانية التركيب عليها أي إنشاء دالة بنفس الاسم في الفصيلة المشتقة ،ثم تم إنشاء الدالة ()Say في الفصيلة الجديدة برسالة جديدة وإنهاء الفصيلة.
- في السطر رقم 6 تم إنشاء فصيلة بالاسم two.
- في السطر رقم 8 تم إنشاء دالة للفصيلة الجديدة بنفس الاسم say وهو اسم الدالة الموجودة في الفصيلة الأساس التي تم توريثها(one) مع وضع الكلمة overrides أمام الدالة ليعني انها بديل الدالة موجودة في الفصيلة الأساس.

- في السطر رقم 7 تم توريث الفصيلة القديمة one .
- بهذا يصبح للفصيلة الأساس (one) دالة بالاسم say .
- ويصبح للفصيلة الجديدة (two) دالة بالاسم say .
- وعند تعريف متغير من كل منهما واستدعاء الدالة say يتم استدعاء الدالة الخاصة بكل منهما بالرغم من وجود خاصية التوريث بها.

ولتوضيح ذلك تابع ما يلي :

بعد كتابة سطور الفصيلتان اكتب السطور التالية فى الدالة الرئيسية للبرنامج Sub Main()

```
1: Dim a as new one ( )
2: Dim b as new two ( )
3: a. say ( )
4: b. say ( )
```

فى لغة C#

```
8       class Class1
9       {
10          /// <summary>
11          /// The main entry point for the application.
12          /// </summary>
13          [STAThread]
14          static void Main(string[] args)
15          {
16  one a = new one ( );
17  two b = new two ( );
18   a. say  ( );
19   b. say ( );
20
21          }
22      }
```

في لغة Java

```
17 public class Override {
18
19   public static void main(String[] args) {
20
21     One a=new One();
22     Two b=new Two();
23     a.Say();
24     b.Say();
25   }
```

في هذه السطور:

- في السطر رقم 1،2 المتغيرات a , b من الفصائل one, two.
- في السطر رقم 3 تم استدعاء الدالة say مع المتغير a المأخوذ في الفصيلة one وبالتالي يتم استدعاء الدالة الأولي وتظهر الرسالة الأولي.
- في السطر رقم 4 يتم استدعاء الدالة say مع المتغير a المعرف من الفصيلة b وبالتالي يتم استدعاء الدالة الثانية وتظهر الرسالة الثانية بالرغبة في توريث الفصيلة الأولي المحتوية علي الدالة say للفصيلة الثانية الا إن إنشاء دالة جديدة بنفس الاسم أدي إلي إلغاء الدالة say مع الفصيلة الجديدة واستعمال الدالة الخاصة بها.

الأمر Mustoverride

هذا الامر عندما يوضع أمام دالة في الفصيلة فإنه يعني عدم إمكانية استدعاء هذه الدالة مباشرة بل لابد من تعريف دالة جديده بنفس الاسم حتى يمكن استعمالها وبالتالي فهذه الدالة تكتب للتعريف فقط.

درجات حماية أعضاء الفصيلة Access Modifiers

عند الإعلان عن أعضاء الفصيلة سواء بيانات data أو دوال Methods لاحظنا إننا كنا نكتب كلمة public قبل الإعلان هذه الكلمة تحدد درجة حماية للاعضاء وتسمى Access Modifiers وكلمة public تعنى عام أى يمكن التعامل معه مباشرة من خارج الفصيلة بدون الاحتياج لى دالة من الفصيلة للتعامل معه.

ربما يكون هذا الشرح غير واضح لأنه نظرى ولكن تعال أولا نعرض لك الكلمات المختلفة التى يمكن أن تستعمل مكان الكلمة public لتحديد درجة حماية الاعضاء والتى تسمى كما أشرنا Access Modifiers وهى:

1- الكلمة Public كما أشرنا تستعمل لتحديد أن هذه الاعضاء عامة يمكن استعمالها من خارج الفصيلة أو يمكن الوصول اليها من خارج الفصيلة.
2- الكلمة Private عكس الكلمة Public وتعنى خاص وتعنى أن هذه الاعضاء لا يمكن الوصول اليها من خارج الفصيلة فقط يتم التعامل معها من داخل الفصيلة و من خلال أعضاء الفصلة.
3- الكلمة Protected هى نفس عمل الكلمة private عدا فى حالة الفصائل المشتقة تستطيع التعامل معها.

بعد سرد معانى الكلمات الثلاثة تعال نوضح ذلك عمليا

1- ارجع إلى البرنامج السابق والفصيلة Person والتى تم فيها الإعلان عن متغيرات سبقت بالكلمة Public كما فى السطر التالى:

```
Public Pname, Paddress As String
```

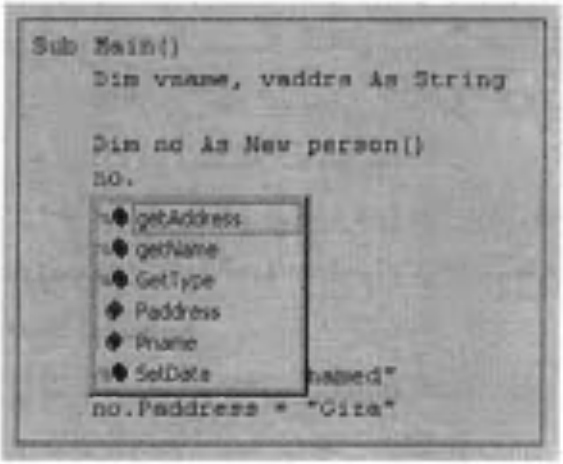
فى هذه الحالة يمكن التعامل مع هذه المتغيرات خارج الفصيلة بمجرد تعريف هدف Object من الفصيلة class كما فى السطور التالية:

```
Dim no As New person()
no.Pname = "mohamed"
no.Paddress = "Giza"
```

فى هذه السطور تم التعامل مباشرة مع المتغيرين Pname,Paddress.

وعندما يتم الإعلان عن الأعضاء بالكلمة Public تظهر هذه الأعضاء مع الهدف عند

استعماله ، إرجع الى تعريف الهدف Object ثم استعماله تلاحظ بمجرد كتابة اسم الهدف object ثم كتابة نقطة . تظهر الأعضاء الممكن التعامل معهم كما فى الشكل (1-8).



الشكل (1-8)

فى هذا الشكل تلاحظ ظهور جميع أعضاء الفصيلة Person لتعريفهم بالكلمة Public.

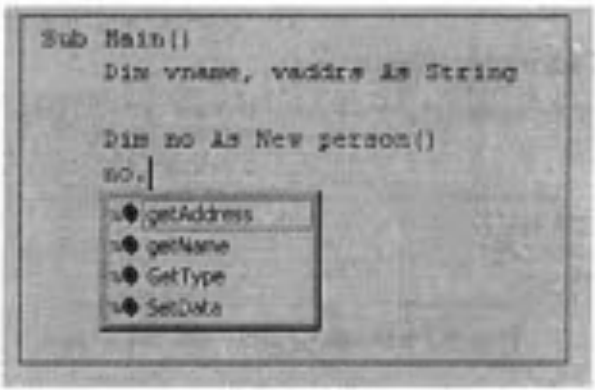
2- استبدل الكلمة Public بالكلمة Private عند تعريف المتغيرات كما فى السطر التالى:

```
Private Pname, Paddress As String
```

فى لغة C#

```
Private string  Pname, Paddress;
```

جرب التعامل مع هدف الفصيلة بنفس الطريقة السابقة بالتعامل المباشر مع المتغيرات تلاحظ أن المتغيرات لا تظهر ضمن قائمة الأعضاء كما فى الشكل (2-8):



الشكل (2-8)

فى هذا الشكل تلاحظ عدم وجود المتغيرات التى تم الإعلان عنها داخل الفصيلة person بالرغم من أنهم أعضاء بها ، ذلك للاعلان عنهم بالكلمة Private .

فى هذه الحالة لا يمكن الوصول إليهم من خارج الفصيلة إلا من خلال دالة Method عضوة فى الفصيلة كما سبق فى الدوال التى عرفناها كما فى السطور التالية:

```
Public Function getName()
   Return Pname

End Function

Public Function getAddress()
   Return Paddress
End Function
```

ملحوظة:

لا يتم الإعلان عن الدوال بالكلمة Private وإلا لن يمكن التعامل معها من خارج الفصيلة وبهذا تفقد أهميتها إلا إذا كنت تريد انشاء دوال للتعامل معها داخل المصنوفة فقط.

متى تستعمل خاصية التوريث ومتى لا نستعملها ؟

هذا السؤال ربما يرد إلى ذهنك عندما تعمل مع الفصائل وإجابته كما يلي:

إذا كانت الوظيفة المطلوبة تحقيقها ليست بكبيرة بحيث لا تحتاج فصيلة جديدة تقوم فيها بإنشاء دوال و تعريف متغيرات فلا تستخدم خاصية التوريث بل تكتب الأوامر التي تحقق العملية فقط ، مثال لذلك إذا كان لديك أداة نص textbox و تريد تحويل لون الأرقام السالبة عندما تكتب به إلى اللون الأحمر فلا تقوم بتوريث فصيلة textbox الى فصيلة جديدة ، وتعريف دالة لذلك ، ولكن اكتب الأوامر التي تحقق العملية لأنها لا تحتاج إلى فصيلة.

إذا كانت الفصيلة الجديدة (التي تستورث فصيلة قديمة) بها لن تستفيد كثيراً من الفصيلة الأساس (المستورثة) فقط سوف تستورثها لإنشاء دوالها من جديد بخاصية override فهذه الحالة يكون التوريث فيها غير جيد.

ولكن يمكنك إنشاء فصيلة بدوال بدون كتابة سطور الدوال (interface) ثم توريثها و بالتالي ننشأ الفصائل الجديدة حسب تركيب (interface) وخاصة إذا كان هناك أكثر من فصيلة سوف نشترك في التركيب فقط.

إذا كانت الفصيلة الجديدة (التي تستورث) تحتاج إلي دوال الفصيلة القديمة بنفس السطور المكتوبة فيها وتحتاج إلي الخواص والمتغيرات ولكن تريد الإضافة إليها فهذه هي الحالة المناسبة للتوريث وهو الاستفادة الكبيرة من الفصيلة المستورثها وإضافة الجديد بما يناسب العمل .

الفصل التاسع

تابع مفاهيم OOP

Programming Concepts في هذا الفصل

9

في هذا الدرس نتسكمل شرح مفاهيم OOP وذلك من خلال النقاط التالية :

- إضافة خصائص properties للفصيلة
- ما هى الخاصية Polymorphism
- ما هى NameSpace
- كيفية إنشاء NameSpace
- ما هى مكتبة الفصائل Class Lib
- كيفية إنشاء مكتبة فصائل Class Lib
- properties
- تحديد الخاصية الافتراضية default propority

إضافة خصائص properties للفصيلة

من الإمكانيات المتاحة عند التعامل مع الفصائل هى إمكانية إضافة خصائص Properties للفصيلة class مثل التى نتعامل معها عند التعامل عند التعامل مع الأدوات tools ويوفر Visual Studio.Net ذلك بسهولة ولتوضيح ذلك تابع الخطوات التالية:

فى لغة VB.NET

1- قم بانشاء تطبيق جديد من النوع ConsoleApplication كما سبق

2- قم بانشاء فصيلة Person كما سبق وكما فى السطور التالية:

```
Public Class person
Private Pname, Paddress As String
End Class
```

فى لغة C#

```
public class person
  {
     private string name;
  }
```

فى هذه السطور:

تم إنشاء فصيلة جديدة Class بالاسم person وهذه الفصيلة class تحتوى على المتغيرين Pname,Paddress والمطلوب إنشاء خاصية بالاسم Name للتعامل مع المتغير Pname بحث يتم تسجيل قيمة هذا المتغير وأخذ قيمة هذا المتغير من خلال هذه الخاصية ولتحقيق ذلك قم بكتابة سطر تعريف خاصية جديدة بالاسم Name كما فى السطر التالى:

```
Public Property Name()
```

- فى هذا السطر يتم الإعلان عن خاصية بالاسم Name باستعمال الكلمة المحجوزة Property.
- وبمجرد كتابة هذا السطر يقوم Visual Basic .Net بكتابة باقى السطور المطلوبة لإنشاء الخاصية كما فى السطور التالية:

```
Public Class person
Public Pname, Paddress As String
Public Property Name()
Get
```

```
End Get
Set(ByVal Value)
End Set
End Property
End Class
```

فى لغة C#

```
public string Name
    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }
```

فى هذه السطور:

قام Visual Basic.Net باضافة أجزاء الخاصية Name تلقائيا حيث اضيفت الدالة Get التى تنفذ عندما تريد أخذ قيمة الخاصية أى عند تقوم باستعمال الخاصية وتكتب السطر:

```
R=Obj.Name
```

فى هذه الحالة أنت تطلب أخذ قيمة الخاصية Name لذلك تنفذ الدالة Get وعليك كتابة الاوامر التى تحقق ذلك وهى ببساطة كما فى السطر التالى:

```
Return Pname
```

فى هذا السطر يتم اعادة قيمة المتغير Pname وهو المطلوب فى حالة Get.

وفى حالة تسجيل قيمة المصفوفة تنفذ الدالة Set والتسجيل يتم كما فى الصورة التالية:

```
Obj.Name="omr"
```

فى هذه الحالة تطلب تسجيل قيمة الخاصية Name بالقيمة omr لذلك نكتب فى الدالة Set السطر التالى:

```
Pname=value
```

وهذا معناه ان القيمة omr توضع تلقائيا فى المتغير Value الذى نضع قيمته فى المتغير Pname وبالتالى تصبح سطور الخاصية داخل الفصيلة ؛ كاملة كما فى السطور التالى:

```
Public Class person
    Public Pname, Paddress As String
    Public Property Name()
      Get
        Return Pname
      End Get
      Set(ByVal Value)
        Pname = Value
      End Set
    End Property

  End Class
```

فى لغة C#

```
public class person
  {
    private string name;

    // A read-write instance property:
    public string Name
    {
      get
      {
        return name;
      }
      set
      {
        name = value;
      }
    }
```

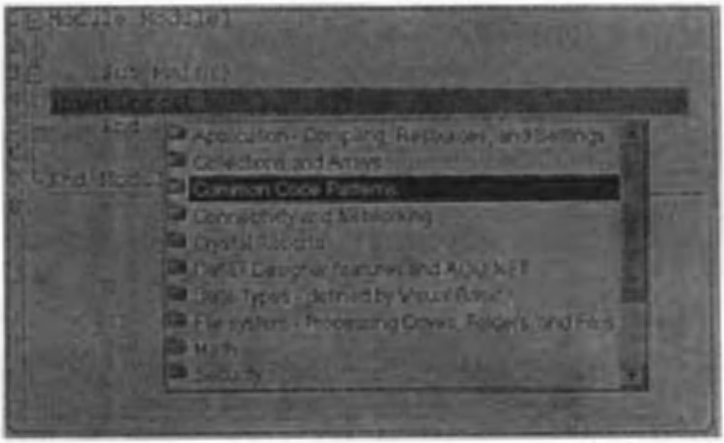
من فضلك قم بمراجعة سطور الفصيلة class وكذلك سطور تعريف خاصية Property بالاسم Name وحاول تكرار ذلك وإنشاء خاصية بالاسم address للتعامل مع المتغر Paddress.

قطع الاكواد الجاهزة : Snippet

1- اضغط طرف الماوس الأيمن داخل صفحة الأكواد تحصل على قائمة كما فى الشكل (1-9).

الشكل (9-1)

اختر Insert snippet كما فى الشكل السابق تحصل على قائمة بمجموعات الأكواد التى كما يوجد فى الشكل (9-2).

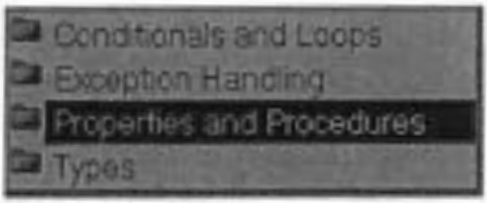


الشكل (9-2)

كما فى الشكل(9-3).

الشكل (9-3)

2- فى هذا الشكل توجد مجموعات الأكواد الأساسية مثل جمل الشرط والتكرار كما يوجد فى الشكل (4-9).



الشكل (4-9)

3- وعند اختيار properties and Procedur تحصل على أنواع كما فى الشكل (5-9).

الشكل (5-9)

اختر تعريف خاصية NewProperty تحصل على الأكواد كما فى السطور.

```
Private newPropertyValue As Integer
   Public Property NewProperty() As Integer
      Get
         Return newPropertyValue
      End Get
      Set(ByVal value As Integer)
         newPropertyValue = value
      End Set
   End Property
```

في هذه السطور:

تم الإعلان عن الخاصية مع تحديد دوال تسجيل Set وإعادة القيمة Get.

ما هي NameSpace

هي شكل تنظيمي يمكن أن يحتوى على أكثر من فصيلة Class ويمكن للـ NameSpace أن يحتوى على أكثر من NameSpace ويتم فتح هذه المكتبات NameSpace باستعمال كلمة Imports كما يحدث في أول البرنامج حيث تلاحظ وجود السطر.

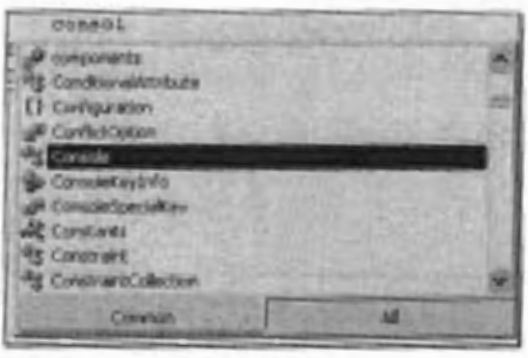
```
Imports System
```

في لغة C#

```
using System;
```

فهذا السطر أمر الى المترجم بأن يقوم باستيراد ملف الفصائل Classes المسمى system لأننا سوف نستعمل بعض الفصائل المعرفة بداخله وإلا سوف نضطر للاشارة إلى System كلما أردنا التعامل مع أي فصيلة class داخل هذا الـ NameSpace.

ولتوضيح ذلك ارجع إلى بيئة التطوير وداخل محرر سطور الأوامر اكتب .System تلاحظ ظهور الفصائل classes الموجودة به كما في الشكل (6-9).



الشكل (6-9)

وكما أشرنا يمكن للـ NameSpace أن يحتوى على Classes ويمكن أن يحتوى على NameSpaces.

كيفية انشاء NameSpace

ويتم انشاء NameSpace بسهولة بمجرد كتابة كلمة NameSpace ثم الاسم وفى النهاية يتم انهائها بالامر End NameSpace تابع الخطوات التالية:

1- قم بإنشاء تطبيق جديد من النوع ConsoleApplication.

2- اكتب فى نهاية سطور الأوامر وبعد الأمر End Module سطور تعريف NameSpace بالاسم OurNameSpace كما فى السطور التالية:

```
1. Namespace OurNameSpace
2. Public Class class1
3. Public var1, var2 As String
4. End Class
5. Public Class class2
6. Public var11, var22 As String
7. End Class
8. End Namespace
```

فى لغة C#

```
namespace OurNameSpace
{
    public class class1
    {
        public string var1, var2;
    }

    public class class2
    {
        public string var11, var21;
    }
```

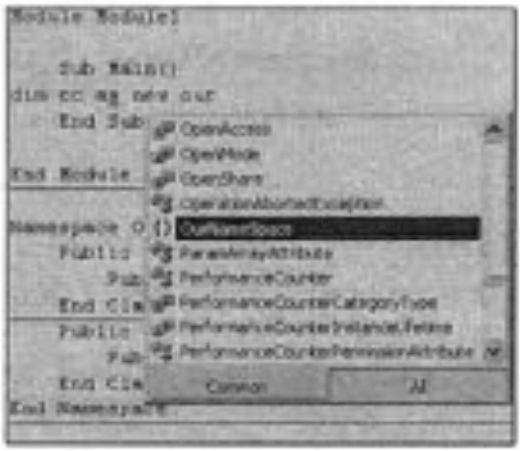
فى هذه السطور :

- فى السطر رقم 1 تم استعمال الأمر Namespace لإنشاء Namespace جديد بالاسم **OurNameSpace**.
- فى السطر رقم 2 تم الإعلان عن فصيلة class جديدة بالاسم class1 .

- فى السطر رقم 5 تم الإعلان عن فصيلة class جديدة بالاسم class2.
- فى السطر رقم 8 تم إنهاء الـ Namespace باستعمال الأمر end Namespace.

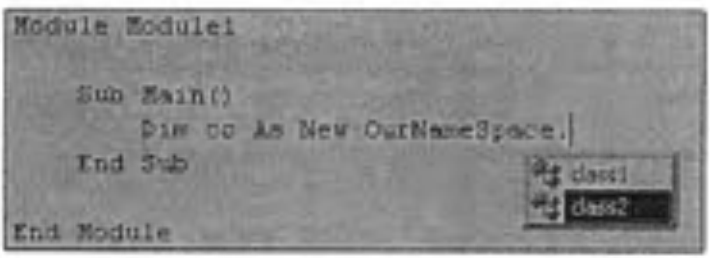
استعمال الـ Namespace

بعد إنشاء Namespace وإنشاء فصائل classes بداخله المطلوب كيفية استعمال الفصائل classes المعرفة داخل الـ Namespace ويتم ذلك بتعريف هدف Object من الفصيلة class المعرفة داخل Namespace ولكن بعد نسبتها إلى الـ Namespace ولتوضيح عد الى بيئة التطوير وداخل الدالة الرئيسية للبرنامج Sub Main() اكتب تعريف هدف Object تلاحظ ظهر اسم الـ Namespace كما فى الشكل (7-9) .



الشكل(7-9)

- اختر الـ Namespace الخاصة بنا والتى أنشأناها بالاسم OurNameSpace ثم اكتب . تلاحظ ظهور فصائلها كما فى الشكل (8-9) .



الشكل(8-9)

فى لغة C#

```
19          person p=new person();
20          p.Name ="Omr";
21      OurNameSpace cc=new OurNameSpace.
22                                  class1   class OurNameSpace.class1
23                                  class2
24  }
```

فى هذا الشكل تلاحظ ظهور فصائل classes الـ Namespace التى أنشاناها ويمكنك التعامل معها بباقى الخطوات كما سبق.

إنشاء مكتبة الفصائل creating class library

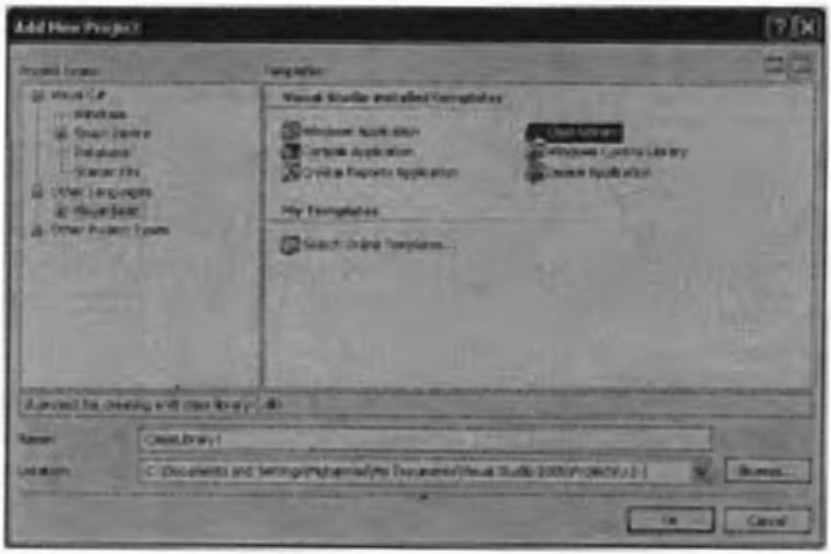
بعد شرح بعض المفاهيم المهمة لمفهوم OOP تعال معنا نشرح عملياً كيفية إنشاء مكتبة الفصائل class Library هى احد انواع التطبيقات التى يوفرها Visual Basic.Net حيث يسمح بانشاء تطبيق عبارة عن ملف بالامتداد DLL(Dynamic Link Lib) يحتوى على مجموعة من الفصائل Classes التى يمكن استعمالها من داخل اى برنامج بعد الإشارة الى ملف المكتبة.

ملحوظة:
الملفات ذات الامتداد DLL عبارة عن ملفات مكتبات يتم إنشائها منفصلة ليصلح ربطها بأكثر من برنامج مما يوفر عدم إعادة كتابة سطور محتوياتها أكثر من مرة بل تكتب مرة واحدة ثم تستعمل أكثر من مرة.
وسميت Dynamic لعدم ربطها ربطا نهائيا بالبرنامج بل تربط مؤقتا فقط أثناء التشغيل.

إنشاء مكتبة فصائل Class library

تابع معنا الخطوات التالية :

1- شغل vb.net ثم انقر (اختر) new project ثم اختر visual basic projects ومن الجانب الأيمن اختر class library ثم اكتب اسماً my class ثم انقر ok كما فى الشكل (9-9) .



الشكل(9-9)

بهذا تم إنشاء مكتبة فصائل class Lib بالاسم ClassLibrary1 وتم تعريف ذلك في ملف بالامتداد VB.

2-قم بتفحص الأوامر الناتجة تلاحظ إنشاء فصيلة بالاسم class1 ليس بها أي شيء ويمكن تعريف أكثر من فصيلة CLASS بهذا الملف ويمكن تجميع أكثر من ملف من هذه الملفات (ملفات الفصائل) في مجموعة تسمى Assembly .

ويمكن إنشاء فصيلة class بهذا الملف في أي وقت بكتابة ما يلي :

```
class  class_name
.....
end class
```

3- غير اسم الفصيلة class باستبدال السطر class class1 بالسطر class student

4- غير اسم الفصيلة في solution وذلك باختياره بالطرف الأيمن وتغير اسمه إلى student.

بهذا تم إنشاء فصيلة class مع ملاحظة إمكانية إنشاء أي فصيلة في أي وقت داخل هذا الملف بكتابة كلمة class وبعدها اسم الفصيلة في نهاية السطور تغلق الفصيلة بالعبارة end class كما سبق.

5- قم بتعريف متغيرات var وخصائص properties الفصيلة student كما في السطور التالية:

```
Public Class student
    Dim student_no As Integer
    Dim student_name As String
    Public Property stno()
        Get
            Return student_no
        End Get
        Set(ByVal Value)
            student_no = Value
        End Set
    End Property
    Public Property stname()
        Get
            Return student_name
        End Get
        Set(ByVal Value)
            student_name = Value
        End Set
    End Property
End Class
```

فى لغة C#

```
public class person
{
public string Pname, Paddress;
public void  SetData( string name_v  ,   string address_v  )
{
Pname = name_v;
Paddress = address_v;
}
public string  getName()
{
return Pname;
}
public string  getAddress()
{
return Paddress;
}
}

public class student: person
{
int degree;
public void  setdegree(int  degree_v)
{
degree = degree_v;
}
```

```
public int  retdegree()
{
return degree;
}

}
```

راجع هذه السطور تلاحظ تعريف متغيرين وتعريف خصائص التعامل مع المتغيرين كما سبق.

6- من القائمة الرئيسية اختر Build ثم Build ClassLibrary1 تلاحظ قيام Visual Basic.Net بترجمة ملف المكتبة والمفترض أننا نحصل على ملف بالاسم Classlibrary1.dll وللتأكد من ذلك من فضلك اذهب الى فهرس هذا المثال حسب مكان حفظك للامثلة تلاحظ وجود الملف داخل الفهرس bin ويمكنك مشاهدة ذلك بمحاولة حفظ أحد ملفات التطبيق كما بالشكل (10-9) .

الشكل (10-9)

إستعمال مكتبة الفصائل Using class library

بعد إنشاء مكتبة الفصائل ClassLibrary1 وإنشاء فصيلة class بالاسم student داخل هذه المكتبة وبعد القيام بترجمة ملف المكتبة الى ClassLibray1.DLL المطلوب معرفة كيفية استعمال هذه المكتبة، ولتوضيح ذلك تابع الخطوات التالية:

1- قم بإنشاء تطبيق جديد من النوع ConsoleApplication كما سبق.

2- من القائمة الرئيسية أختر Project ثم Add Refrence للإشارة إلى ملف المكتبة classLibrary1.Dll كما فى الشكل (11-9).

الشكل(11-9)

3- يظهر مربع حوار Add Refrence به ثلاث صفحات (tabs) اختر الصفحة Projects ثم اضغط الزر Browse واختر ملف المكتبة ClassLibrary1.DLL كما فى الشكل (12-9) .

الشكل(12-9)

4- اختر الملف ClassLibrary1.dll ثم اضغط الزر Open تعود إلى بيئة التطوير وقد تم الإشارة إلى ملف المكتبة ويمكن استعمال فصائلها.

5- قم بتعريف هدف Object من الفصيلة student داخل الدالة الرئيسية للبرنامج Sub Main() كما في الشكل (13-9).

```
Sub Main()
    Dim a As New ClassLibrary1.
                                student

End Sub
```

الشكل(13-9)

في لغة C#

```
14        static void Main(string[] args)
15        {
16            //
17            // TODO: Add code to start application here
18            //
19            ClassLibrary1.
20        }                    person
21    }                        student
```

6- قم باستكمال سطور استعمال الفصيلة student وذلك كما في السطور التالية:

```
Sub Main()
    Dim a As New ClassLibrary1.student()
    Dim v1 As Integer
    Dim v2 As String
    a.stname = "OMR"
    a.stno = 125
    v1 = a.stno
    v2 = a.stname
    Console.WriteLine(v1)
    Console.WriteLine(v2)

End Sub
```

فى لغة C#

```
14        static void Main(string[] args)
15        {
16
17            ClassLibrary1.person p1=new ClassLibrary1.person ();
18            p1.SetData ("Omr","Cairo");
19            Console.WriteLine (p1.getName());
20            Console.WriteLine (p1.getAddress());
21
22        }
```

7- قم بتنفيذ البرنامج تحصل على القيم التى تم تحديدها وتم طباعتها باستعمال الدالة WriteIn() وبهذا تكتمل خطوات إنشاء واستعمال مكتبة الفصائل Class Library.

إضافة خصائص creating proprities

نعود إلى الفصيلة السابقة ونرى كيفية إضافة خصائص proprities ويتم ذلك بتعريف متغيرات و تحديد الدوال التي تتعامل معها كما يلي :

قم بتعريف متغير داخل سطور الفصيلة كما يلي :

```
dim  m first name as string
```

يعد تعريف المتغير الذي يمثل الخاصية وهو m first name .

قم بإضافة الدالة ( )get لقراءة قيمة الخاصية (المتغير) وكذلك إضافة الدالة( )set لتسجيل قيمة الخاصية (المتغير) كما يلي:

```
1: public property first name as  string
2: Get
3: return m first name
4: end Get
5: set(by val   value as string)
6: m first name = value
7: end set
8: end property
```

في هذه السطور :

اختيار الفصيلة test the class

بعد إضافة فصيلة بالاسم student وتعريف خاصية first name وتم إنشائها إنشاء

دالة لتسجيلها تابع معنا كيفية إنشاء تطبيق لاختبار (استعمال)هذه الفصيلة كما يأتي :

1- اختر قائمة file ثم new ثم project اختر windows apps وأعطيه الاسم test class وتأكد من اختيار أداة الاختيار add to solution وهذا ما يقابل group في vb6 .

2- يتم إضافة التطبيق الجديد إلى التطبيق القديم (class) ويظهرا معاً في نافذة soluation explor.

- انقر تطبيق لاختبار test class بالطرف الأيمن واختر set as start up project من القائمة التي تظهر .
- من نافذة soluation explor لنقر refrance بالطرف الأيمن في برنامج test class يظهر اسم الفصيلة السابق اختارها .
- عد إلى تطبيق test class وأخف زر أمر buttons علية ثم انقرها مرتين للدخول إلى صفحة الأوامر و اكتب السطور التالية:

```
1- Dim NC AS new  my class . student( )
2- NCO FIRST NAME ="OMR"
3- MSG (NC . first name )
4- end sub
```

في هذه السطور:

- في السطر رقم 1 يتم تعريف المتغير NC كهدف جديد للفصيلة Student المعرفة في المكتبة my class.
- في السطر رقم 2 تم تسجيل الخاصية first name للهدف الجديد NC المأخوذ من الفصيلة STUDENT وذلك لتسجيل العبارة الحرفية OMR .
- في السطر رقم 3 يتم عرض قيمة الخاصية first name بعد تسجيلها لاحظ الكتابة المكتوبة الموجود بها الفصيلة student هي my class وبدون كتابة اسم المكتبة تظهر رسالة خطأ تعني عدم التعرف على الفصيلة student ويمكن التعويض عن اسم المكتبة باستعمال الأمر import كما يلي :

```
1- import my class
2- dim NC as new student
3- NC .first name ="OMR"
4- MSGBOX (NC .first name)
```

في هذه السطور:

- في السطر رقم 1 استيراد المكتبة my class باستعمال الأمر import .
- في السطر رقم 2 تم تعريف المتغير NC من نوع الفصيلة student وفي هذه الحالة لم نحتاج لكتابة اسم المكتبة my class وباقي السطور كما سبق .

إنشاء خصائص للقراءة فقط أو للكتابة فقط read-only and write-only

في بعض الأحيان نحتاج للإنشاء الخاصية للقراءة فقط بحيث يمكن معرفة قيمة الخاصية ولا نسجلها و أحياناً نحتاج للعكس .

ولإنهاء خاصية read-only يتم عبارة read-only أمام الخاصية وكذلك يتم حذف دالة set التي تكتب في الخاصية ويتم ذلك فيما يلي:

```
1: public read only property first name ( ) as string
2:  Get
3: return m first name
4: end get
5: end property
```

في هذه السطور:

- في السطر رقم 1 تم وضع العبارة read-only للإشارة أن هذه الخاصية للقراءة فقط .
- ومن السطر رقم 2 إلى السطر رقم 4 تم إنشاء دالة Get التي تقرأ قيمة الخاصية وتم حذف دالة set وبالتالي أصبحت الخاصية للقراءه ولا يسمح بكتابة قيمه بها .
- بالمثل يمكن عمل العكس وذلك بإنشاء خاصية للكتابة وليس للقراءة وذلك بكتابة دالة set و إلغاء دالة Get وكذلك كتابة العبارة write-only.

إضافة دالة بناء للفصيلة add constructor

كما شرحنا من قبل أن دالة البناء constructor يتم إنشائها في الفصيلة لتحقيق العمليات الأولية فمثلاً في الفصيلة student التي أنشأناها نريد إرسال كود الطالب عند تعريف الهدف (متغير) من الفصيلة يتم ذلك بإنشاء دالة CONSTRUCTOR وفي لغةVBيتم إنشاء دالة البناء بإنشاء دالة بالاسم NEW( ) كما في السطور التالية:

```
Public sub new (optional by val sid as intege =a)
Stno =sid
End sub
```

في هذه السطور يتم إنشاء دالة البناء بالاسم ( )sub new وفيها يتم تسجيل قيمة المتغير stno المفترض تعريفة بالفصيلة وذلك بقيمة المتغير sid وبالتالي عند التعامل مع الفصيلة وتعريف متغير (هدف) يتم ذلك كما يلي :

```
dim NS as new student (1)
```

في هذه الحالة تم تعريف المتغير NS مع إرسال القيمة 1 كمعامل وبالتالي تأخذه دالة البناء وتسجله في المتغير STNO لمجرد تعريف المتغير .

إضافة دوال أعضاء للفصيلة Adding methods

كما شرحنا أن الفصيلة تتكون من بيانات (خصائص) ودوال أعضاء methods التي تحدد عمليات الفصيلة ، ولدوال الأعضاء مثل أي دالة يتم إنشائها داخل الفصيلة لتحقيق عمليه معينه مع الفصيلة .

ويتضح ذلك من السطور التالية:

- قم بكتابة الدالة التالية داخل سطور الفصيلة .

```
public function add student ( ) as BOO learn
  add new student to data basic
return true
end function
```

- بمجرد كتابة هذه الدالة داخل سطور الفصيلة تصبح دالة عضوه في الفصيلة وبالتالي استعمال واستدعائها تسجيل بيانات طالب ويتم استدعائها بالصور التالية:

```
dim NS AS NEW STUDENT( )
NS .Add student   ( )
```

الفصل العاشر

معالجة الأخطاء
Error Handling

Programming Concepts 10

فى هذا الفصل

في هذا الفصل نعرض موضوع الأخطاء الوارد حدوثها مع المبرمج ومع مستخدم البرنامج وذلك من خلال النقاط التالية:

- الأنواع المختلفة للأخطاء : Bugs
- الخطأ الهجائي Syntax error
- الخطأ أثناء التشغيل Runtime Error
- ON ERROR GOTO-
- ON ERROR RESUME NEXT-
- الخطأ المنطقي Logic Error
- طرق حل الأخطاء :
- طرق تلافى خطأ أثناء التشغيل
- طرق تتبع الخطأ المنطقي ومعرفة سببه
- استعمال الشاشات المختلفة
- استعمال التركيب الجديد Try.....Catch

عند القيام بإعداد برنامج يصادفك أخطاء ما أثناء الإعداد أو بعد توزيع البرنامج للاستخدام ، ومن عوامل نجاح البرنامج عدم ظهور أي أخطاء ، وفيما يلي نوضح أنواع الأخطاء وطرق تلافيها .

أنواع الأخطاء :

1 ـ الخطأ الهجائي Syntax Error

وهو أبسط أنواع الأخطاء حيث يظهر عند كتابة نصوص البرنامج وهو خطأ في الحروف الهجائية للأوامر المعرفة للغة أو خطأ في قواعد اللغة فمثلاً عند كتابة جملة For بدون Next أو العكس.

- وعند كتابة جملة If بدون End If.
- عند كتابة أمر MSGBOX ناقص حرف وهكذا.
- وهذا الخطأ يظهر سريعاً ويسهل علاجه.

ولتوضيح ذلك سوف نقوم بشرح مثال توضح فيه النوع الاول من الأخطاء SYNTAX ERROR كما يلي:

مثال :

1. ابدأ برنامج جديد بالخطوات المعتادة من النوع Windows Application .
2. وقع زر أمر Button على الـ FORM كما هو معتاد.
3. اضغط بالماوس على زر الأمر Button1 مرتين لتحصل على دالة Click الخاصة به واكتب بها السطور التالية :

```
Dim i As Integer
Dim a(10) As Integer
For i = 0 To 40
a(i) = I
```

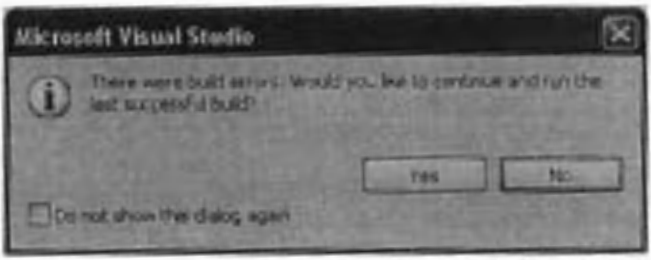
فى لغة C#

```
89        private void Button1_Click(object sender, System.EventArgs e)
90        {
91            int i;
92            for (i=0;i<5)
93                MessageBox.Show (i+"");
94        }
```

فى لغة Java

```
1  public class Test1 {
2
3      public static void main(String[] args) {
4      System.out.print("test the Syntax Error ");
5      }
6  }
```

نفذ البرنامج (F5) تلاحظ ظهور رسالة خطأ كما بالشكل (1-10).

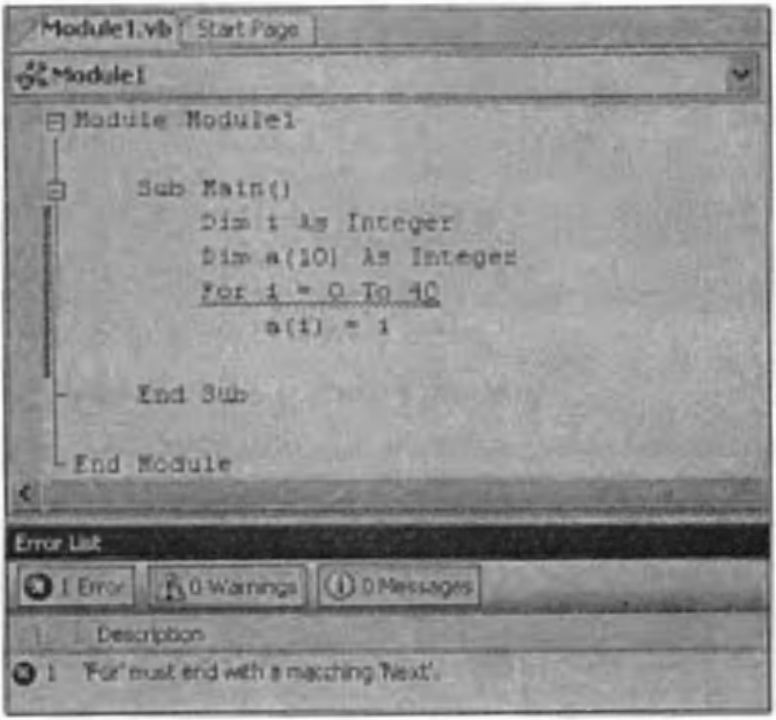


الشكل(1-10)

فى لغة Java

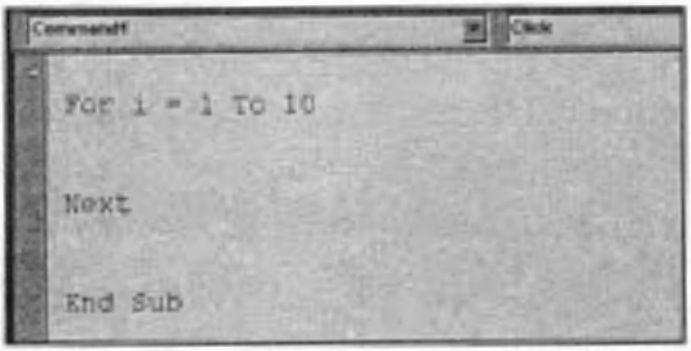
تظهر رسالة الخطأ كما فى الشكل:

1. اضغط زر No تلاحظ وجود رسالة الخطأ فى مربع Task Lists أسفل الشاشة اضغط الرسالة مرتين بالماوس تدخل الى محرر السطور مكان الخطأ اكتب كلمة NEXT I كما بالشكل (2-10).



الشكل(2-10)

نفذ البرنامج F5 يتم تنفيذ البرنامج بعد علاج الخطأ واختفاء رسالة الخطأ، وهذا هو النوع الأول من الأخطاء وهذه هى طريقة علاجه ، يظهر مع أى خطأ فى قواعد اللغة ويسهل تجنب أو تقليل هذه الأخطاء باستعمال القاعدة التى تظهر عند كتابة الأمر فمثلا عند كتابة الدالة MSGBOX يظهر تركيب الدالة بالصورة الصحيحة كما فى الشكل (3-10) .



الشكل(3-10)

فى لغة C#

قم بتصحيح الخطأ ليصبح كما فى السطور.

```
89          private void button1_Click(object sender, System.EventArgs e)
90          {
91              int i;
92              for (i=0;i<5;i++)
93                  MessageBox.Show (i+"");
94          }
```

و المقصود بهذا الشكل أننا كتبنا الأمر بهيكله الصحيح أولاً.

النوع الثانى : الخطأ أثناء التشغيل : Runtime Error

لا يظهر هذا الخطأ أثناء أعداد البرنامج الا إذا قمت بتجربة البرنامج بينما يظهر للمستخدم نتيجة قيام المبرمج بكتابة أوامر تحقق عملية غير محققة أثناء التنفيذ فمثلاً:

أ-عند صدور أمر فتح ملف من مشغل أقراص (A,B,...) والمشغل غير جاهز تظهر رسالة خطأ RunTime Error ويتقطع البرنامج ويعود الى بيئة التشغيل

ب - عند صدور أمر فتح قاعدة بيانات غير موجودة وهكذا ويعتبر هذا النوع من الأخطاء التى تعيب البرنامج فعندما يظهر للمستخدم الخطأ يقطع البرنامج ويخرج وتعتبر هذه عيوب فى البرنامج ، وبالتالى يجب تلافى هذا الخطأ والا ينقطع العمل فى البرنامج فمثلاً يمكن التأكد من وجود القرص قبل تنفيذ أمر فتح الملف أو وجود قاعدة البيانات أو غيره ومن أشهر هذه الأخطاء القسمة على القيمة صفر Divid By Zero حيث تعطى خطأ Over Flow ولتوضيح ذلك تابع المثال التالى:

1-وقع زر أمر Button واكتب به السطور التالية:

```
Dim d, i, r As Integer
    d = 100
    i = 0
    r = d / i
    MsgBox(r)
```

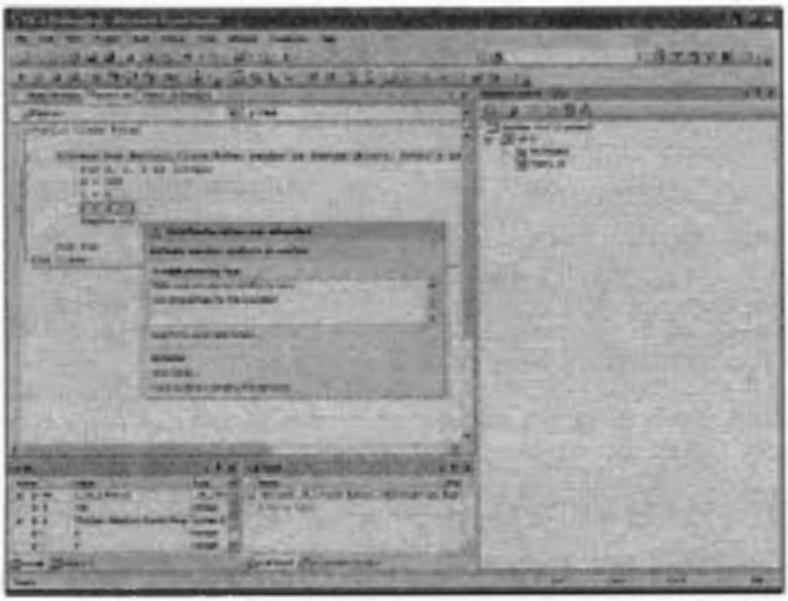
فى لغة C#

```
89          private void button1_Click(object sender, System.EventArgs e)
90          {
91              int d1=100;
92              int d2=0;
93                  int result = d1/d2;
94          }
```

فى لغة Java

```
1  public class Test1 {
2
3      public static void main(String[] args) {
4      int k;
5      k=100/0;
6      }
7  }
```

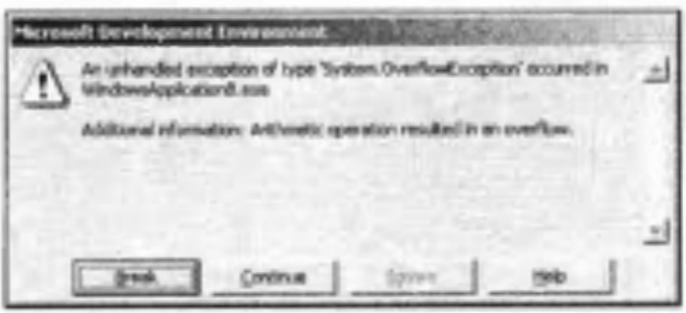
- فى هذه السطور يتم القسمة على المتغير I الذى يساوى 0 نفذ البرنامج واضغط زر الأمر Button1 تحصل على خطأ كما فى الشكل (4-10).



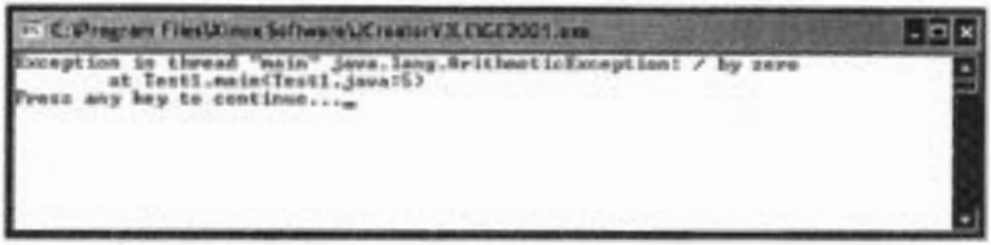
الشكل(4-10)

فى C#

تظهر رسالة الخطأ التالية



فى لغة Java



طرق علاج الخطأ

توجد أكثر من طريقة لعلاج هذا النوع من الأخطاء حسب أهمية العملية وهى :

1- استعمال الجملة ON Error Resume Next.

2- استعمال الجملة ON Error GOTO.

3- استعمال مصفوفة الأخطاء.

4- استعمال التركيب الجديد Try.....Catch.

فى Vb.Net

1. الجملة On Error Resume Next

تستخدم هذه الجملة لتلاشى الخطأ وتكتب لكى تجاهل تنفيذ الجملة التى تسبب الخطأ والانتقال الى الجملة التالية و لن يتم تنفيذ الجملة ولا الاشارة الى عدم تنفيذها ويتم استعمالها عندما تريد ذلك وذلك عندما لا توجد أهمية لإظهار رسالة الخطأ.

ولتوضيح ذلك نعود الى البرنامج السابق ونجرب استعمال هذه الجملة كما يلى :

1- افتح المثال السابق وأعد كتابة السطور لتصبح كما يلى :

```
On Error Resume Next
   Dim d, i, r As Integer
d = 100
i = 0
r = d / i
MsgBox(r)
```

2- نفذ البرنامج كما سبق.

3- اضغط الزر Button1 كما سبق تلاحظ عدم ظهور رسالة الخطأ ولكن عدم ربط ذلك بالخطأ وعدم ظهور ما يفيد ذلك.

2. الجملة ON ERROR GO TO

تستخدم هذه الجملة لتوجيه تنفيذ البرنامج الى مجموعة سطور تظهر رسالة الخطأ وتتولى علاجه وهى أفضل من الطريقة السابقة حيث تظهر رسالة مفيدة توضح موقف البرنامج ولتوضيح ذلك أعد فتح البرنامج السابق وتابع ما يلى :

اضغط أداة Button1 مرتين وأعد كتابة السطور لتصبح كما بالشكل (5-10) :

```
Dim lab As Label
On Error GoTo lab
Dim d, i, r As Integer
d = 100
i = 0
r = d / i
MsgBox(r)

lab:
MsgBox(Err.Description)
```

الشكل(5-10)

فى هذه السطور :

- فى السطر رقم 2 تم استعمال العبارة ON ERROR GOTO lab بمعنى إذا حدث خطأ فى السطور التالية ووجه التنفيذ إلى الخطوة التى يوجد أمامها العنوان lab .
- فى السطر 3 الأمر exit sub للخروج من الامر ما لم يصل إلى الخطأ.
- فى السطر رقم 4 العنوان lab لبداية جمل معالجة الأخطاء.
- فى السطر 5 يتم عرض رسالة الخطأ الهدف ثقق الذى يشير الى الخطأ والرسالة يتم عرضها بالخاصية DESCRIPTION.
- فى السطر رقم 6 يتم عرض رقم الخطأ.
- ولكن الرسالة تظهر باللغة الإنجليزية ولاظهارها باللغة العربية أعد كتابة السطور لتصبح كما بالشكل (6-10) :

```
59          Dim lab As Label
60          On Error GoTo lab
61          Dim d, i, r As Integer
62          d = 100
63          i = 0
64          r = d / i
65          MsgBox(r)
66      lab:
67          ' MsgBox(Err.Number)
68          If Err.Number = 6 Then
69              MsgBox("تم القسمة على صفر")
70          End If
71      End Sub
```

الشكل(6-10)

فى هذه السطور نختبر رقم الخطأ اذا كان 6 وهو رقم الخطأ الذى يعنى القسمة على صفر يتم عرض رسالة "تم القسمة على صفر".

ولكن بهذا الأسلوب لابد أن نختبر أرقام الأخطاء لتلافى الأخطاء المتوقعة لذلك يفضل استخدام الحل الثالث وهو مصفوفة الأخطاء.

استعمال التركيب الجديد Try.....Catch

هذا الأسلوب يعتبر جديد فى الاصدار Vb.Net ولكنه كان موجود فى اللغة VC++ وكذلك لغة Java ويأخذ الشكل.

```
Try
Statment1
Statment2

Catch Exception1
Action1

Catch Exception2
Action2

Catch Exception3
Action3
End Try
```

- فى هذه السطور نبدأ بالكلمة Try بمعنى حاول تنفيذ الجمل التالية وهى Statment1,Statment2 فاذا تم تنفيذهم بنجاح اذهب الى End Try وتابع البرنامج
- واذا ظهرت مشاكل أذهب الى مجموعة جمل Catch (أى امسك) التى تتعامل مع الخطأ الناتج.
- فاذا كان الخطأ الناتج من النوع Exception1 نفذ Action1 و كان الخطأ الناتج من النوع Exception2 نفذ Action2.

وهكذا ولتوضيح ذلك تابع الخطوات التالية:

فى Vb.NET

1- فى أول البرنامج قم بكتابة السطر التالى

```
Imports System.IO
```

فى هذا السطر تم استعمال الأمر Imports للإشارة إلى الـ IO NameSpace

2-وقع زر أمر Button واكتب بداله السطور التالية كما بالشكل (8-10).

```
62          Dim filest As FileStream
63
64          Try
65              filest = New FileStream("a:\data.txt", FileMode.Open)
66          Catch ex As FileNotFoundException
67              MsgBox("The file could not be found.")
68              Exit Sub
69          Catch ex As IOException
70              MsgBox("An error occurred: " & ex.Message)
71              Exit Sub
72          End Try
73
```

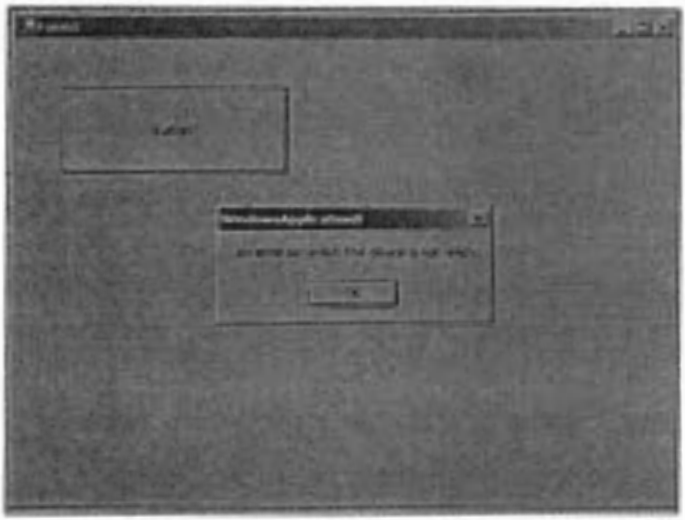
الشكل(8-10)

فى هذه السطور:

يتم تعرف متغير كم النوع filestream لتتمكن من فتح ملف والتعامل معه ثم محاولة فتح هذا الملف من مشغل الديسكات A فإذا كان الديسك غير موجود يتحول الخطأ إلى IOException وتظهر الرسالة الخاصة بهذ الخطأ.

وإذا كان الديسك موجود ولكن الملف غير موجود بالديسك يتحول الخطأ إلى FileNotFoundException وتظهر الرسالة الخاصة بهذا الخطأ.

3-نفذ البرامج واضغط زر الأمر Button1 يحاول الكمبيوتر فتح الملف data.txt من مشغل الديسكات A ثم تظهر رسالة خطأ كما فى الشكل (9-10).



الشكل(9-10)

النوع الثالث : الخطأ المنطقى Logical Error

لا يعطى هذا النوع أى رسائل خطأ بل أن البرنامج قد يعمل جيداً بدون مشاكل مع ظهور الرسائل المعبرة وكل شئ سليم ولكن الخطأ فى نتيجة البرنامج :

فمثلاً : عند قيام برنامج المرتبات بطباعة صافى مرتب كل موظف يطبع المرتب خطأ بالرغم من تحقق جميع العلميات ويرجع هذا النوع إلى منطق البرنامج وخط السير، فقد يكون معد البرنامج وضع علامة الطرح مكان علامة الجمع أو ما شابه ذلك أو وضع قاعدة خطأ باستعمال جملة If وغير ذلك .

وفى حالة بساطة البرنامج يسهل اكتشاف الخطأ بتتبع سطور البرنامج وتحديد الأخطاء ولكن فى حالة البرامج الكبيرة يصعب تتبع سطور البرنامج لذلك توجد طرق لإكتشاف الخطأ المنطقى.

طرق اكتشاف الخطأ المنطقى

وتنقسم هذه الطرق الى مجموعتين:

أ - التحكم فى سير البرنامج لمعرفة السطر الذى به الخطأ.

ب - إظهار معلومات عن متغيرات أو تعبيرات فى نوافذ خاصة.

(أ) التحكم فى سير البرنامج

ولتحقيق ذلك استخدم شريط أدوات Debug كما يلى :

1- قم باظهار شريط أدوات Debug وذلك بعمل نقرة يالطرف الأيمن للـ mouse أثناء وضعه فوق شريط القائمة تظهر قائمة اختار منها debug كما بالشكل (10-10).

الشكل(10-10)

2- هذه القائمة توفر أدوات تسهل عملية متابعة سير البرنامج

جرب استعمال هذه المفاتيح من خلال برنامج كما يلى :

استعمال نقاط التوقف Break Points

تعتبر فكرة اضافة نقاط توقف Break Points فى سطور البرنامج بحيث يتوقف تنفيذ البرنامج عنده لمشاهدة قيم المتغيرات ومتابعة البرنامج عندها من أشهر وأقدم طرق اكتشاف الخطأ المنطقى.

ويتم ذلك باختيار السطر الذى ترغب فى وضع نقطة توقف Break Point عنده ثم الضغط على المفاتيح Ctrl+B.

ثم تنفيذ البرنامج تلاحظ أنه يتوقف عند نقطة التوقف كلما ضغط على F5.

الاستفادة من الـ Snippet

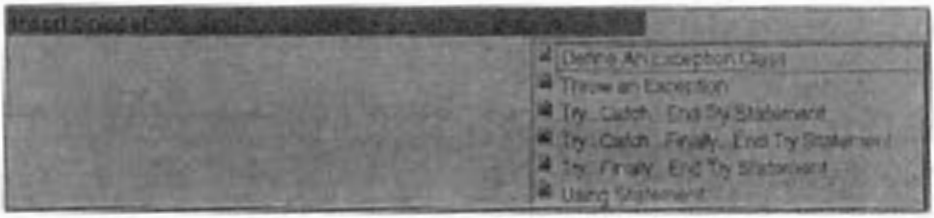
1- توفر قطع الكود الجاهزة Snippet بعض العمليات المفيدة مع أخطاء البرنامج كما فى الشكل (11-10).

الشكل(11-10)

2- وعند اختيار Common Code Patterns تحصل على قائمة أخرى كما فى الشكل (12-10).

الشكل(12-10)

3- اختر Exception Handling تحصل على مجموعة من الاكواد كما فى الشكل (13-10).



الشكل(13-10)

وفيما يلى عرض بعض هذه الأكواد.

```
Try
    Catch ex As ApplicationException
    End Try
```

فى هذه السطور:

- يتم عرض الشكل التقليدى للتركيب Try...Catch.. Enf Try كما عرضناه.

اضافة الامر Finally

```
    Try
    Catch ex As ApplicationException
    Finally
    End Try
```

وهكذا باقى قطع الكود.

أدوات معالجة الإستثناءات فى لغة Java

معذرة ربما أطلنا عليك بتناول هذا الموضوع بدون أمثلة كثيرة .. ولكن هذا الموضوع مهم ، حيث لا يمكن أن يتجاهله مبرمج، وإدارة معالجة الإستثناءات تتم من خلال خمسة كلمات محجوزة Keywords وهى الكلمات.

```
finally, try , catch , throw, throws
```

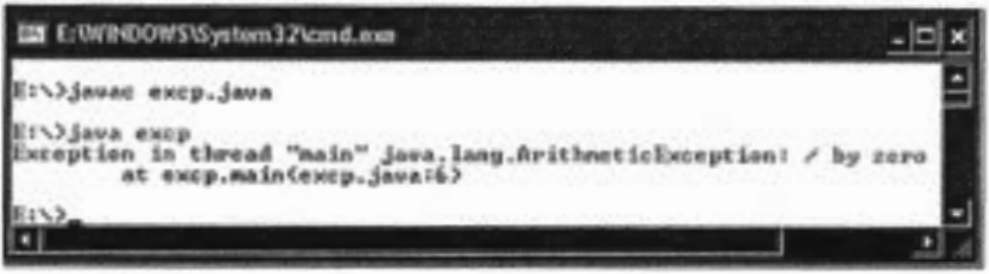
فتعال معى نرى كيفية استخدامها.

التركيب Try ------ catch

أول أدوات معالجة الإستثناءات هم التركيب try ------ catch وهما من أسهل الأدوات ولكى نرى كيفية عملها فلنلقى نظرة على نص البرنامج التالى:

```
1.  class Excp
2.  {
3.  public static void main (String Arg[ ])
4.  {
5.  int d= 0;
6.  int a=60 /d;
7.  System.out.println("After Exception");
8.      {
9.  {
10.
```

لو حاولت ترجمة ثم تنفيذ هذا المثال ستجد أن المترجم قد اظهر الرسالة التالية:



من الواضح أن البرنامج قد ترجم Compile بنجاح ولكن عند التنفيذ اظهر هذه الرسالة وهذه الرسالة تخبرنا بالمعلومات التالية:-

- هناك خطأ ظهر أثناء وقت التنفيذ ولاكتشاف الخطأ نقرأ الرسالة فنجد انه حدث استثناء Exception وهذه أول كلمة فى الرسالة لكن أين حدث هذا الاستثناء؟ نجد أن الرسالة تقول أنة فى Thread main اى انه حدث خطأ عند تنفيذ الدالة الرئيسية main ولكن ما نوع هذا الاستثناء نجد انه من نوع Arithmetic وقد نتج عن محاولتنا القسمة على صفر وهذا ناتج عن الجملة رقم 6 ونجد أن الرسالة تدلنا على اسم الفصيلة التى حدث بها الاستثناء ورقم السطر الذى حدث به الاستثناء كما هو واضح بين الأقواس.

من كل ما سبق نستخلص أن الدالة main حدث بها استثناء Exception وقد ألقت الاستثناء إلى نظام وقت التنفيذ Runtime system وقام نظام وقت التنفيذ بالبحث عن من يعالج هذا الاستثناء فلم يجد فألقى الاستثناء ثم خرج من النظام . وستجد أيضا أنه لم يصل إلى السطر 7 لأنه توقف عند الاستثناء الحادث.

ولمعالجة هذا الاستثناء تعالى معى ننظر إلى نسخه معدله من البرنامج فى الجزء التالى:

```
2.  class Excp
3.  {
4.  public static void main (String [] arg)
5.  {
6.  try{
7.  int d=0;
```

```
8.  int a;
9.  a =60/d
10.  {catch (ArithmeticException  e)
11. }
12. System.out.println("division by zero");
13. System.out.println(e.getMessage());

14. }
15. System.out.println("After Exception");
16. }
```

- حاول تنفيذ البرنامج الآن ستجد أن تم التنفيذ بدون مشاكل وستجد ان السطر 14 قد تم تنفيذه فماذا حدث.
- نجد أن الذى حدث أن الدالة أخبرت المترجم بأن يحاول تنفيذ الكود فى المنشأ try Block وإذا حدث استثناء وهو ما حدث يتم التقاطة عند طريق المنشأ catch Block وتم إظهار الرسالة التى تفيد حدوث القسمة على صفر.
- واخيراً تم ظهور نتيجة السطر 14 لأن الاستثناء الحادث قد تم إلتقاطة ومعالجته.
- الصورة العامة try –catch كالأتى:

```
try {
 // الكود الذى يحدث استثناء
}catch(Exception Object Declaration)
{
}
```

تعدد الكلمة catch

فى بعض الأحيان قد ينتج عن كود ما أكثر من استثناء فما الحل فى هذه الحالة؟

الحل أن يوجد العديد من معالج الاستثناء Exception Handler وفى حالة استخدام try catch ---- تكون catch هى معالج الاستثناء وبالتالى لابد من وجود أكثر من بلوك catch ويصبح تركيب البرنامج كما فى السطور التالية:

```
try {
// Code that might generate exceptions
} catch(Type1 id1) {
// Handle exceptions of Type1
} catch(Type2 id2) {
// Handle exceptions of Type2
} catch(Type3 id3) {
// Handle exceptions of Type3
```

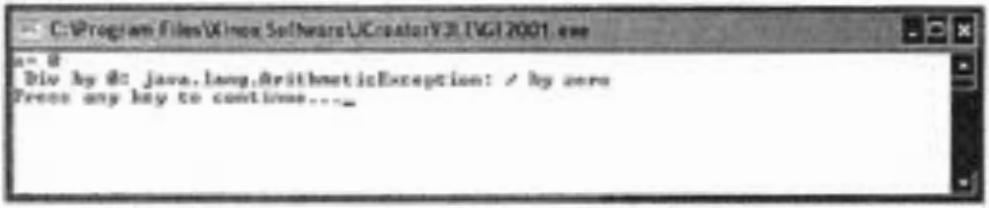
ولتوضيح ذلك تابع معنا سطور البرنامج التالية :

```
1  public class Test1 {
2
3      public static void main(String[] args) {
4
5  try {
6  int a= args.length;
7  System.out.println("a= " + a);
8  int b = 42/a;
9  int c [] = {1};
10 c [42 ] = 99;}
11 catch (ArithmeticException e)
12 {
13 System.out.println(" Div by 0: "+e);
14 }
15 catch (ArrayIndexOutOfBoundsException e)
16 {
17 System.out.println(" array  index Out of Bounds "+e);
18 }
19
20     }
```

- في هذا المثال ينتج استثناءان.
- الأول ناتج عن القسم على صفر وذلك كما في السطر 7 إذا بدأنا البرنامج بدون إعطاؤه أي معاملات.
- والثاني ناتج عن محاولة تخطي سعه المصفوفة c فقد أعلنا أن سعتها عنصر واحد وذلك كما في السطر 8 ثم في السطر 9 نحاول تخصيص قيمة للعنصر 43 بينما المصفوفة سعتها عنصر واحد فقط.
- لذلك قمنا بإعداد اثنان من معالجي الاستثناء Exception Handler وذلك باستخدام اثنان من المنشئات Block من catch.
- المنشأ Block الأول يعالج الاستثناء الناتج عن محاولة القسمة على صفر وذلك كما يتضح في السطور من 11 إلى 14 .

- المنشأ Block الثانى يعالج الاستثناء الناتج عن عن محاولة تخطى سعه المصفوفة c وذلك كما يتضح فى السطور من 15-18.

ويظهر ذلك من نتيجة التنفيذ كما فى الشكل.



معالجة الاستثناءات عن طريق الكلمة المحجوزة Throws

فى بعض الأحيان عندما نقوم بتصميم دالة Method فى فصيلة ما ونعرف أن هذه الدالة ستلقى باستثناء فمن المستحب ان تجعل مستخدم الدالة هو الذى يقوم بمعالجة الاستثناء وللتوضيح أكثر تعال معى نرى نص البرنامج التالى.

```
1.  class ThrowsDemo {
2.  static void delta() throws IllegalAccessException
3.  {
4.  System.out.println("inside delta");
5.  throw new IllegalAccessException ("demo");
6.  }
7.   public static void main (String args[ ])
8.   {
9.        try {
10.             delta();
11. }
12. catch (IllegalAccessException e)
13.{
14.       System.out.println(" caught" + e);
15.   }
16.  }
17.}
```

- فى هذا المثال قمنا بتصميم الدالة delta() وهذه الدالة نعرف إنها ستلقى باستثناء وفى هذه الحالة لابد من تحديد الاستثناء الذى ستلقيه الدالة حتى تستطيع الدوال الأخرى

عند استدعاء الدالة delta() حماية نفسها بتوفير معالج استثناء من نوع الاستثناء والذى ستلقيه delta() .

- والدالة delta ستلقى الاستثناء بالكلمة المحجوزة throws ثم نوع الاستثناء وذلك كما فى السطر 2 .
- والشكل العام لمثل هذه الدوال كالآتى:

Accesstype methodtype methodName (arg- list) throws Exception 1, Exception 2,....

وكمثال كما رأينا فى المثال السابق:

static void delta ( ) throws IllegalAccessException.

- ومن هذه الصورة العامه يتضح إننا نستطيع إرسال أكثر من استثناء.
- بعد السطر 2 جملة بناء الدالة Delta ولكن ما يهمنا هنا هو السطر رقم 5 وهذا السطر هو المسئول عن عملية إرسال الاستثناء ولاحظ أنه لو عندنا أكثر من استثناء لابد من إلقائهم بالأمر throw وليس throws.
- لاحظ أيضا أن الدالة انتهت عند السطر 6 ولم تهتم بمعالجة الاستثناء لأن هذا أصبح ليس مهمتها ولكن مهمة من يستدعى هذه الدالة وذلك كما فى السطور من 9-15.
- ففيالسطور 9-15 قامت الدالة main باستدعاء الدالة delta ولأن الدالة main تعرف ان الدالة delta ستلقى بإستثناء فقامت بحماية نفسها عند استدعاء الدالة delta بـ try-catch والتقطت الاستثناء وعالجته وذلك كما فى السطور من 11 -14.

معالجة الاستثناءات عن طريق الكلمة المحجوزة Throw

يستخدم الكلمة المحجوزة throw فى إلقاء استثناء كما أوضحنا فى المثال السابق ولكنه ليس بالضرورة أن يستخدم مع throws فيمكن أن يستخدم وحدة كما سنوضحه فى نص البرنامج التالى:

```
1.  class ThrowDemo {
2.        static void demo ()
3.        {
4.              try {
```

```
5.                throw new NullPointerException ("demo");
6.               } catch (NullPointerException e){
7.               System.out.println("cought inside demo");
8.               throw e ;
9.               }
10.       }
11.    public static void main (String arg[])
12.    {
13.         try {
14.            demo ();
15.            } catch (NullPointerException e)
16.            {
17.               System.out.println("Recaught : " + e);
18.            }
19.     }
20. }
```

- إن الكلمة المحجوزة throw يستخدم لإرسال الاستثناء بطريقة إجبارية.
- ولكى نستخدم throw لابد من إنشاء هدف من نوع الاستثناء الذى أود أن أرسله وذلك كما فى السطر 5 ثم لابد من وجود معالج للاستثناء Exception Handler وذلك كما يوضحه السطر 6 وذلك إذا أردت ان أعالج الاستثناء داخل الدالة.
- أما إذا أردت إرسال استثناء لتعالجه دالة أخرى عندما نستدعى الدالة المعلن فيها الاستثناء فلابد أن إرسال يهدف كما يوضحه السطر 8 والذى يستقبله معالج الاستثناء الموجود فى دالة الاستدعاء كما فى سطر 15.

معالجة الاستثناءات عن طريق الكلمة المحجوزة Finally

عندما يتم ارسال استثناء Exception فإن تسلسل flow التنفيذ فى الدالة Method يتم تغييره ويتم تحويل التسلسل للبحث عن أقرب معالج للاستثناء وفى حالة عدم وجود معالج للاستثناء مثل catch لا يتم تنفيذ الجمل الموجودة بعد الموقع الحادث به الاستثناء.

إذن ماذا نفعل فى حالة إذا أردنا أن يتم تنفيذ جمل أو قطعة من الكود بغض النظر عن ما إذا كان الاستثناء قد تم معالجته أم لا؟

فى هذه الحالة يبرز لنا الكلمة المحجوزة finally حيث أنه يتم تنفيذ الجمل التى بداخله بغض النظر عن ما حدث للاستثناء فحتى ولو لم يكن هناك معالج مثل catch لالتقاط

الاستثناء ومعالجته فسيتم تنفيذ الكود الموجود داخل finally ويتم تنفيذ الموجود داخل finally مباشرة بعد try Block ولتوضح الكلام أكثر فى نص البرنامج التالى:

```
1. class FinallyDemo {
2. static void procA() {
3. try {
4. System.out.println("inside proc A");
5. throw new RuntimeException ("demo");
6. }
7. finally {
8. System.out.println("proc A finally");
9. }
10.}
11. static void procB() {
12. try {
13. System.out.println(" inside proc B");
14. return;
15.}
16. finally {
17. System.out.println("proc B funally");
18.}
19.}
20. public static void main (String arg [ ])
21.{
22. try {
23. procA();
24.}
25. catch (Exception e){
26. procB();)
27.}
28.}
```

- فى هذا المثال على الرغم من إلقاء الدالة procA() باستثناء كما يوضحه السطر 5 إلا أنه لا يوجد معالج للاستثناء فلا يوجد لدينا catch فكان من الطبيعى أن يتوقف عمل البرنامج ولكنه المترجم يجد أمامه finally فيضطر إلى إكمال التسلسل وتنفيذ البرنامج بدون مشاكل وذلك كما فى السطور 6-8.
- ثم نجدد شيئاً آخر غريب وهو الدالة procB وخصوصاً السطر 12 والموجود به جملة return فوجود تلك الجملة كان يعنى أن يتم كسر التسلسل والخروج ولكن لأن الدالة

بها finally فقد تم إجبار المترجم على تكملة التسلسل أما بقية البرنامج فسير الدالة طبيعى جدا حيث يتم استدعاء الدالة procA وذلك كما فى السطر 20 ولأننا نعرف أن الدالة الإرسال باستثناء فقد حمينا نفسنا منه بـ try --- catch.

- وفى السطر 22 استدعاء الدالة procB وحيث أن الدالة لا ترسل باستثناء فقد تم استدعاؤها بالطريقة المعتادة.

وعند التنفيذ نحصل على نتيجة التنفيذ كما فى الشكل.

ويصبح تركيب البرنامج كما فى الشكل.

```
try {
// The guarded region: Dangerous activities
// that might throw A, B, or C
} catch(A a1) {
// Handler for situation A
} catch(B b1) {
// Handler for situation B
} catch(C c1) {
// Handler for situation C
} finally {
// Activities that happen every time
```

إنشاء أنواع جديدة من الاستثناءات User Defined Exceptions

حتى الآن فإن كل أنواع الاستثناءات التى استخدمناها هى من الفصائل الموجودة فى اللغة وهذه الفصائل تغطى تقريباً كافة الاستثناءات الممكنة الحدوث ولكن ماذا نفعل إذا احتجنا استثناء غير موجود فى اللغة أى إذا كان الخطأ الذى يظهر غير معرف باللغة ؟

الحل هو إنشاء (تعريف) هذا الاستثناء فكيف يتم ذلك!؟

بما أن الاستثناءات فى الجافا هى فصائل Classes إذن فلنا الحق فى أن ننشىء فصائل ترث الفصائل الموجودة فى الجافا ونطيف إليها الجديد بما يناسب منطق البرنامج.

والفصيلة التى نقوم ببنائها لا تفعل أكثر من إصدار رسائل توضح أن هناك استثناء ما قد حدث وتبين نوعه وهذا للتوضيح فقط ولكن فى الواقع يتم إنشاء فصائل لعلاج أخطاء حقيقية بالبرنامج ويتضح ذها المثال من خلال السطور التالية :

```
1. class MyException extends Exception {
2. private int details;
3. MyException (int a)
4. {
5. detail=a;
6. }
7. public String toString()
8. {
9. return "My Exception [" + detail + " ]";
10. }
11. }
12. class UsingExceptionDemo
13. {
14. static void compute (int a) throws MyException
15. {
16. System.out.println("called compute (" + a + "). ");
17. if (a>15)
18. throw new MyException (a);
19. System.out.println("normal exit ");
20. }
21. public static void main (String args [])
22. {
23. try{
24. compute(1);
25. compute(20);
26. } catch (MyException e)
27. {System.out.println("caught" + e);}
28. }
29. }
```

فى هذه السطور:

- فى هذا المثال افترضنا أننا نريد أن ننشئ استثناء غير موجود.
- قمنا بإنشاء فصيلة وهذه الفصيلة ترث الفصيلة Exception وذلك كما فى السطر 1.
- فى هذه الفصيلة أردنا لو تم استدعاء دالة ما بقيمة أكبر من 15 فإنه يحدث استثناء وتظهر رسالة توضح قيمة الرقم فإنشاءنا Constructor يأخذ قيمة كما يوضحة السطر 3.

- ثم قمنا بعملية overload للدالة toString والتى مهمتها إظهار قيمة الإستثناء الذى حدث.
- ثم قمنا بإنشاء الفصيلة UsingExceptionDemo وقمنا بإنشاء دالة compute والتى تلقى باستثناء من النوع الذى أنشأته فى الفصيلة My Exception وذلك كما فى السطر 15.
- ثم حددنا شرط لحدوث الاستثناء وهو إذا كانت القيمة أكبر من 15 وذلك كما فى السطر 18 فيتم إلقاء الاستثناء أما إذا كانت القيمة أقل فلا يتم إلقاء الاستثناء.
- ثم قمنا باستدعاء الدالة compute مرتين كما فى السطرين 25،26 مرة بقيمة أقل من 15 فلا يحدث استثناء ومرة أكبر فيحدث استثناء وهكذا نجد أن النتيجة كالآتى:

```
C:\Java Exception Demo
    Called Compute(1)
    Normal exit (20)
    Caught My Exception (20)
```

وهكذا نكون قد انتهينا من الإستثناءات و طرق معالجتها.

يرجع مرة أخرى للكتاب.

فى Oracle PL/SQL

معنى Exception

الكلمة معناها استثناء وهو جزء من الأوامر يتم توجيه سير البرنامج إليه عند حدوث خطأ ليقوم هو بتناول الخطأ والقيام بعمل اللازم Error Handling بالإضافة إلى إظهار الرسالة وذلك بعد قيام المبرمج بتحديد العمل اللازم عند حدوث الخطأ وكذلك الرسائل المطلوب إظهارها عند حدوث الخطأ.

أنواع Exceptions

توجد ثلاث أنواع للاستثناء Exceptions وهى :

1- استثناء معرف Predefined Exception.

2- استثناء من تعريف المبرمج User defined Exception.

3- استثناء داخلي Internal Exception.

1. الاستثناء المعرف من قبل Predefined Exception

وهو استثناء معرف من قبل Oracle لمجموعة من الحالات والتعامل معها وتسمى Predefined أو built-in ، فمثلاً هناك خطأ معرف يسمى NO DATA عند قيام جملة Select بإعادة بيانات ولا تعيد يظهر No Data أو خطأ أخر وهو قيام جملة select بإعادة اكثر من سجل في حين أنك تستعمل select into حيث تخزن هذه البيانات فى متغير سجل واحد ، وهذه الأخطاء يتم التعامل معها من قبل Oracle ولكن بإظهار رسالة من داخل Oracle ولكن لتغير الرسالة والتعامل مع هذا الخطأ من قبل المبرمج تقوم بكتابة الأوامر فى جزء الـ Exception فى نهاية بلوك الأوامر كما فى شرح بلوك الدالة كما سبق.

وهذا النوع يتميز بأن الـ Oracle يعرف حالة حدوثه لذلك ما عليك سوى كتابة الأوامر التى تنفذ عند حدوث هذا الخطأ.

2. خطأ معرف من قبل المبرمج User_ defined exception

هذا النوع لا يعرفه الـ Oracle ولكن يتم تعريفه من قبل المبرمج لذلك يسمى User defined وله ثلاثة أجزاء هى :

1. جزء التعريف Exception declaration

وهو جزء الإعلان عن اسم هذا النوع من الخطأ لأنه غير معرف لدى Oracle فيتم تعريف متغير جديد من النوع exception

2. اختبار الخطأ Exception teasing

وفيه يتم وضع شرط حدوث هذا الخطأ لأن الـ Oracle لا يعرف معناه ولا يعرف وقت حدوثه ليوجه سير البرنامج إلى قسم الـ Exception.

3. التعامل مع الخطأ Exception handling

وهو قسم الـ Exception التقليدى الذى ينتظر أن يوجه إليه الخطأ بعد تعريف وتحديد حالة حدوثه للقيام باللازم وتنفيذ الأوامر والرسائل التى يكتبها المبرمج.

ولتوضيح هذا الإجراء تابع السطور التالية:

```
EXCEPTION
  WHEN exception1 [OR exception2 . . .] THEN
    statement1;
    statement2;
    . . .
  [WHEN exception3 [OR exception4 . . .] THEN
    statement1;
    statement2;
    . . .]
  [WHEN OTHERS THEN
    statement1;
    statement2;
    . . .]
```

في هذه السطور:

يبدأ التعامل مع الاخطاء بقسم الاخطاء بالكلمة EXCEPTION.

يلي ذلك الامر WHEN الذي يعني عندما ويليه نوع الخطأ المحدد بكلمات معرفة بديل للعبارة exception1 ومن هذه القيم المعرفة ما يلي:

- NO_DATA_FOUND
- TOO_MANY_ROWS
- INVALID_CURSOR
- ZERO_DIVIDE
- DUP_VAL_ON_INDEX

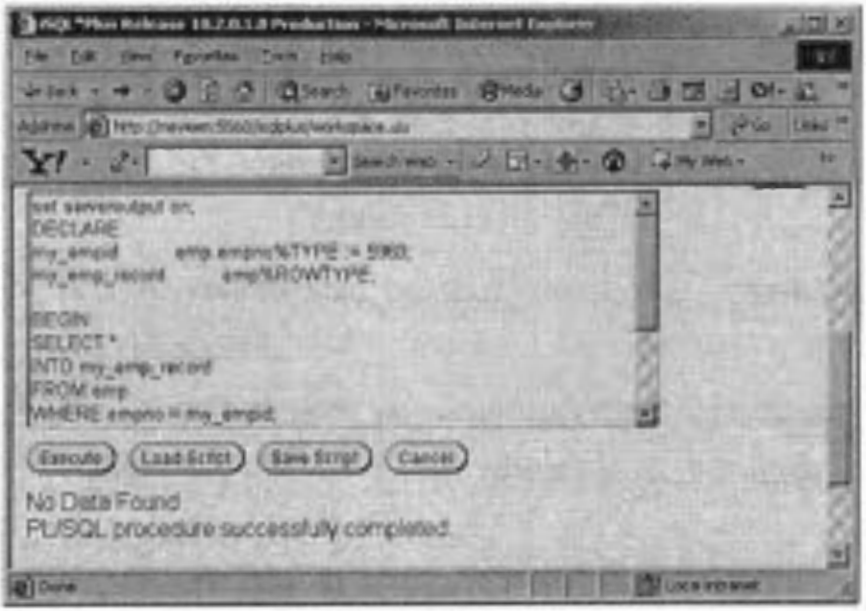
- وفيها العبارة NO_DATA_FOUND تستعمل عند تنفيذ أمر SELECT أو CURSOR ولكن لم تعيد بيانات.
- العبارة TOO_MANY_ROWS وهي عكس العبارة السابقة حيث تستعمل عندما تكون عدد القيم المسترجعة من الامر SELECT كبير بحيث لا يصلح وضعه في متغير واحد.
- المتغير INVALID_CURSOR وتنتج عندما يتم التعامل مع متغير على أنه CURSOR ولكنه ليس كذلك.
- المتغير ZERO_DIVIDE ويعني حدوث عملية قسمة على ZERO.

- المتغير DUP_VAL_ON_INDEX معناها محاولة الإضافة فى جدول مع محاولة تكرار القيمة فى الحقل PRIMARY KEY.

والسطور التالية توضح كيفية التعامل مع الأخطاء المعرفة :

```
DECLARE
my_empid     emp.empno%TYPE := 5969;
my_emp_record     emp%ROWTYPE;
BEGIN
SELECT *
INTO my_emp_record
FROM emp
WHERE empno = my_empid;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No Data Found');
 END;
```

كما فى الشكل (14-10).



الشكل (14-10)

فى هذه السطور:

- بعد الأمر BEGIN الذى يعنى بداية الأوامر يتم استدعاء الأمر SELECT لعرض بيانات الموظف رقم 5969 ولكن فعليا لا يوجد موظف بهذا الرقم.
- لذلك يحدث خطأ يعرفه الـ ORACLE وهو عدم وجود بيانات NO_DATA_FOUND فيؤدى ذلك الى انتقال التنفيذ الى قسم التعامل مع الأخطاء EXECPTION.
- فى القسم الخاص بالأخطاء EXCEPTION تم كتابة الجملة WHEN التى تقول عندما ويليها حالة الخطأ وهى NO_DATA_FOUND أى عندما يحدث الخطأ NO_DATA_FOUND أى لا توجد بيانات وبالتالى يتم طباعة الرسالة المحددة.

ومما سبق نلاحظ أن الصورة العامة للتعامل مع الأخطاء هى:

```
0. DECLARE
1. BEGIN
2. EXCEPTION
3. WHEN Exception1 THEN
4. Sequence of statements
5. WHEN Exception2 THEN
6. Sequence of statements
7. WHEN Exception3 THEN
8. Sequence of statements
9. END;
```

الدوال التى تتعامل مع الأخطاء

يوجد فى ORACLE بعض الدوال للتعامل مع الأخطاء منها الدالة SQLCODE وتعيد رقم يعبر عن رقم الخطأ الناتج والدالة SQLERRM تعيد الرسالة المحفوظة لهذا الخطأ والجدول التالى يوضح أمثلة لبعض القيم المرجعة من SQLCODE.

Example SQLCODE Values

SQLCODE Value	Description
0	No exception encountered
1	User-defined exception
+100	NO_DATA_FOUND exception
negative number	Another Oracle server error number

والسطور التالية توضح استعمال هذه الدوال والمتغيرات

```
DECLARE
  v_error_code      NUMBER;
  v_error_message   VARCHAR2(255);
BEGIN
...
EXCEPTION
...
  WHEN OTHERS THEN
    ROLLBACK;
    v_error_code := SQLCODE;
    v_error_message := SQLERRM;
    INSERT INTO errors
    VALUES(v_error_code, v_error_message);
END;
```

في هذه السطور:

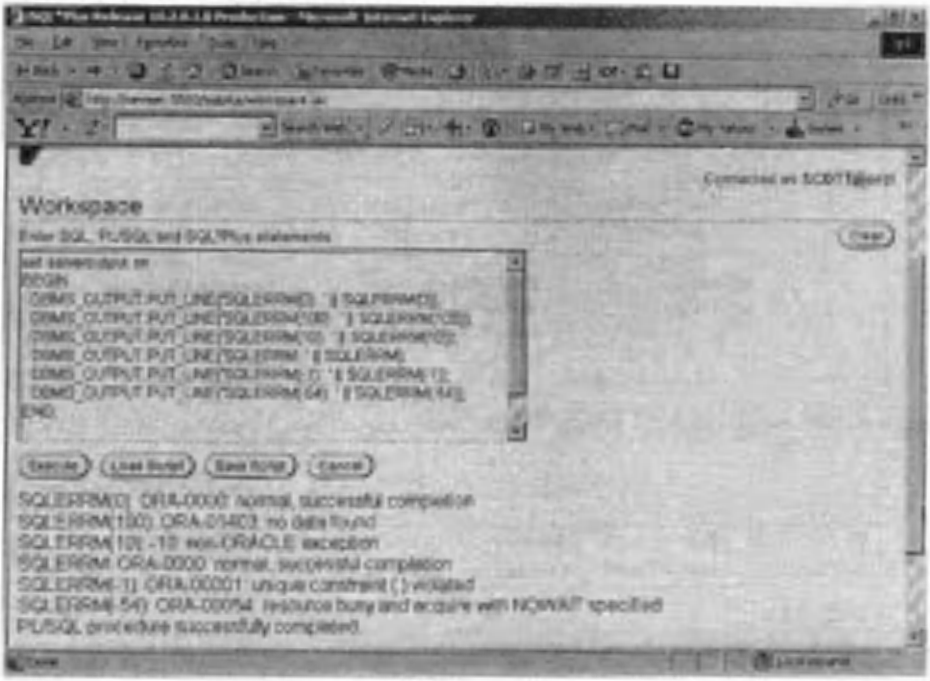
- تم تخزين القيمة الناتجة من SQLCODE في المتغير v_error_code وهي رقم الخطأ error الناتج.
- و تم تخزين القيمة الناتجة من SQLERRM في المتغير v_error_message وهي نص الخطأ error الناتج.

الدالة SQLERRM

الدالة SQLERRM تعرض الرسالة المقابلة للخطأ المحدد رقمه والسطور التالية توضح كيفية استعمالها لطباعة الرسائل المقابلة لبعض الأرقام.

```
/*
 * SQLERRM.sql
 */
set serveroutput on
BEGIN
  DBMS_OUTPUT.PUT_LINE('SQLERRM(0): ' || SQLERRM(0));
  DBMS_OUTPUT.PUT_LINE('SQLERRM(100): ' || SQLERRM(100));
  DBMS_OUTPUT.PUT_LINE('SQLERRM(10): ' || SQLERRM(10));
  DBMS_OUTPUT.PUT_LINE('SQLERRM: ' || SQLERRM);
  DBMS_OUTPUT.PUT_LINE('SQLERRM(-1): ' || SQLERRM(-1));
  DBMS_OUTPUT.PUT_LINE('SQLERRM(-54): ' || SQLERRM(-54));
END;
/
```

وعند تنفيذ هذه السطور نحصل على نتيجة التنفيذ التالية: كما في الشكل(15-10).

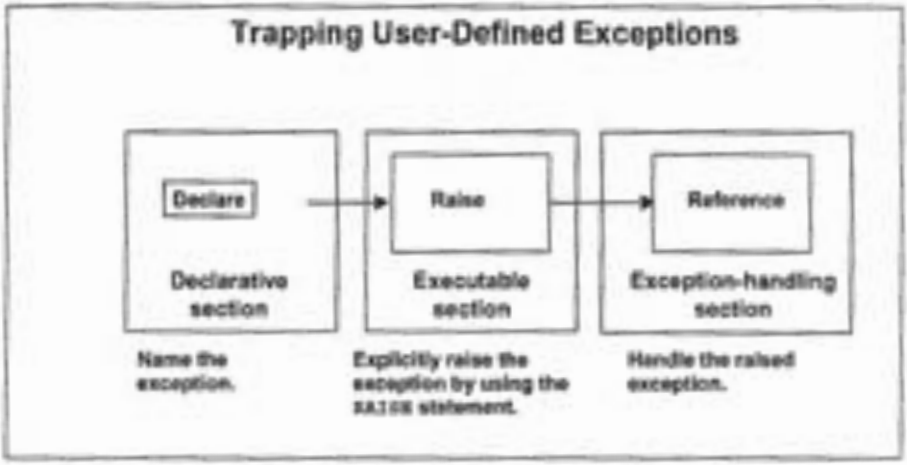


الشكل (15-10)

التعامل مع الاخطاء المعرفة من المبرمج USER DEFINED ERRORS

الأخطاء المعرفة من قبل المبرمج USER DEFINED ERRORS هي أخطاء لا يعرفها الـ ORACLE وربما لا تكون أخطاء مباشرة ولكنها من وجهة منطق البرنامج أخطاء ، فمثلا عندما تزيد نسبة نجاح الطالب عن 100% يعتبر البرنامج ذلك خطأ في حين لا يعرف الـ ORACLE ذلك وبالتالي يقوم المبرمج بتعريف معنى وحالة حدوث هذا الخطأ وكيفية التعامل.

ويأخذ ذلك التركيب التالي:



والمثال التالى يوضح كيفية تعريف خطأ EXCEPTION من قبل المبرمج ثم التعامل معه فى قسم الـ EXCEPTION.

```
DEFINE p_department_desc = 'Information Technology '
DEFINE P_department_number = 300
```

```
DECLARE
   e_invalid_department EXCEPTION;
BEGIN
  UPDATE       departments
  SET          department_name = '&p_department_desc'
  WHERE        department_id = &p_department_number;
  IF SQL%NOTFOUND THEN
    RAISE e_invalid_department;
  END IF;
  COMMIT;
EXCEPTION
  WHEN e_invalid_department  THEN
    DBMS_OUTPUT.PUT_LINE('No such department id.');
END;
```

فى هذه السطور:

- تم تعريف متغيرين الأول أخذ عبارة حرفية والثانى أخذ قيمة رقمية وذلك فى بيئة SQL PLUS.
- تم الإعلان عن متغير بالاسم v_invalid_department من النوع EXCEPTION.
- بعد الأمر BEGIN تم استعمال الأمر UPDATE للتعديل فى بيانات الإدارة رقم 300 حسب المتغير المعرف من قبل.

- ثم تم استعمال الأمر IF لاختبار هل تم التعديل أم لا وبالطبع لا لعدم وجود الموظف فتستدعى الجملة RAISE التى تنقل التنفيذ إلى قسم الأخطاء EXCEPTION لتجد الأمر WHEN الذى يتعامل مع الخطأ.

مثال:

نوضح كيفية تعريف خطأ من قبل المستخدم من النوع USER DEFINED ERROR وكيفية التعامل معه وذلك كما فى السطور التالية:

```
DECLARE
my_empid employee.empid%TYPE := 59694;
my_emp_record employee%ROWTYPE;
my_salary_null EXCEPTION;
BEGIN
SELECT * FROM employee
INTO my_emp_record
WHERE empid = my_empid;

IF my_emp_record.salary IS NULL THEN
RAISE my_salary_null;
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No Data Found');
WHEN my_salary_null THEN
DBMS_OUTPUT.PUT_LINE('Salary column was null for employee');
END;
```

فى هذه السطور:

- فى السطر رقم 4 تم الإعلان عن exception بالاسم salary_ NULL.EXR مع ملاحظة أن نوعه EXCEPTION.
- في السطر رقم 9 تم استعمال جملة IF للاختبار قيمة حقل المرتب SALARY فإذا كانت NULL تم استعمال الأمر RAISE للإرسال التنفيذ إلى القسم EXCEPTION إلى السطور الخاصة بهذا الخطأ .
- في السطر رقم 14 تم استعمال WHENمع اسم الخطأ Salary_ NULL.EXR أي في حالة حدوث هذا الخطأ THEN أي إذن قم بتنفيذ ما يلي.
- وهو طباعة الرسالة.

الفصل الحادى عشر

البـرمجـة المـرئيـة
Visual Programmig

Programming Concepts 11 فى هذا الفصل

فـي هـذا الفصـل نبـدأ فـي اسـتعمال أجـزاء VB.NET فـي كتابـة البـرامج و نبـدأها بصريـع الأدوات Toolboxو ذلـك مـن خـلال النقـاط التالية:

- استخدامات الأدوات الأساسيه .
- كيفيه إضافه أداة أخرى.
- الخصائص المشتركه.
- أداة العنوان LABLE.
- الأداة LinkLabel
- أداة الكتابه TEXT BOX.
- أداة الصورة PICTURE.
- أداة الإطار FRAME.
- أدائي الاختيار OPTION , CHECK.
- الاداة :WebBrowser
- الاداة RichTextBox
- استعمال كتب المساعدة HELP.

معنى Visual Programmig

بعد تقديم مفهوم البرمجة بالأهداف oop تم تقديم مفهوم آخر يعتبر امتداد له وهو البرمجة المرئية وهى فكره جيدة توفر عليك كتابة أوامر لاستعمال فصيلة معينة ، فمثلا بدلا من كتابة أوامر لاستعمال الفصيلة button وتعريف button لاستعمالها يمكنك سحب الأدكة button وتوقيعه على نموذج البرنامج فيتم التعريف تلقائيا ولذلك سمى برمجة مرئية حيث يمكنك تكوين شكل البرنامج بشكل مرئى وليس عليك كتابة الأوامر لإنشاء قائمة اختيارات menu بل تعدها بشكل مرئى (visual) ، وقامت على هذه الفكرة معظم اللغات الحديثة مثل اللغات التالية:

Delphi, Borland C, Oracle, Visual Basic, Visual Basic.Net , Visual C++,Visual C#,

ويعتبر هذا هو المفهوم السائد حاليا ومعظم اللغات تطبقه حيث تتميز اللغات الحالية بأنها :

وذلك يسهل عملية إعداد البرامج ويزيد من سرعة عملية التطوير والإعداد فليس عليك كتابة أوامر تعريف الفصائل التى تستعملها فقط تستعملها بشكل مرئى ،حيث تقوم بسحب الأدكة من شريط الأدوات Tool Box وتوقعه على نموذج خلفية التطبيق Form.

وتقوم البرمجة فى لغات Visual programming language على ثلاث خطوات

أي برنامج نوافذي يمر بثلاثة خطوات هي :

1- تصميم شكل البرنامج CREATEING USER INTERFACE ويتم باستعمال مربع الأدوات.
2- تغيير خصائص الأدوات CHANGING PROPRITIES بما يناسب البرنامج المطلوب من مربع الخصائص.
3- كتابة الأوامر التي تحقق أغراض البرنامج WRITING CODE ويتم باستعمال مجموع أوامر اللغة.

وتتفق معظم اللغات فى الخطوة الأولى والثانية وتختلف فى الخطوة الثالثة وهى كتابة الأكواد حيث أن لكل لغة طريقة كتابة الأوامر المحددة بها .

غير أن معظم اللغات الحديثة تقاربت كثيرا في قواعد اللغات كما لاحظت في الفصول السابقة.

في لغة VB.NET وفي لغة C#

تجد نفس شاشات التصميم ونفس الأدوات ونفس الخصائص والفارق هو طريقة كتابة الأوامر التي شرحناها في الفصول السابقة.

في لغة VB.NET يوجد مربع الأدوات ToolBox واستخدامات الأدوات الأساسية :

في هذه الفقرة نقوم بعرض الأدوات الأساسية في مربع الأدوات ToolBox و عرض وظيفة كل أداة علي أن نتناولها بالتفصيل فيما بعد والشكل (1-11) يعرض مربع الأدوات و علي كل أداة رقم يشير إليها.

شكل (1-11)

حاول توقيع هذه الأدوات علي خلفية البرنامج و مشاهدة شكلها بعد التوقيع .

الرقم	الاسم الأجنبي	اسم العربي	الاستخدام
1	LABLE	العنوان	هذه الأداة لكتابة عنوان ثابت لا نريد تغييره و هي تستخدم في كتابة العناوين.
2	TextBox	أداة الكتابة	هي أكثر الأدوات استعمالا و هي تستعمل لاستقبال قيم من المستخدم كما تستخدم لأخراج نتائج العمليات.
3	PICTURE BOX	أداة عرض الصورة	أداة تستخدم هذه الأداة لعرض صورة
4	COMMAND	زر الأوامر	تستخدم لتنفيذ عملية معينة حيث يضغطه المستخدم فينفذ العملية المقلوبة.

وتنقسم خصائص الأدوات إلى نوعين هما:

الخصائص المشتركة: Common Properties

وهى الخصائص التى تشترك فيها معظم أو كل الأدوات تقريباً مثل خصائص تغيير الخط واللون وغيره، ويمكنك معرفة الخصائص المشتركة بتوقيع أكثر من أداة من أنواع مختلفة ثم اختيارها جميعا ثم عرض مربع الخصائص تلاحظ عرض الخصائص المشتركة فقط

الخصائص الخاصة:

وهى الخصائص التى تنفرد لها كل أداة وتحقق أعمالها.

و قبل أن نعرض الخصائص المشتركة نقوم باستعمال أداة العنوان LABLE و تطبيق هذه الخصائص عليها حتي تكون واضحة ثم نجمعها في جدول .

يتم عرض مربع خصائص أي أداة بضغط الأداة مرة ثم الضغط على مفتاح F4 . أو ضغط الأداة بالطرف الايمن للماوس واختيار Properites من القائمة الفرعية التى تظهر

أداة العنوان . LABLE

تستخدم هذه الأداة لكتابة عنوان ثابت ولتوضيح ذلك قم بتوقيع أداة عنوان LABLE على خلفية البرنامج FORM ثم اضغط الأداة مرة ثم اضغط F4 يظهر مربع خصائص الأداة غير خاصية TEXT للأداة من LABLE1 إلى WELCOME WITH VB.Net كما بالشكل (11-2)

شكل(11-2)

قم بتجربة الخصائص على أداة العنوان كما يلي :

خاصية BACK COLOR

تؤدي هذه الخاصية إلى تغيير لون خلفية الأداة (الأداة هي زر أمر أو text أو form أو..)، اضغط الأداة ثم اضغط F4 إذا كان مربع الخصائص غير ظاهر ثم اضغط زر خاصية BACKCOLOR ثم اضغط الزر المجاور للخاصية تلاحظ ظهور مربع الألوان اختر لونا تلاحظ تغيير لون الخلفية إلى هذا اللون كما بالشكل (11-3).

شكل (11-3)

خاصية FORE COLOR

تؤدي هذه الخاصية إلى تغيير لون الكتابة ForeColor، جرب ذلك مع أداة العنوان كما سبق باختيار الأداة ثم F4 ثم اختيار الخاصية FORE COLOR لتحصل على مربع اختيار الألوان اختر لونا تلاحظ تغيير لون كتابة الأداة كما في الشكل (4-11).

شكل (11-4)

خاصية FONT NAME

تؤدي هذه الخاصية إلى تغيير نوع و حجم خط الكتابة نأجرب ذلك باختيار خاصية FONT NAME من مربع خصائص أداة العنوان يظهر مربع الفونتات اختر خط و غير الحجم و لاحظ التغيير كما بالشكل (5-11) .

شكل (5-11)

مجموعات الخصائص فى هذه الفقرة

نحاول تقسيم الخصائص إلى مجموعات.

خصائص الأبعاد و الموضع :

تحتوي كل أداة على خصائص تحدد موقعها من الخلفية FORM و خصائص تحدد حجمها طولا Height و عرضا Width و هي :

الخاصية LEFT:

و تحدد بعد الأداة عن الجانب الأيسر لها عن الجانب الأيسر لخلفية البرنامج FORM.

الخاصية TOP :

تحدد بعد الأداة من أعلى عن خلفية البرنامج FORM .

وتوجد الخاصية Top,Left داخل المجموعة Location تحت المسمى X و Y.

الخاصية WIDTH :

تحدد عرض الأداة أي البعد الأفقي للأداة .

الخاصية HEIGHT:

تحدد ارتفاع الأداة أي البعد الرأسي للأداة .

وتوجد الخاصية Width,height داخل المجموعة Size.

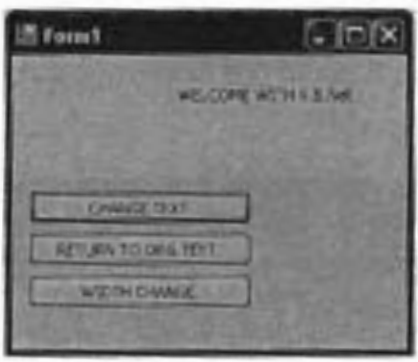
باقي الخصائص

بعض هذه الخصائص يمكن تغييره في حالة التصميم فقط و بعضها يمكن تغييرة في حالة التنفيذ (RUN TIME) فقط و بعضها ينفذ في الحالتين .

و يمكنك معرفة ذلك بضغط الأداة ثم نقر الخاصية ثم ضغط F1 لحصل على شرح للخاصية و مثال عن كيفية استعمالها.

مثال

في هذا المثال نحاول توضيح معظم الخصائص التي تناولناها في الفقرة السابقة وعند تنفيذه لحصل على نتيجة التنفيذ كما بالشكل (6-11) .



شكل (6-11)

في هذا الشكل تلاحظ وجود أداة عنوان Label في أعلى البرنامج مع وجود مجموعة أزرار Button تعبر عن تغيير الخصائص ، جرب استعمال البرنامج كما يلي:

← اضغط زر CHANG TEXT تلاحظ تغيير عنوان الأداة .

← اضغط زر RETURN TO ORG TEXT تلاحظ عودة العنوان الأصلي للأداة .

← اضغط زر CHANG WIDTH تلاحظ تغيير عرض Width الأداة وهكذا.

بعد تجربة استعمال البرنامج تابع معنا خطوات تصميم البرنامج.

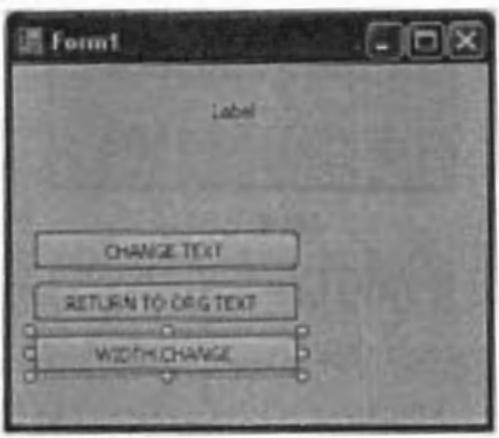
خطوات تصميم البرنامج:

ابدأ برنامج جديد بالخطوات المعتادة و هي FILE ثم NEW ثم Project ثم Visual Basic Projects ثم Windows Application

1. وقع أداة عنوان LABLE وأزرار أوامر Button و رتبها كما بالشكل (7-11).

شكل (7-11)

2- غير عناوين الأدوات بتغيير خاصية Text لتصبح كما بالشكل (8-11) (شكل البرنامج).



شكل (8-11)

3- اكتب أوامر أزرار الأوامر Button و ذلك بالضغط مرتين على الزر و كتابة سطوره كما يلي :

الزر CHANGE TEXT

```
Lable1.Text = "WELCOME WITH V.B.Net"
```

الزر RETURN TO ORG TEXT

```
Lable1.Text = "Label1"
```

الزر CHANGE WIDTH

```
Lable1.Width = Label1.Width + 50
```

بعد كتابة الأوامر نفذ البرنامج بالضغط على F5 تحصل على نتيجة التنفيذ كما سبق.

جرب استعمال البرنامج.

في هذه السطور تم التعامل مع الخصائص باستعمال الأوامر فمثلاً السطر التالي:

```
Lable1.Text = "Welcome With VB"
```

معناه = اجعل خاصية العنوان Text للأداة Label1 تساوى العبارة Welcome وهذا ما يحدث بالمثل باقى السطور الخاصة بباقي الأزرار .

2ـ الأداة LinkLabel

تستخدم هذه الأداة لكى تصل المستخدم إلى مكان آخر.

3ـ أداة الكتابة TEXT BOX

تستخدم هذه الأداة في استقبال و آخراج القيم من و إلى المستخدم و هي من أكثر الأدوات المستخدمة حيث لا يخلو برنامج منها .

و تحتوي هذه الأداة على مجموعة من الخصائص التي تعدد استخداماتها و نعرضها فيما يلي :

الخاصية TEXT

هي الخاصية الأساسية للأداة و هي تعبر عن محتوي الأداة كما في المثال الأول في الفصل الأول فمثلا العبارة TEXTBox1.TEXT تعبر عن محتوي الأداة بالخاصية TEXT.

الخاصية PASSWORD CHAR

تفيد هذه الخاصية عند استخدام أداة الكتابة في استقبال كلمة سر حيث تأخذ الخاصية أحد الحروف و تكتبه بدلا من أي حرف يكتبه المستخدم و بالتالي لا تظهر كلمة السر بينما يظهر الحرف و أشهر الحروف المستخدمة هو العلامة *.

الخاصية MULTI LINE

تأخذ هذه الخاصية القيمة TRUE أو FALSE فإذا كانت TRUE تسمح الأداة للمستخدم بكتابة أكثر من سطر فيها أي متعدد السطور و العكس صحيح .

الخاصية SCROLL BAR

تحدد هذه الخاصية قضيب الانزلاق هل يكون أفقيا أم رأسيا أم الاثنين معا و ذلك إذا كانت خاصية MULTILINE تأخذ القيمة TRUE.

الخاصية BORDORSTYLE

- وتفيد في تحديد شكل إطار الأداة وتأخذ القيم
- None وتعني عدم إضافة إطار Border
- Fixed Single ويعني إضافة إطار فردي ثابت
- Fixed3D ويعني إضافة إطار ثابت مجسم

بالإضافة لهذه الخصائص تحتوي الأداة TEXT على الخصائص المشتركة مثل لون الخلفية BACK COLOR و لون الكتابة FORECOLOR و نوع و حجم الخط FONT و هكذا.

أداة الصورة PICTURE BOX

تستخدم هذه الأداة في عرض صورة في البرنامج كما يمكن أن تأخذ هذه الصورة من ملف قواعد بيانات أو لتوضيح ذلك اتبع ما يلي :

قم بتوقيع أداة الصورة على خلفية برنامج جديد و غير حجمها لتصبح كما سبق

- اضغط أداة الصورة ثم اضغط مفتاح F4 و اختر خاصية Image .
- اضغط الزر المجاور لها تلاحظ ظهور مربع حوار فتح ملف.

- ابحث عن ملف صورة على الجهاز و اختاره تلاحظ عرض هذه الصورة في مربع الصورة كما بالشكل (10-11) .

شكل (10-11)

و بالتالي تعتبر أهم خاصية في أداة الصورة هي الخاصية Image و لكن توجد خصائص أخري مهمة منها.

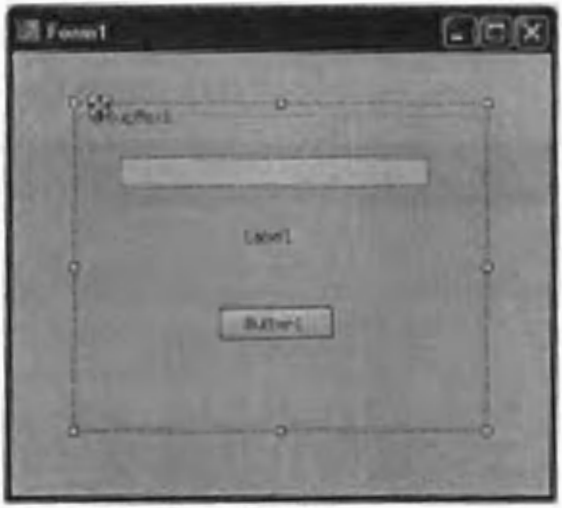
الخاصية BORDER STYLE

وتحدد شكل إطار للصورة ويأخذ القيم ذلك بالإضافة للخصائص المشتركة.

أداة الإطار GroupBox

تستعمل هذه الأداة لوضع مجموعة أدوات داخلها و معاملة هذه الأدوات كأداة واحدة ولتوضيح ذلك اتبع ما يلي:

- قم بتوقيع الأداة GroupBox على FORM1 و ارسم مستطيل ثم قم بتوقيع أداة كتابة TextBox ، أداة عنوان Lable ، زر أوامر Button مباشرة داخل الـ GroupBox الذي انشأته لتحصل على الشكل (11-11).



شكل (11-11)

- بهذا تعتبر جميع هذه الأدوات داخل GroupBox1 كأداة واحدة و للتأكد من ذلك اضغط طرف الإطار بالفأرة ثم اضغط مفتاح DEL تلاحظ مسح أداة الإطار و ما بداخلها، اضغط المفتاحين Ctrl+2 يعود كل شيء وسوف نصمم برنامج يوضح ذلك.
- كما تتصف أداة الإطار بالخصائص العامة مثل الألوان و الفونتات و العنوان.

شكل (11-12)

الأداة MaskedTextBox:

تستخدم هذه الأداة TextBox لاستقبال مدخلات المستخدم ولكن تتميز عنها بامكانية التحكم فى شكل ومواصفات مدخلات المستخدم بسهولة ، حيث يمكن إجبار المستخدم على إدخال أرقام فقط أو تواريخ فقط وهكذا.

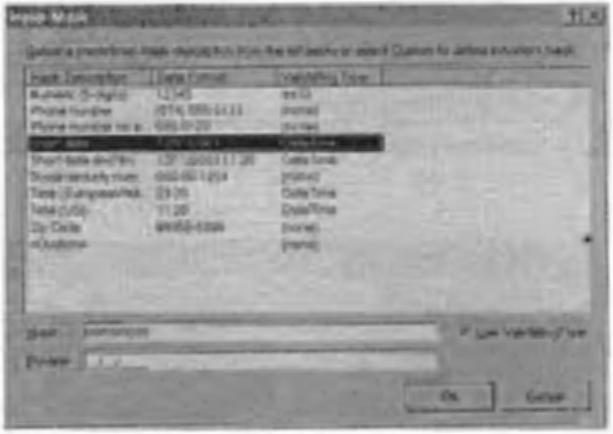
شكل (11-13)

ولتوضيح ذلك تابع الخطوات التالية:

1- قم باعداد تطبيق جديد كم النوع Windows Application.

2- قم بتوقيع أداة MaskedTextBox وأداة Button وأداة TextBox.

3- قم بعرض خصائص الأداة MaskedTextBox واختر الخاصية Mask تلاحظ ظهور مجموعة من الاشكال القياسية التى يمكن اختيارها للتحكم فى مدخلات المستخدم كما فى الشكل (11-14).



الشكل (11-14)

4- اختار Short date وهذا يعني انك ترغب في استعمال مربع النص MaskedTextBox في استقبال تاريخ.

5- قم بكتابة السطر التالي في الحدث Click لزر الأمر Button1.

```
TextBox1.Text = MaskedTextBox1.Text
```

في هذا السطر يتم كتابة ما يكتبه المستخدم في MaskedTextBox1 في مربع النص TextBox1

6- نفذ البرنامج وحاول الكتابة في مربع MaskedTextBox1 تلاحظ عدم امكانية كتابة أي شيء غير التاريخ كما حددت في الخصائص.

اكتب تاريخ ثم اضغط زر الأمر Click تلاحظ نسخ التاريخ الى مربع النص TextBox1 كما في الشكل (15-11).

الشكل (15-11)

الأداة NumericUpDown1

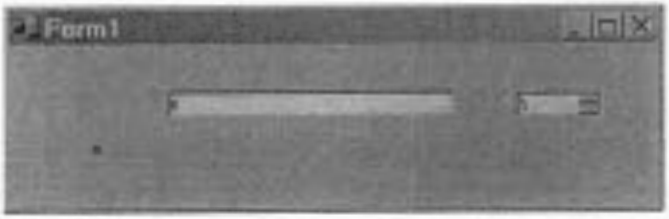
توفر هذه الأداة عمل عداد للأرقام ولتوضيح ذلك تابع ما يلي

1- قم بتوقيع الأداة NumericUpDown ومربع نص على نموزج Form التطبيق

2- قم بكتابة السطر التالي في الحدث NumericUpDown1_ValueChanged

```
TextBox1.Text = NumericUpDown1.Value
```

3- نفذ البرنامج وجرب تغيير قيم الأداة NumericUpDown تلاحظ تغيرها في مربع النص كما في الشكل (16-11).



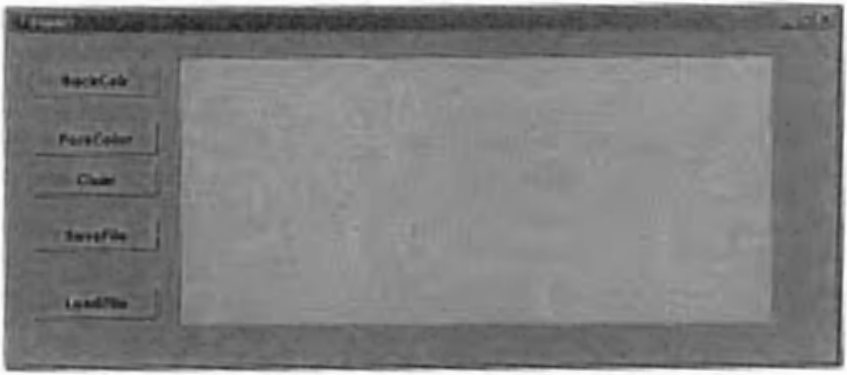
الشكل (16-11)

الأداة RichTextBox

توفر هذه الأداة معظم عمليات الكتابة أكثر بكثير من مربع النص TextBox حيث يمكنك تحقيق معظم عمليات برنامج الكتابة تماما ولتوضيح ذلك تابع الخطوات التالية :

1- قم باعداد تطبيق جديد من النوع Windows Application.

2- قم بتوقيع الأداة RichTextBox وأدوات Button حتى تحصل على الشكل (17-11).



الشكل (17-11)

3-قم بكتابة سطور البرنامج كما يلى:

1. زر الأمر BackColor

```
RichTextBox1.SelectionBackColor = Color.Coral
```

يقوم هذا السطر بتغيير لون خلفية النص الذى يختاره المستخدم.

2. زر الأمر ForeColor

```
RichTextBox1.SelectionColor = Color.Red
```

يقوم هذا السطر بتغيير لون كتابة النص الذى يختاره المستخدم.

3. زر الأمر Clear

```
RichTextBox1.Clear()
```

يقوم هذا السطر بمسح محتويات أداة الكتابة كلها.

4. زر الأمر Save File

```
RichTextBox1.SaveFile("c:\Rtf1.doc")
```

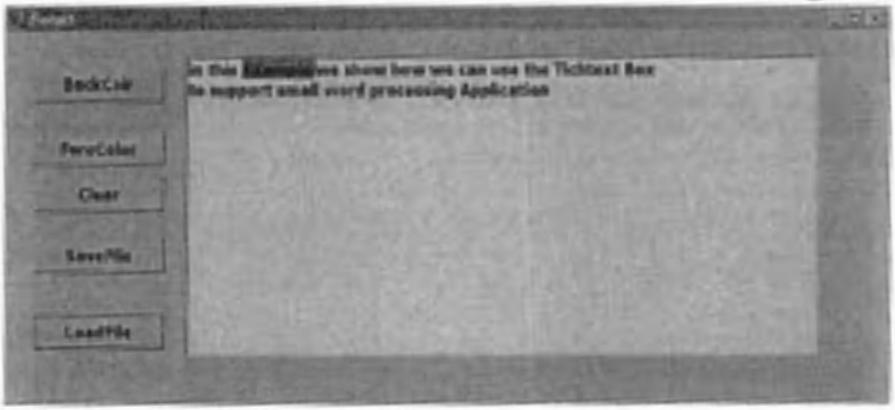
يقوم هذا السطر بحفظ محتويات أداة الكتابة كلها فى الملف "c:\Rtf1.doc" .

5. زر الامر LoadFile

```
RichTextBox1.LoadFile("c:\Rtf1.doc")
```

يقوم هذا السطر باسترجاع محتويات الملف "c:\Rtf1.doc" وعرضها في أداة الكتابة.

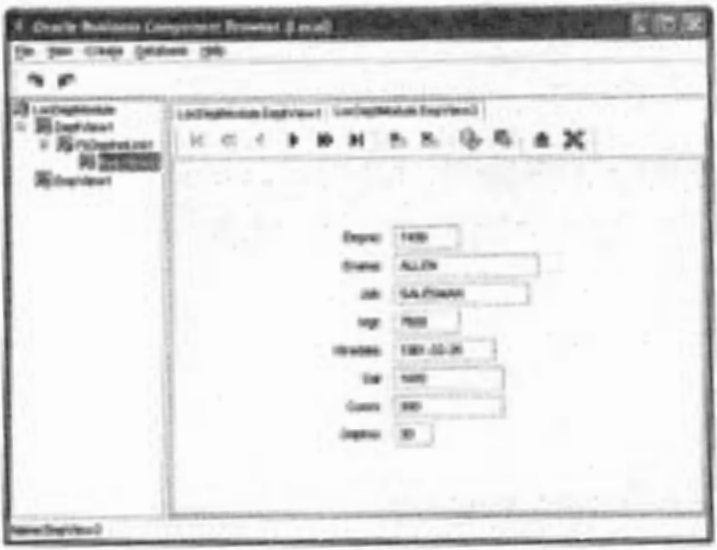
نفذ البرنامج وجرب استعماله تلاحظ انه يحقق هذه الوظائف كما في الشكل (18-11).



الشكل (18-11)

في لغة Java

يعتمد ذلك على بيئة التطوير التي تستعملها فاذا كنت تستعمل مثلا Oracle Jdeveloper تلاحظ ظهور الأدوات كما في الشكل (19-11).



الشكل (19-11)

حيث يوفر مربع أدوات كامل مثل المتوفر فى Visual Stidio.net مع توفر نفس الطريقة فى البرمجة من سحب وتوقيع الادوات ثم تغيير الخصائص ثم كتابة الأوامر.

ويعتبر البرنامج Jdeveloper اداة البرمجة واعداد تطبيقات باستعمال لغة Java بجميع اشكالها مثل كود Java أو jsp أو Java beans أو تطبيقات قواعد بيانات بين Java وقاعدة بيانات oracle أو غيرها وهذه الأداة ولغة Java تعتبر المستقبل الذى ترمى اليه شركة Oracle لذلك يفضل الاهتمام بها من الآن .

والشكل التالى يعرض الشاشة الرئيسية للبرنامج وسوف تجد مقدمة عن استعمال البرنامج وكذلك لغة الـ Java فى أخر هذا الكتاب.

كما فى الشكل(20-11)

الشكل (20-11)

كما توفر لغة الـ Java مجموعتين من فصائل اعداد واجهة التطبيق الأولى هى AWT والثانية والحديثة Swing والتى توفر معظم الأدوات التى يمكن أن تحتاج إليها فى التصميم وتظهر هذه الأدوات بشكل مرئى Visual فى بيئات تطوير تطبيقات لغة Java مثل Jdeveloper كما عرضنا وكذلك أداة بيئة التطوير Net Beans من إنتاج شركة Sun.

فى Oracle Forms

أدوات مطور النماذج FORM BUILDER TOOLS

أثناء عملية التصميم وتطوير النماذج Forms توجد مجموعة من الأدوات التي تحتاجها في عملية التصميم وبناء النماذج والتى يوفرها مطور النماذج Form Builder منها ما يلى :

1. مصمم شكل النماذج LAYOUT EDITOR

وهو ما نستعمله كثير عند التصميم كما بالشكل(21-11)

الشكل (21-11)

ويعرض شكل النموذج ويمكنك فيه ضبط محتويات النموذج ونقلها من مكان إلى أخر وتغيير عناوينها وغيره.

ويتم الانتقال إليه إما بالضغط علي المفتاح F2 أو أو اختيار TOOLS ثم LAYOUT EDITOR

2. نافذة عناصر النموذج OBJECT NAVIGATOR

وهي نافذة يعرض فيها مكونات التطبيق من نماذج FORMS وقوائم MENUS ومكتبات LIB وغيره كما بالشكل (22-11).

الشكل (22-11)

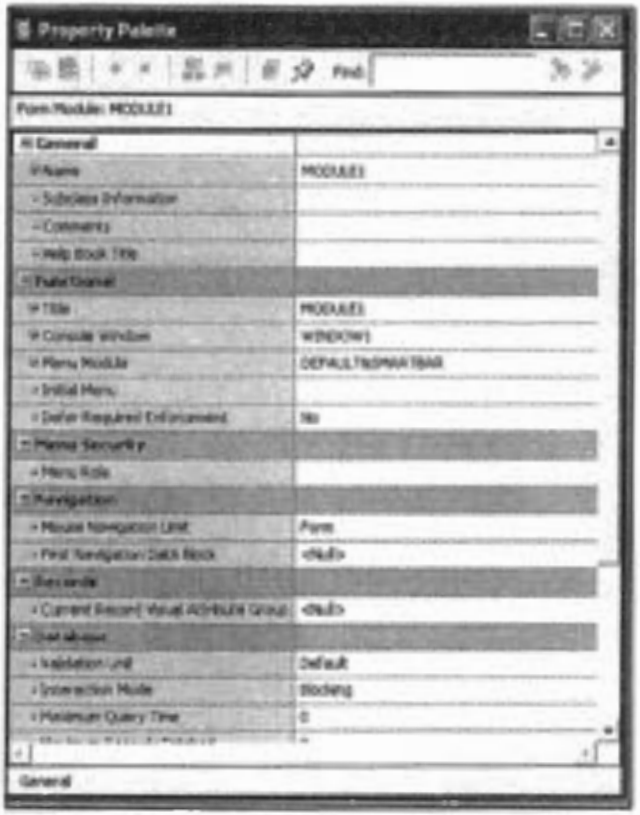
كما تلاحظ في الشكل يتم عرض المكونات علي شكل شجرة.

ويمكن الوصول لهذه النافذة بالضغط علي F3 أو اختيار TOOLS من القائمة الرئيسية ثم اختيار OBJECT NAVIGATOR وسوف نتناول فائدة كل عنصر من عناصر هذه النافذة.

3. نافذة الخصائص PROPERTY PALLET

وهي نافذة مهمة جدا أثناء التصميم فهي تعرض خصائص للأداة التي تحتاجها وعن

طريق هذه النافذة تستطيع إعطاء قيم لهذه الخصائص ، وبالتالي التغيير في العناصر ويمكن الوصول إليها بالضغط على زر F4 وفي الإصدار G 9i,1 تأخذ هذه النافذة الشكل (23-11).



الشكل (23-11)

جرب التنقل بين هذه النوافذ و التعامل معها حتى تعتاد عليها.

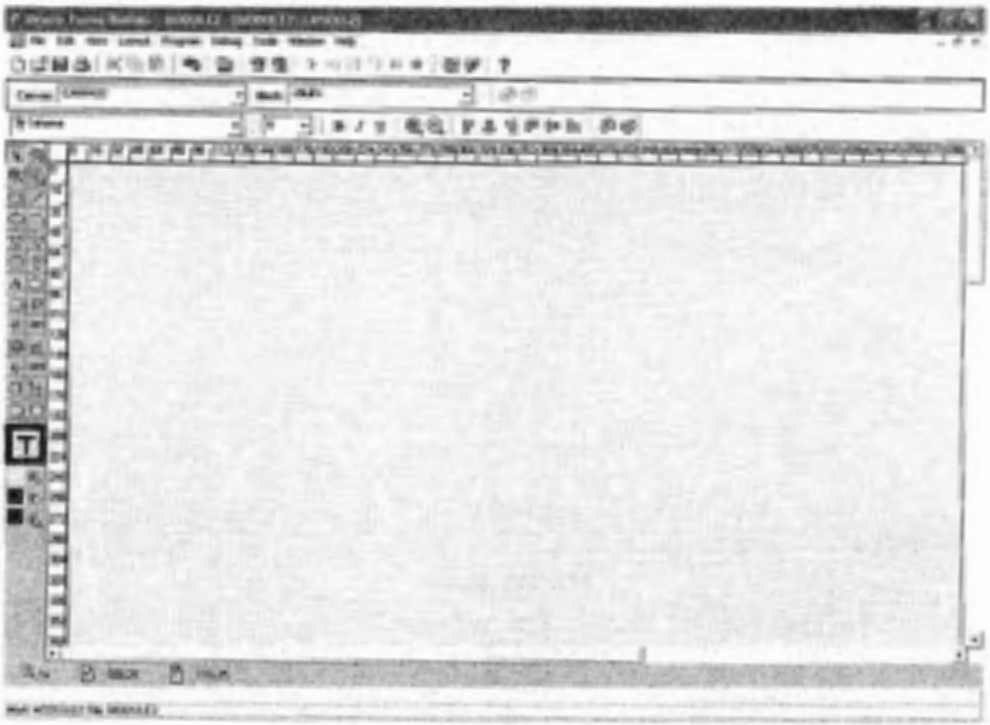
العناصر التي يمكن أن يتعامل معها مطور النماذج

إذا تفحصت نافذة العناصر OBJECT NAVIGATOR تجد أنه هناك مجموعة من العناصر التي يمكن أن تتعامل معها منها.

1. النموذج FORM MODULE

وهو العنصر الأساسي الذي يمثل البيانات للمستخدم ويستقبلها منه ويخزنها في قاعدة البيانات.

ويسترجعها له عند الحاجة وفيها يتم تحديد قيود الإدخال وغيره وفى الإصدار G 9I,10 تأخذ هذه النافذة الشكل (24-11).



الشكل (24-11)

2. قوائم الاختيارات MENUS

وهي الشكل المعتاد لقوائم الاختيارات والتي تسمي PULL DOWN MENUS أو الشكل الثاني POP UP MENU لعرض الاختيارات الكثيرة للمستخدم بشكل منظم ويأخذ مصمم القوائم MENU BUILDER فى الشكل (25-11).

الشكل (25-11)

1- مكتبة الدوال والبرامج الفرعية PL/SQL LIB.

وهى مكتبة من الدوال FUNCTIONS والبرامج الفرعية PROCEDURES التى يتم إنشا تها وحفظها فى مكتبة بحيث يمكن استعمالها من داخل أكثر من نماذج FORM.

2- مكتبة العناصر OBJECT LIB.

هى مكتبة من العناصر ITEMS مثل الأزرار BUTTONS ومربعات النصوص TEXT_ITEMS.

3- الخصائص المرئية المشتركة VISUAL ATTRIBUTES.

طريقة لتجميع مجموعة من المرئية مثل الألوان والفونتات تحت اسم لاستعمالها أكثر من مرة.

بالرغم أن المستخدم يري نموذج تسجيل البيانات FORM كوحدة واحدة آلا أنه من وجهة نظر المطور واقعيا يتكون من أكثر من وحدة ، فهو عبارة عن وحدات متداخلة ويظهر ذلك من الشكل (26-11).

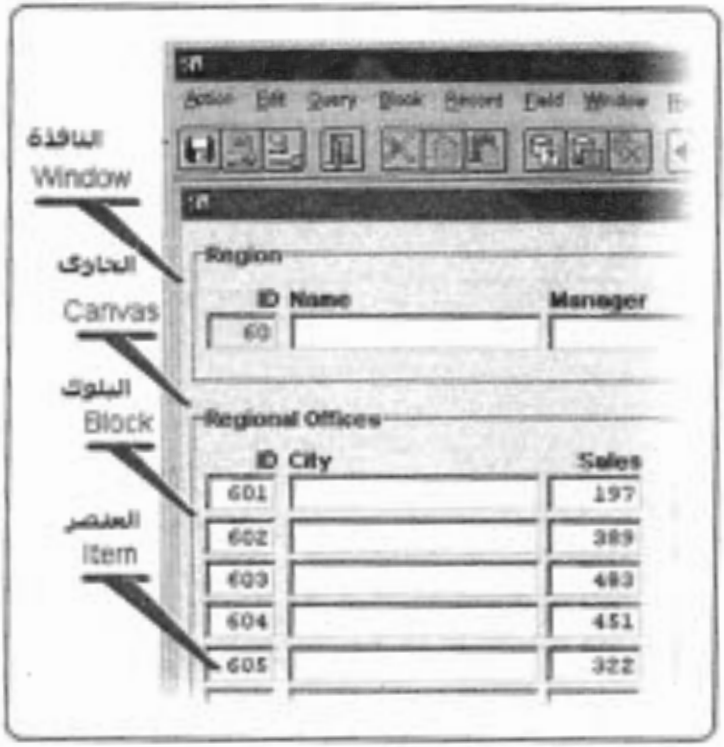
الشكل (26-11)

من الشكل 57 تلاحظ أن النموذج FORM يتكون من أربعة عناصر أساسية وها أنواع مختلفة هذه العناصر هي :-

1. عنصر النافذة WINDOW OBJECT.
2. عناصر الحاوي (الكانفس) CANVAS OBJECT.
3. عنصر البلوك BLOCK OBJECT.
4. عنصر الأدوات التي توضع علي النموذج ITEM OBJECT.

هى مكونات النماذج التى تمثل البيانات المطلوب التعامل معها سواء بالإدخال أو الاستعلام.

ويظهر ذلك كما فى الشكل (27-11).


النافذة
Window
الحاوي
Canvas
البلوك
Block
العنصر
Item
Action Edit Query Block Record Field Window
Region
ID Name Manager
60
Regional Offices
ID City Sales
601 197
602 389
603 483
604 451
605 322

القسم الخامس

الفصل الثانى عشر

مقدمة عن قواعد البيانات وأوامر لغة الاستعلامات SQL

فى هذا الفصل Programming Concepts 12

فى هذا الدرس نشرح استخدامات لغة الاستعلامات SQL وذلك من **خلال النقاط التالية :**

- ما المقصود بقواعد البيانات
- مزايا استعمال الكمبيوتر فى إنشاء قاعدة بيانات
- تركيب قاعدة البيانات
- كيفية استعمال MS ACCESS كمثال بسيط
- عناصر قاعدة البيانات
- التكامل فى قاعدة البيانات
- تصحيح تصميم الجداول NORMALIZATION
- ما هى لغة الاستعلامات SQL
- أجزاء لغة SQL
  - لغة التعريفات DDL
  - لغة التحكم DCL
  - لغة الوصول للبيانات DML

قواعد البيانات database

فى الفترات القديمة والأولى لتطوير الكمبيوتر لم يكن لقواعد البيانات مكان بل كانت البيانات تحفظ فى ملفات تسمى ملفات مستوية Flat Files ، ولكن مع التطور اصبح مفهوم قواعد البيانات ، من أهم المفاهيم ، وهو تسجيل البيانات فى شكل منظم عبارة عن حقول وسجلات فى جداول ، ومجموعة الجداول تشكل قاعدة بيانات وتوفر عمليات متعددة مثل البحث ، والترتيب ، والعمليات الحسابية على الحقول وغيرها .

وبدأت قواعد البيانات قديما برامج بسيطة مثل dbase III+ ، وطورت بعد ذلك لتشتمل الآن على الكثير من البرامج منها :

1- برنامج MS Access الذى يأتى ضمن مجموعة MS Office

2-قاعدة بيانات Oracle والتى تمثل أقوى قاعدة بيانات حتى الآن

3- MS SQL server والذى أصدرته مايكروسوفت ليضاهى قاعدة بيانات Oracle

4- قاعدة بيانات DB2 من شركة IBM ، والكثير من لغات قواعد البيانات

ما المقصود بقواعد البيانات WHAT IS DATABASE

بصرف النظر عن استعمال الكمبيوتر لو نظرت ورقيا تجد أن أى مؤسسة تحتفظ بالبيانات فى شكل قاعدة بيانات فمثلاً.

بيانات العاملين

تجد إدارة شئون العاملين تحتفظ بملف لكل موظف يحتوى هذا الملف على بياناته الشخصية ومؤهلاته والشهادات الحاصل عليها حتى الآن. وتواريخ الترقي وكذلك العلاوات والجزاءات والدورات الحاصل عليها وهذه تسمى قاعدة بيانات فى أبسط صورة.

صديقى

اذا كنت على دراية بأساسيات قواعد البيانات واستخدام قواعد البيانات MS ACCESS أو SQL SERVER أو ORACLE (كما هو متوقع) يمكنك الانتقال إلى الفقرة المناسبة لمستواك حتى لا تشعر بالملل حيث أننا فى هذه الفقرة نراجع أساسيات فى قواعد البيانات يجب الاتفاق عليها

بيانات العملاء

فى إدارة المبيعات تلاحظ الإدارة تحتفظ بيانات العملاء الشخصية وحركات البيع وحجم المبيعات والمواقف المالية وغيرها ، وهذه تسمى قاعدة بيانات.

وهناك الكثير من الأمثلة الأخرى ، ولذلك تلاحظ أنه لا غنى عن وجود قواعد بيانات ولو ورقية ، ولكن السؤال لماذا نحول قاعدة البيانات الورقية إلى كمبيوتر؟

الإجابة:

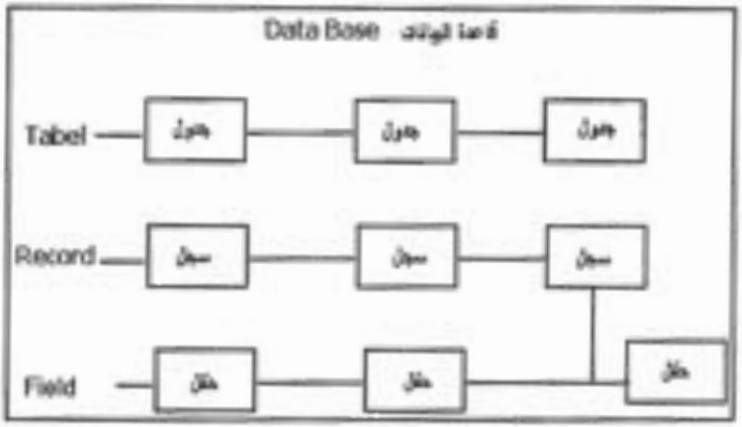
- صعوبة البحث فى قاعدة البيانات الورقية والوقت المستغرق فى حين يتوفر ذلك فى قاعدة البيانات الكمبيوتر بسهولة.
- صعوبة الحصول على معلومات تحليلية من البيانات فى حين يتوفر ذلك فى قاعدة بيانات الكمبيوتر.
- صعوبة التأكد من دقة المعلومات لاحتمال الخطأ فى حين توفر الدقة فى معلومات قاعدة بيانات الكمبيوتر وكثير من المزايا التى سوف نشرحها.

مزايا استعمال الكمبيوتر فى إنشاء قاعدة بيانات

- الاحتفاظ بكم هائل من البيانات بعيداً عن الأوراق وسهولة عمل نسخ كثيره من هذه البيانات.
- سهولة ربط البيانات ببعضها مثل بيانات العاملين وبيانات أصناف المخازن.
- سرعة البحث الهائلة عن البيانات التي لا تقارن مع قاعدة البيانات الورقية.
- إمكانية استخلاص البيانات (المعلومات) التحليلية من كثير من البيانات مثل بيانات موظفين على درجة معينة أو موظفين حاصلين على دورة معينه ..إلخ.
- دقة المعلومات المستخلصة حيث يخرج الكمبيوتر ما تم تسجيله.
- إمكانية إجراء العمليات المختلفة على جدول أو أكثر من البيانات فى وقت قصير مثل زيادة أجر الموظفين بنسبة معينه أو صرف مكافأة لهم.
- إمكانية استخراج التقارير بأشكال مختلفة مثل الرسوم البيانية وغيرها.

تركيب قاعدة البيانات

تأخذ قاعدة البيانات التركيب التالي (1-12):



شكل(1-12)

من هذا الشكل نلاحظ أن:

- قاعدة البيانات مثلا العاملين تتكون من جداول مثل جدول البيانات الأساسية - جدول الإجازات - جدول الدورات. [TABLES]
- والجدول (TABEL) يتكون من سجلات (RECORDS) حيث يمثل كل موظف داخل الجدول بسجل ويسمى أحيانا ROW أي صف.
- والسجل (RECORD) يتكون من حقول (FIELDS) حيث يتكون سجل الموظف من حقول مثل كود الموظف، اسم الموظف العنوان، التلفون، تاريخ الميلاد ..إلخ ، أي تأخذ قاعدة البيانات التركيب التالي :

DATA BASE –TABLES – RECORDS- FIELDS

كيفية استعمال MS ACCESS كمثال بسيط

بعد شرح مقدمات عن قاعدة البيانات وأجزائها نعال معنا نشاهد ذلك عملياً باستعمال البرنامج MS ACCESS وهو أحد البرامج الموجودة في حزمة البرامج (MICROSOFT OFFICE) والتي أنتجتها شركة مايكروسوفت ولا يخلو جهاز كمبيوتر الآن من هذه المجموعة ، ووظيفة برنامج MS ACCESS إنشاء قواعد البيانات والتعامل معها

فإذا كنت تجيد استعماله انتقل إلى الخطوة التالية .

خطوات الاستعمال

قم بتشغيل برنامج MS ACCESS تحصل على أول شاشة كما بالشكل التالي (2-12):

شكل(2-12)

في هذا الشكل يعرض لك البرنامج ثلاثة اختيارات هي:

- قاعدة بيانات فارغة BLANK DATA BASE.
- وفيها لا يوفر البرنامج أي مساعدات وعليك أنت إنشاء كل شيء حسب رغبتك كما سبق شرح ذلك.
- معالج قاعدة البيانات DATA BASE WIZARD.
- وفيها يوفر البرنامج معالج عبارة عن مجموعة من الخطوات تساعدك في إعداد قاعدة بيانات حسب مجموعة كبيرة من قواعد البيانات المقترحة.
- فتح قاعدة بيانات موجودة OPEN EXISTING DATA BASE.
- وفيها يتم فتح ملف قاعدة بيانات موجودة بالفعل وتم إنشائها من قبل.
- وللسهولة ابدأ بالاختيار الثاني وهو DATA BASE WIZARD (معالج قواعد البيانات) ثم انقر الزر OK.

- تحصل على مجموعة من قواعد البيانات الجاهزة.
- فى هذا الشكل تجد مثلاً قاعدة بيانات بالاسم ADDRESS BOOK وهى قاعدة بيانات لعناوين وتليفونات الأصدقاء، وتجد أيضا قاعدة بيانات بالاسم INVENTORY CONTROL وهى خاصة بنظام مخزون وإضافة، وقاعدة بيانات بالاسم STUDENTS & CLASSES وهى خاصة بمتابعة فصل تدريب به طلبه والكثير من قواعد البيانات الأخرى.

دعنا نجرب إحداها ولتكن ADDRESS BOOK كما يلى:

1- اختر ADDRESS BOOK ثم انقر الزر OK.

2- تحصل على مربع لكتابة اسم لها وافق على الاسم الموجود ثم انقر CREATE إنشاء.

3- تحصل على الشاشة الأولى من المعالج.

تابع خطوات المعالج بتحديد البيانات التى سوف توضع فى قاعدة البيانات لتضيف أو تحذف منها ما تريد.

- ثم اضغط الزرالتالى (NEXT) حتى تحصل على شاشة لكتابة عنوان انقر الزر NEXT.

فى النهاية انقر الزر إنهاء (FINISH) فيتم إنشاء قاعدة البيانات وتشغيلها كما فى الشكل التالى:

- تظهر الشاشة الرئيسية لقاعدة بيانات العناوين والتليفونات.
- انقر الزر إدخال / عرض العناوين تحصل على الشاشة الأساسية لتسجيل البيانات.
- جرب استعمال هذه الشاشة فى تسجيل بيانات تهمك ثم إغلق الشاشة تلاحظ العودة إلى الشاشة الرئيسية.
- انقر الزر معاينة التقارير تحصل على شاشة فرعية.
- اختر أي تقرير تلاحظ عرض البيانات التى أدخلتها من قبل.
- جرب استعمال باقى اختيارات البرنامج.
- اغلق البرنامج وعد إلى برنامج النوافذ.

النوع
نص TEXT
رقم NUMBER
تاريخ / وقت
عملة
ترقيم تلقائي
نعم/ لا
كائن OLE
ارتباط تشعبي
معالج البحث
مذكرة

الحقل المميز PRIMARY KEY

هو الحقل الذى لن تتكرر قيمته فى الجدول مثل كود الموظف أو الرقم القومى.

خصائص الحقول FIELD PROPERTIES

بالإضافة لأنواع بيانات الحقول يوجد أيضا مواصفات يتم تحديدها لكل حقل لزيادة التحكم فى بيانات هذا الحقل تظهر فى النصف السفلى من الشاشة عند النقر على الحقل فى مربع التصميم كما بالشكل (5-12).

شكل(5-12)

والجدول التالي يعرض الخصائص الأساسية للحقول ومعنى كل واحدة.

الخاصية	المعنى
حجم الحقل	عدد الحروف
عنوان	عنوان يظهر بدلا من اسم الحقل
القيمة الافتراضية	قيمة إذا لم يسجل المستخدم قيمة
قاعدة التحقق من الصحة	شرط للقيمة المدخلة
مطلوب	لابد للمستخدم أن يدخل قيمته
مفهرس	إنشاء جدول فهرس مرتب على أساسه

تدريب على الجداول

من فضلك قم بإنشاء جدول الموظفين السابق EMP2 بتركيب جديد وأنواع حقول جديدة كما يلي:

اسم الحقل	نوع البيانات	عدد الحروف
EMP_NO	رقم NUMBER	-
EMP_NAME	نص TEXT	30
EMP_PHONE	نص TEXT	12
EMP_BDATE	تاريخ/ وقت DATE/TIME	
EMP_PHOTO	كائن OLE	-
EMP_STATUS	نعم/ لا BOOL	-

ما هو ملف الفهرس INDEX؟

هو جدول مصاحب للجدول الأساسي يتم فيه حفظ رقم السجل والحقل المفهرس على أساسه الجدول فمثلاً عند إنشاء فهرس حسب حقل NAME يتم إنشاء جدول فهرس به حقل NAME مرتب ترتيباً أبجديا ومعه رقم السجل.

يستعمل عند البحث بالحقل المفهرس INDEX على أساسه حيث يتم إنشاء هذا الجدول المفهرس من خصائص الحقل وذلك بتحديد خاصية فهرس (INDEXED).

ويوجد من جدول الفهرس نوعين.

- الأول بدون تكرار وبالتالى لا يسمح بتكرار قيمة هذا الحقل
- والثانى بتكرار وفيه يتم ترتيب البيانات حسب الحقل مع السماح بتكرار قيمته

التكامل فى قواعد البيانات INTEGRITY

من الخصائص التى تتمتع بها البيانات فى قواعد البيانات DATABASES التكامل فى قواعد البيانات والذى يتم على أساسه قياس مدى الدقة ACCURACY والصحة CORRECTNESS ونتناول ذلك من خلال النقاط التالية:

أ- المفاتيح KEYS:

ماهى المفاتيح KEYS وما هى أنواعها وما هى أهميتها.

ب- القيود التكاملية INTEGRITY CONSTRAINTS:

نتناول فى هذه الفقرة التعريف الخاص بالقيود CONSTRAINTS وأهميتها وأنواعها وتطبيق ذلك على بيانات فعلية.

ج- خرق العمليات فى قاعدة البيانات OPERATION VIOLATION

ونعرض فيها كيف يتم خرق القيود CONSTRAINTS من خلال العمليات المختلفة وأساليب التغلب عليها و التعامل معها.

أ- المفاتيح KEYS:

المفاتيح KEYS من الموضوعات الهامة فى قواعد البيانات DATABASES وذلك لكونها العنصر المتحكم فى العلاقات RELATIONSHIPS بين الجداول وبعضها البعض.

وتأخذ الانواع التالية:

1- المفتاح الرئيسى(المميز) PRIMARY KEY.

2- المفتاح الخارجى FOREIGN KEY.

3- المفاتيح المرشحة لتكون مفاتيح رئيسية CANDIDATE KEYS.

4- المفتاح SUPER KEY.

1. المفتاح الرئيسى PRIMARY KEY :

المفتاح الرئيسى هو أحد حقول الجدول TABLE يحدد على أساس الحقل الذى لا تتكرر أى قيمة من قيمه ، أى تكون له قيم وحيدة UNIQUE VALUES ، وأيضا لا يسمح بترك قيمة هذا الحقل خالية NULL VALUE.

مثلا، حقل الرقم القومى فى جدول المواطنين حيث لا يمكن أن يأخذ أكثر من مواطن نفس الرقم القومى.

2. المفتاح الخارجى FOREIGN KEY :

المفتاح الخارجى FOREIGN KEY الخاص بجدول TABLE ما هو عبارة عن حقل FIELD يشير إلى حقل آخر(هو المفتاح الرئيسى PRIMARY KEY) فى جدول آخر مع مراعاة أن تكون قيمه موجودة فى الرئيسى.

ويظهر ذلك فى جدول الموظفين حيث يوجد حقل كزود الإدارة DEPTNO الذى يعتبر FOREIGN KEY حيث يوجد هذا الحقل كمفتاح رئيسى PRIMARY KEY فى جدول الإدارات.

3. المفاتيح المرشحة لتكون مفاتيح رئيسية CANDIDATE KEYS :

بفرض إننا لدينا الجدول R. فمن تعريف الجدول، إذا كان هناك أكثر من حقل يمكن أن يكون مفتاح رئيسى PRIMARY KEY بمعنى أن كل واحد منهم من الممكن أن يتم تحديد الصفوف الخاصة بالجدول من خلاله . وهذه الحقائق تدفعنا الى تكوين التعريف الخاص بالمفتاح المرشح الذى هو كالآتى:

بفرض أن K هى مجموعة خصائص بيانات خاصة بالعلاقة R . إذن K يكون مفتاح مرشح للعلاقة R وذلك إذا وفقط إذا كانت تتمتع بالصفات الآتية:

- التفرد: حيث ان كل قيم R لا تحتوى لا صفرين لهم ذات القيم.
- عدم القدرة على تجزئتها: لا يمكن عمل مجموعة فرعية من K يمكنها ان تتمتع بخاصية التفرد.

4. المفتاح SUPER KEY

إن المجموعة الكلية من المفتاح المرشح هو الـSUPER KEY .

مثلا، فان مجموعة خصائص البيانات (S#, CITY) هى SUPER KEY للعلاقة S . ان SUPER KEY يمتاز بخاصية التفرد لكن ليس من الضرورى ان يمتاز بخاصية عدم القدرة على تجزئته (وخاصة ان المفتاح المرشح هو حالة خاصة من الـSUPER KEY).

ب- القيود التكاملية INTEGRITY CONSTRAINTS:

والقيود التكاملية هى القواعد التى تتخذ لضمان صحة البيانات والعمليات التى تتم عليها مثل الحذف DELETE والادخال INSERTION و التعديل UPDATE و ذلك فى قواعد البيانات DATABASES.

ومن هذه القيود ما يلى:

1- قيد التكامل فى وحدة البياناتENTITY INTEGRITY CONSTRAINT

2- قيد النوع TYPE CONSTRAINT

3- قيد القيمة الخاليةCONSTRAINTS ON NULL

4- قيود الخصائص ATTRIBUTE CONSTRAINTS

5- القيود المتعلقة بقاعدة البيانات DATABASE CONSTRAINT

6- القيود الراجعة للتكامل REFERENTIAL INTEGRITY CONSTRAINT

7-القيد الخاص بوحدة الجدولUNIQUE CONSTRAINT

8-قيد التحقق من شرط CHECK CONSTRAINT

1- قيد التكامل فى وحدة البياناتENTITY INTEGRITY CONSTRAINT:

ويحدد قيد التكامل فى وحدة البيانات ENTITY INTEGRITY CONSTRAINT أنه لا يوجد مفتاح رئيسى PRIMARY KEY يمكن أن تكون أحد قيمه بـ NULL .

وذلك لأن قيمة المفتاح الرئيسى PRIMARY KEY هى التى تحدد كل صفTUPLE على حدة فى الجدول TABLE حيث إنه عندما يأخذ المفتاح الرئيسى PRIMARY KEY قيمة بـNULL فلا يمكن هنا تحديد بعض الصفوف.فعلى سبيل المثال إذا كان هناك صفين أو أكثر يحتويان على القيمة NULL فى المفتاح الرئيسى PRIMARY KEY فلا يمكننا التمييز بينهما.

2. قيد النوع TYPE CONSTRAINT:

ويحدد قيد النوع القيم المتاحة LEGAL VALUES والتى يمكن تسجيلها فى الحقل ، فليس من المنطقى أن نحدد نوع البيانات DATA TYPES والخاصة بحقل معين على إنها أرقام NUMBERS ثم نقوم بإدخال نص عليها.

3. قيد القيمة الخالية CONSTRAINTS ON NULL:

ويحدد هذا القيد ما إذا كان حقل من حقول الجدول يسمح بوجود NULL VALUE فى أحد قيم صفوفه (أو أكثر) أم لا.

فعلى سبيل المثال فى جدول STUDENT إذا كان حقل الـ NAME لا يسمح بوجود قيمة خالية NULL VALUE إذن فهذا هو القيد على القيمة الخالية NULL VALUE .

4. قيود الخصائص ATTRIBUTE CONSTRAINTS:

تتعلق قيود الخصائص ATTRIBUTE CONSTRAINTS بأنه لابد من أن يتم تحديد نوع البيانات المرتبطة بالحقل FIELD بمعنى آخر أنه لابد من أن يكون الـ FIELD من نوع محدد.

5. القيود المتعلقة بقاعدة البيانات DATABASE CONSTRAINT:

قيد قاعدة البيانات هو القيد الخاص بالتداخل بين جدولين أو أكثر

6. القيود التكاملية المرجعية REFERENTIAL INTEGRITY CONSTRAINT

تحدد القيود التكاملية المرجعية REFERENTIAL INTEGRITY CONSTRAINT بين جدولين TWO TABLES وذلك لضمان اتساق صفوف هذين الجدولين ، بمعنى أن الصف فى أحد الجدولين يشير إلى صف آخر موجود فى الجدول الأخر.

7. القيد الخاص بوحدة الجدول UNIQUE CONSTRAINT

ويعنى صفة التفرد أو عدم التكرار UNIQUE ، فكل منها يستطيع أن يكون محدد لباقى الصفوف و الأعمدة ولكن لابد وأن يكون هناك مفتاح رئيسى واحد وباقى الحقول FIELDS تحدد قيد CONSTRAINT خاص بها يسمى الـ UNIQUE CONSTAINT ، وذلك

جدول الموظفين EMP

اسم الحقل	نوع الحقل
EM-NO	TEXT
EM-NAME	TEXT
EM-PHONE	TEXT
EM-ADDRESS	TEXT
EM-MOBILE	TEXT

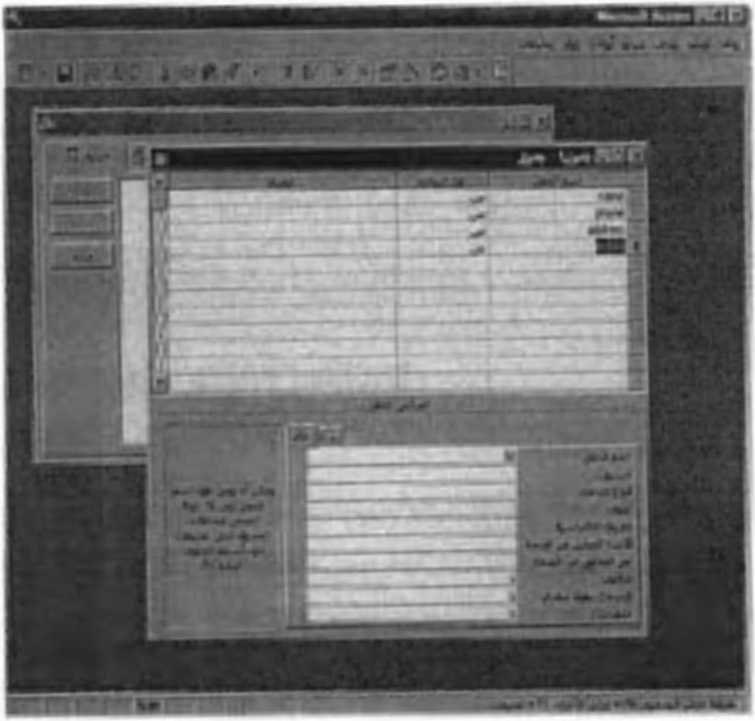
حاول إنشاء هذه الجداول للتدريب على خطوات إنشاء الجدول.

أنواع بيانات الحقول DATA TYPES

بعد إنشاء جدول نتعرض لأنواع بيانات الحقول ، فبعد كتابة اسم الحقل لابد من تحديد نوع الحقل حتى يمكن إدخال بيانات تناسب الحقل ، فمثلا عند تحديد نوع الحقل تاريخ (DATE) ، لابد على المستخدم من إدخال قيمة تاريخية سليمة في هذه الخانة ، ولمشاهدة أنواع البيانات أرجع إلى صفحة التصميم بنقر اسم الجدول ثم نقر الزر تصميم DESIGN ثم انقر نوع الحقل كما بالشكل التالي (4-12).

شكل(4-12)

والجدول التالي يعرض أنواع بيانات الحقول.



شكل (3-12)

إغلق شاشة تصميم الجدول تحصل على سؤال هل تريد الحفظ اختر نعم واكتب اسماً للجدول وليكن FRIENDS.

بهذا تم إنشاء جدول بالاسم FRIENDS بالتركيب السابق ، انقر اسمه ثم اختر فتح OPEN فيتم فتح الجدول لتسجيل بيانات.

تدريب على الجداول

من فضلك حاول تكرار الخطوات السابقة لإنشاء جدول مماثل وجرب ذلك بإنشاء الجداول التالية:

جدول الإضافة ITEMS

اسم الحقل	نوع الحقل
ITEM-CODE	TEXT
ITEM DESK	TEXT
ITEM – UNIT	TEXT
ITEM MODAL	TEXT
ITEM PRICE	TEXT

جرب استعمال المعالج فى إنشاء قواعد البيانات الجاهزة المختلفة والتى عرضناها مثل:

نظام المخزون	INVENTORY
نادى فيديو	VIDEO COLLECTION
مكتبة الصور	PICTURE LIBRARY
مركز تدريب	STUDENTS & CLASSES

وبهذا تكون قد قدمنا لك مقدمة عن قواعد البيانات والبرنامج MS ACCESS .

وفى الفقرات التالية نراجع معا بعض المفاهيم أساسية .

عناصر قاعدة البيانات

1ـ الجدول TABLE

الجدول هو أول عنصر يتم إنشاؤه فى قاعدة البيانات ، فهو العنصر الوحيد الذى يحتوى على البيانات ، وعليه تبنى باقى عناصر قاعدة البيانات ، وقاعدة البيانات عبارة عن مجموعة جداول ، والجدول عبارة عن مجموعة سجلات و السجل عبارة عن مجموعة حقول.

النماذج FORMS

النموذج (FORM) هو العنصر الثانى ، وهو عبارة عن شاشة إدخال للبيانات تصب فى الجدول ، ويتم إنشاء النموذج حتى لا يقوم المستخدم بتسجيل البيانات مباشرا فى الجدول ، والنموذج هو الشكل الذى يراه المستخدم ويتعامل معه فلابد أن يوفر الشكل الجيد والخطوات السهلة لتسجيل البيانات.

الاستعلام QUERY

الاستعلام (QUERY) عنصر من عناصر قاعدة البيانات الهامة وهو عبارة عن عنصر يعرض بيانات معينه تظهر من جدول أو أكثر بشروط معينة ، فمثلا يمكنك تكوين

استعلام بيانات العاملين الخاصين بقسم معين ، أو تكوين استعلام بالعاملين فى منطقه معينة.

التقرير REPORT

التقرير هو الطريقة الأساسية لعرض البيانات فى شكل تقرير مع إمكانية طباعته، و يمكن بناؤه على جدول أو استعلام.

الماكرو MACROS

الماكرو هو طريقه مختصره لتنفيذ عمليه معينة بشكل سريع مثل إنشاء ماكرو لفتح نموذج (FORM) ، أو ماكرو لفتح استعلام فهو يختصر كثيرا من الخطوات.

الوحدة النمطية MODULE

هى وسيلة لكتابة الأوامر بلغة VBASIC والتى تسمى VBA لتحقيق عمليات معينة لا تتحقق باستعمال الماكرو .

بعد عرض عناصر برنامج MS ACCESS بشكل مختصر تعالى معنا نتابع شرح هذه العناصر.

إنشاء الجداول TABLES

كما ذكرنا أن الجدول هو أهم عنصر فى قاعدة البيانات فهو الذي يحتفظ بالبيانات، وللقيام بتصميم جدول نفترض أن بياناته تخص الأصدقاء وهى عبارة عن الاسم ، والعنوان ، والتليفون ، والمحمول طبقا للخطوات التالية :

- قم بتشغيل برنامج MS ACCESS.
- انقر صفحة TABLE جدول.
- انقر زر جديد NEW.
- اختر عرض تصميمى DESIGN VIEW وقم بانشاء جدول.
- فى خانه اسم الحقل FIELD NAME اكتب NAME وفى السطر الثانى اكتب PHONE ثم ADDRESS ثم MOBILE كما فى الشكل (3-12).

القيد الـ CONSTRAINT موجود في معظم قواعد البيانات اختيار OPTION يتم اختياره عند إنشاء الجدول.

8. قيد التحقق CHECK CONSTRAINT:

هذا القيد هو عبارة عن شرط CONDITION لابد من حدوثه ، ويتم تحديده في مرحلة إنشاء قاعدة البيانات DATABASE ونجد أنه يعرف عن طريق اختيار أيضا في قاعدة البيانات أو من الممكن تحديده عن طريق اللغة الأساسية في التعامل مع قواعد البيانات SQL .

جـ. خرق العمليات في قاعدة البيانات OPERATION VIOLATION

العمليات الأساسية التي نقوم بها في قاعدة البيانات DATABASES هي الإدخال INSERTION والحذف DELETION و التعديل UPDATE، وقد يحدث عند إجراء بعض العمليات خرق VIOLATION للقيود الموضوعة CONSTRAINTS لذا وجب دراسة أنواع خرق القيود و طرق التعامل معها و تجنبها .

وسوف نتناول في هذا الجزء الآتي:

1- عملية الادخال INSERT OPERATION

2- عملية الحذف DELETE OPERATION

3- عملية التعديل UPDATE OPERATION

عند عملية الإدخال قد يحدث خرق للقيود الآتية والموضوعة على قاعدة البيانات DATABASE:

- فبالنسبة لقيد النطاق DOMAIN CONSTRAINT، يتم خرقه إذا كانت القيمة الموضوعة في الحقل FIELD لا توجد في النطاق المحدد DOMAIN.
- وبالنسبة للقيود الموجودة على المفاتيح KEY CONSTRAINTS نجد أنه يتم خرقها إذا كانت قيمة المفتاح في صف معين في جدول ما موجودة في نفس الجدول.
- وبالنسبة للقيد الموجود على وحدة البيانات ENTITY INTEGRITY CONSTRAINT إذا كان المفتاح الرئيسي PRIMARY KEY يحتوي على NULL.

- وبالنسبة لقيد التكامل المرجعي REFRENTIAL INTEGRITY CONSTRAINT نجد أنه يتم خرقه إذا كانت القيمة المراد ادخالها في المفتاح الخارجي FOREIGN KEY ليست موجودة في المفتاح الرئيسي PRIMARY KEY الذي يشير إليه المفتاح الخارجي FOREIGN KEY.

2. عملية الحذف DELETE OPERATION:

القيد الوحيد الذي يمكن خرقه في عملية الحذف DELETE هو قيد التكامل المرجعي REFRENTIAL INTEGRITY CONSTRAINT، فإذا تم حذف صف من جدول TABLE كان يشير إليه مفتاح خارجي من جدول آخر إذن فهذا خرق لقيد التكامل المرجعي REFERENTIAL INTEGRITY CONSTRAINT، وسيتضح ذلك من خلال الأمثلة الآتية:

1- عندما نحذف الصف الموجود في الجدول WORKS_ON والذي فيه:

ESSN=999887777 وفيه DNO=10

نجد هنا أن عملية الحذف ستقبل.

2- عندما نحذف الصف الموجود في الجدول EMPLOYEE والذي فيه SSN = 999887777

نجد هنا أن عملية الحذف لن تقبل و ذلك لأن هناك صفوف في الجدول WORKS_ON تشير إلي هذا الصف.

وهناك طريقة حل بالنسبة لحالات الخرق VIOLATION الناتجة عن عملية الحذف و هي CASCADING OPTION وهو موجود في اللغة الأساسية SQL و موجود بصورة مباشرة أو غير مباشرة في كل قواعد البيانات فعلى سبيل المثال في قواعد بيانات ACCESS نجد أن هناك اختيارا OPTION عند عمل العلاقة بين جدول وآخر وهو CASCADE DELETE RELATED RECORDS والذي يقوم بحذف كل المفاتيح الخارجية FOREIGN KEYS التابعة لمفتاح رئيسي PRIMARY KEY قد تم حذفه.

3. عملية التعديل UPDATE OPERATION :

تستخدم عملية التعديل في تغيير القيم لخاصية أو أكثر في صف أو صفوف، لذلك

فمن الضرورى أن يكون هناك شرط معين لصحة عملية التعديل حتى لا يحدث خرق للقيود CONSTRAINTS.

2- عدل الخاصية DNO بحيث تساوى7 فى الموظف الذى فيه خاصية ال SSN تساوى 999887777.

نجد أن عملية التعديل هنا لن تقبل وذلك لأنها خرقت قيد التكامل المرجعى REFERENTIAL INTEGRITY CONSTRAINT.

3- عدل الخاصية SSN فى الموظف الذى فيه الخاصية SSN تساوى 999887777 إلى 987654321.

وسنجد أيضا أنها لن تقبل لأنها خرقت المفتاح الرئيسى PRIMARY KEY و قيد التكامل المرجعى REFERENTIAL INTEGRITY CONSTRAINT.

وسنجد أيضا أنه كما فى عملية الحذف DELETE كان هناك طريقة لتجنب الخرق الذى يمكن أن تحدثه عملية الحذف نجد أن هناك طريقة مشابهة لتجنب الخرق الذى يمكن أن تحدثه عملية التعديل UPDATE

تصحيح تصميم الجداول NORMALIZATION

نتناول كيفية تصحيح تصميم الجداول أو بمعنى آخر سوف نتناول المراحل التى يمكننا من خلالها الوصول إلى نظام قاعدة بيانات صحيح قائم على تصميم جيد للجداول وذلك من خلال النقاط التالية:

1- المراجعة الاولى لتصميم قاعدة البيانات FIRST NORMAL FORM (FNF).

2- المراجعة الثانية لتصميم قاعدة البيانات SECOND NORMAL FORM (SNF).

3- المراجعة الثالثة لتصميم قاعدة البيانات THIRD NORMAL FORM (TNF).

المراجعة الأولى لتصميم قاعدة البيانات: FIRST NORMAL FORM (FNF)

وفيه يتم مراجعة تركيب الجدول بحيث لا يحتوى الجدول على أى حقول مركبة ولتوضيح ذلك تابع تصميم الجدول الموجود فى الشكل (6-12).

EmpNo	EName	Project	
		ProjNo	ProjHours

EmpNo	EName	ProjNo	ProjHours
1	Mohamed	10	33
1	Mohamed	20	77
1	Mohamed	30	88
2	omr	55	74
3	Nada	61	65
4	Esraa	34	69

شكل (6-12)

ولحل هذه المشكلة نقوم بتطبيق قاعدة النموذج المحسن الأول 1NF بتقسيم الجدول إلى جدولين يصبح كما فى الشكل (7-12).

EmpNo	EName

EmpNo	ProjNo	ProjHours

شكل(7-12)

وبالتالى لا تتكرر البيانات بل تصبح كما فى الشكل (8-12)

EmpNo	EName
1	Mohamed
2	omr
3	Nada
4	Esraa

EmpNo	ProjNo	ProjHours
1	10	33
1	20	77
1	30	99
2	55	74
3	61	65
4	34	69

شكل(8-12)

وهذا ما يسمى1NF اى النموذج المحسن الأول أما النموذج المحسن الثانى (SNF) يتم فيه مراجعة نتيجة 1NF فإذا كانت هناك حقول لا تعتمد على الحقل PRIMARY KEY اعتماد كامل أى لا يصلح لأن يميزها تفصل هذه الحقول وينشأ لها وحدة جديدة ENTITY و النموذج المحسن الثالث TNF يتم فيه مراجعة SNF فاذا كانت هناك حقول تعتمد على حقول أخرى يتم فصلها فى وحدات ENTITIES أخرى وسوف يتم تناول ذلك فى بالتفصيل فى فصل منفصل.

ما هى لغة الاستعلامات SQL

الحروف SQL اختصاراً للعبارة Structure Query Language أى لغة الاستعلامات المركبة وهى عبارة عن لغة (مجموعة من الأوامر والدوال) للتعامل مع قواعد البيانات حيث توفر كيفية انشاء وتعديل الجدول وتوفر الاستعلام بشتى أنواعه ، ولا يكاد يوجد برنامج قواعد بيانات ألا ويستعمل اللغة SQL مثل برنامج Access وبرنامج Informix و Oracle وكذلك لغات البرمجة مثل Delphi, ++VC visual basic ويفضل لأي مصمم برنامج قواعد بيانات أن يلم بقواعد لغة SQL لما توفره من إمكانيات كثيرة بأوامر قليلة .

أجزاء لغة الاستعلامات SQL

تنقسم لغة SQL إلى الأجزاء التالية :

1. لغة التعريفات DDL (Data Defination Language)

هو الجزء المسئول عن إنشاء عناصر ملف قواعد البيانات مثل إنشاء وتعديل وحذف الجداول والفهارس وغيرها .

2. لغة التحكم DCL (Data Control Language)

هى مجموعة من الجمل المفيدة عند تصميم برنامج متعدد للمستخدمين فهى، فتوفر التحكم فى درجة السماحية لكل مستخدم للبرنامج وكذلك الاستعلام عن البيانات.

3. لغة الوصول للبيانات DML (Data Manipulation Language)

وهى أهم الأجزاء حيث توفر معظم العمليات المطلوبة من قواعد البيانات مثل الاستعلامات وإضافة وحذف البيانات للجداول وفيما يلى تفصيل كل جزء من هذه الأجزاء.

لغة التعريفات DDL

هى مجموعة الجمل المسئولة عن إنشاء وحذف عناصر قواعد البيانات مثل الجداول والفهارس وغيرها .

ويتكون هذا الجزء من الكلمات الثلاثة التالية :

Create : لإنشاء عنصر جديد مثل جدول بيانات او ملف فهرس Index مع تحديد مواصفات حقول الجدول

Alter : للتعديل فى عنصر موجود (جدول ، فهرس او حقل فى جدول) و يقصد هنا بالتعديل في بناء الجدول و ليس البيانات الموجودة داخله.

Drop : حذف عنصر موجود (جدول ، فهرس ، .....)

أنواع البيانات

قبل شرح كيفية استعمال الأوامر السابقة مثل إنشاء الجدول ، نقوم أولاً بعرض أنواع البيانات التى نحتاجها عند تصميم الجدول والمتاحة فى لغة SQL والجدول التالى يعرض هذه الأنواع ومعنى كل نوع.

اسم النوع	المعنى
Smallint	حقل رقم صحيح طوله 2 بايت
Integer	حقل رقم صحيح طوله 4 بايت
Real	حقل رقم حقيقي طوله 4 بايت
Bit	حقل من نوع بت واحد
Tinyint	حقل رقم صحيح طوله 1 بايت
Bigint	حقل رقم صحيح طوله 8 بايت
Binary (n)	حقل ثنائي (Binary) ثابت الطول بعدد n من البايت
Varbinary(n)	حقل ثنائي متغير الطول بحد أقصى n من البايت
Date	حقل من نوع تاريخ
Time	حقل من نوع وقت
Timestemp	حقل يحتوى على تاريخ ووقت
Varchar (n)	حقل حرفي بالعدد n من الحروف .
Number(p,d)	حقل رقمي حقيقي عدد خاناته p وبعد العلامة d من الأرقام

جدول بانواع البيانات المستعملة فى SQL

إنشاء الجداول Create Table

غالباً ما يتم إنشاء وتعديل وحذف الجداول باستخدام أحد برامج قواعد البيانات مثل برنامج Access ، ولكن تحتاج أحياناً بعض العمليات استخدام الأوامر لإنشاء الجداول ، ويتم إنشاء الجدول باستعمال الأمر Create حيث يأخذ الصورة العامه التالية :

```
Create Table Table_Name ( Column - list - .....data types )
```

وفى هذه الصيغة يتم استعمال الأمر Create ثم كلمة Table لإنشاء جدول بالاسم Table_name وتركيب هذا الجدول هو Column List أي حقول الجدول ، وأنواع الحقول هى Data types ويوضح ذلك المثال التالى :-

```
Create Table Employee ( Emid   Integer,Emname Varchar ( 60),
Salary          Numeric ( 8,2 ),
Dept            Varchar (12 )
```

فى السطور السابقة تم إنشاء جدول بالاسم Employee ويحتوى على أربعة حقول أسمائها DEP,NO , NAME , TEL

وأمام كل حقل نوع هذا الحقل ، فالحقل الأول مثلاً من النوع Integer اى رقم صحيح وهكذا والثاني حرفي ويمكن تحديد شروط لمواصفات كل حقل مثل عدم احتواء الحقل على NULL. بكتابة ذلك أمام الحقل ولتوضيح ذلك نعدل المثال السابق ليصبح كما يلى :-

مثال :

```
Create Table Employee ( NO     Integer, not null,
NAME            Varchar ( 60), not null,
TEL             Nmmeric ( 8,2 ),
Dept            VARCHAR  (12 0)      Defautt      "Eng")
```

فى السطور السابقة يتم وضع العبارة NOT NULL أمام الحقل الأول والثاني وذلك لعدم السماح بتركهم فارغين (NULL) .

وفى السطر الأخير تم وضع الكلمة Default لوضع قيمة افتراضية للحقل Dept وهى الكلمة .Eng وهناك كثير من الكلمات التي يمكن استخدامها لتحديد مواصفات الحقل فمثلاً يمكن استعمال الكلمة Unique لعدم تكرار قيمة حقل معين.

حذف الجداول باستعمال الأمر Drop

ويأخذ الصورة العامة التالية:

```
Drop Table Table-Name
```

وفيها المتغير Table-Name هو اسم الجدول المطلوب حذفه ويقوم هذا الأمر بحذف الجدول وما فيه من بيانات ، أما حذف البيانات فقط فيكون باستعمال الأمر Delete كما سيرد فيما بعد.

تعديل مواصفات جدول Alter Table

أحيانا نحتاج لتعديل مواصفات جدول - مثل إضافة حقل إلى الجدول أو حذف

حقل غير مرغوب فيه ، ويتم ذلك باستعمال الأمر Alter حيث يأخذ الصورة العامة التالية :

```
Alter Table   Tabl-name   Add Column
Column - name     data type
```

وفي هذه الصورة تم استعمال كلمة Add Column لإضافة حقل جديد وهذا هو التعديل المطلوب.

ولتوضيح ذلك نقوم بإضافة حقل جديد للجدول السابق كما يلي :-

```
Alter table Employee Add Column (ADDRESS,VARCHAR)
```

في هذا المثال يتم إضافة حقل جديد بالاسم Address وهذا هو التعديل.

```
Alter table Employee Drop Column Bdate
```

وفي هذا السطر يتم حذف الحقل باستخدام الكلمة Drop.

ملفات الفهارس Indexes

هي ملفات مصاحبة للجداول تزيد من سرعة الوصول (البحث) والاستعلام عن السجلات :

- وهي عبارة عن ترتيب(فرز) الجدول حسب حقل أو أكثر ، وغالباً ما يتم إنشائها عند إنشاء الجدول.
- و يمكن إنشائها باستعمال الأمر CREATE .

كما يلي:

```
Create ( Unique ) Index  Index - name
On Table -name ( Column - nam (s) { Asc, desc }
```

والمثال التالي يوضح طريقة أستعمال هذه الصيغة :

```
Create Unique Index  Empindx

ON  EMployed ( Emid )
```

وفي هذا المثال يتم إنشاء فهرس مرتب حسب الحقل Emid وذلك بالاسم Empindex

ويتضح ذلك في المثال التالي :

```
Create Index Empindx2 ON  Eployee  ( Emname , Dept )
```

وفى هذا المثال تم بناء ملف الفهرس على أساس الحقلين (Emname , Dept)
ويتم حذف ملف الفهرس بالأمر DROP.

```
Drop Index  Index - name
```

لغة التحكم DCL

كما أشرنا أن هذا الجزء يقوم بتحديد من المسموح له باستعمال هذه البيانات ودرجة السماحيه ، هل هى للاستعلام أم للإضافة أم للحذف أم للتعديل ، ولتوضيح ذلك نبدأ بالمثال التالى :

```
Grant Select ON Employee To Public
```

هذا المثال يسمح لجميع المستخدمين باستعمال الجدول ، حيث يبدأ بالكلمة Grant وهى سماحية ثم Select لتحديد السماحية للاستعلام وختم بـ To Public أى عام. أما أذا أردنا تحديد الإضافة والاستعلام نستعمل الكلمتين Insert , Select ثم تحديد المستخدم المسموح له بذلك بالعبارة TO Hamdy,Samy.
ويمكن تحديد حقول معينة لأشخاص معينة كما فى المثال التالى:

```
Grant UpDate ( Salary , Dept ) ON Employee
TO Nabil , Azab
```

فى هذا المثال يتم تحديد سماحية تحديث حقلين بالعبارة (Up date (Salary , Dept وذلك لشخصين هما Azab , Nabil ومن هذا يتضح أهمية هذا الجزء عند تصميم برنامج متعدد المستخدمين:

لغة الوصول للبيانات DML

يعتبر هذا الجزء من لغة SQL من أهم الأجزاء ويتكون من الأوامر التالية:

```
Select
Insert
Delete
Update
```

الأمر Select :

يعتبر هذا الأمر هو قلب لغة SQL فهو الذى يوفر اختيار بيانات مختلفة من جدول أو أكثر ، والصيغة العامة له بسيطة ولكن يمكن إضافة معاملات لها تزيد من قوتها وتأخذ

هذه الصيغة الصور التالية :

```
Select    Select-List    From    Table-name
```

وفيها المتغير Select-List عبارة عن الحقول المطلوبة ، ويمكن أن تأخذ صور بسيطة كاختيار جميع حقول جدول (*) أو حقول مختارة من أكثر من جدول، والمتغير Table-name هو اسم الجدول . وفيها يلى عدة صور لاستخدامها.

الصورة الأولى :

```
Select * From My table
```

فى هذه الصورة يتم اختيار جميع الحقول وذلك بالعلامة (*) من الجدول My table.

الصورة الثانية :

```
Select Emid , Emname From My table
```

فى هذه الصورة تم اختيار حقلين Emid , Emname من نفس الجدول My Table

الصورة الثالثة :

```
Select Emid , "....." Emname From My table
```

فى هذه الصورة تحديد صورة تحديد قيمة الحقل التالى للحقل Emid بالقيمة " ....." وبالتالى يتم اختيار الحقول التى قيمتها " " وهى مفيدة فى كثير من التطبيقات لعرض السجلات التى تحتوى على قيمة معينة.

العامل Where

هذا الأمر مفيد جداً فى تحديد حقول معينة (شرط معين) باستعمال مؤثرات المقارنة (=، < >، = < ، > = ، < ، >) لتحديد الشرط المطلوب وفيها يلى عدة صور لاستعمالها.

الصورة الأولى :

```
Select * from Employee where Emid=123
```

فى هذه الصورة يتم اختيار جميع حقول الجدول Employee التى يساوى حقل Emid فيها القيمة 123 ويتم عرض سجل واحد وهو الموظف رقم 123.

الصورة الثانية :

```
Select * From Employee Where Salary > 500
and Salary <900
```

وفى هذه الصورة كان الشرط أن تقع قيمة الحقل Salary.

بين القيمتين 500 و 900 وبالتالى يتم عرض الموظفين التى تقع مرتباتهم بين 500، 900

الصورة الثالثة :

```
Select * From Employee Where Dept = ( Eng) OR Dept = ( Mid )
```

الصورة الرابعة : يتم استعمال المؤثر المنطقى OR.

```
Select * From Employee Where Not Dept = ( tec )
```

فى هذه الصورة كان الشرط لاختيار السجلات هو أن لا يساوى الحقل Dept القيمة TEC وهو استعمال المؤثر المنطقى NOT.

ويمكن كتابة الشرط بالشكل `Where Dept < > " Tec "`

```
Select * From Employee where Dept != Tec
```

التعبير الرياضي (الحسابي)

يمكن استعمال التعبير الحسابي مع الأمر Select أو Where ويظهر ذلك كما يلى :

```
Select  Partno , Pcost + oth cost From Parts
```

فى هذا المثال يتم اختيار الحقل Partno ، ثم اختيار مجموعة الحقلين P Cost – Oth Cost من الجدول Parts ويمكن استعمال التعبير مع Where كما يلى :

```
Select Partno from Parts Where ( Oth Cost / Pcost ) > 0.20
```

وذلك معناه إمكانية استعمال عملية حسابية مكان حقل سواء فى الاختيار أو الشرط.

مجموعة دوال الحرفيات String Functions

هذه المجموعة من الدوال توفر عمليات الحرفيات مثل معرفة طول عبارة حرفية ( ) Length حذف المسافات من العبارة الحرفية وغيرها من الوظائف .

الدوال الرقمية

هى الدوال التى توفر العمليات الحسابية مثل الدوال Asin() ، Acos() ، Atan() ، Cos() ، Log() ، PI() ، Power() وكثيراً من الدوال .

دوال التاريخ والوقت Date&Time Functions

مجموعة من الدوال التى تتعامل مع التاريخ والوقت مثل الدالة Current Date التى تعيد التاريخ الحالى ، الدالة Current-Time() التى تعيد الوقت الحالى .

- الدالة Day of week تعيد رقم اليوم الحالى فى الاسبوع وهو الرقم من 1إلى 7.
- الدالة Day of Month() تعيد رقم اليوم الحالى فى الشهر .
- الدالة Hour() تعيد الساعة وكذلك الدالة Now() وغيرها من الدوال .

عرض بيانات من أكثر من جدول

Displaying Data from multiple tables

ويتم تناول ذلك من خلال النقاط التالية:

- استعمال جملة select لعرض بيانات من اكثر من جدول
- إنشاء علاقة خارجية outer join
- ربط جدول بنفسه joining table to it self

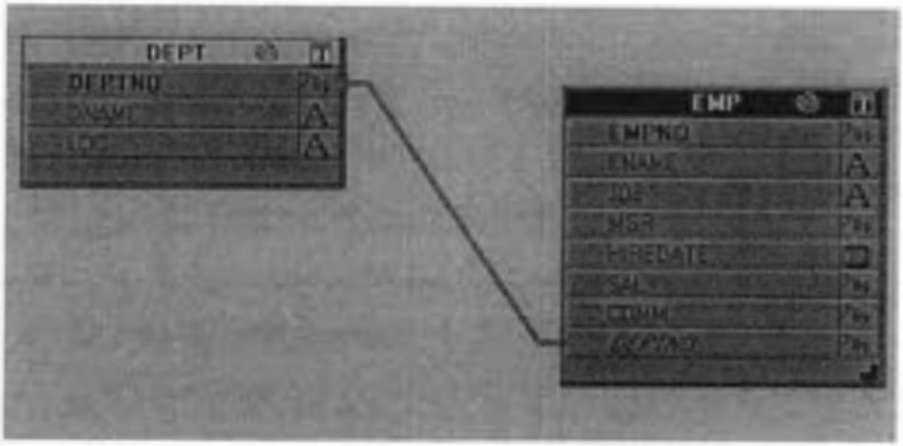
تعريفات

الحقل primary key

هو الحقل الذي يميز الجدول والذي لا يمكن تكرار قيمته مثل كود الموظف في جدول الموظفين ، كود الصنف في جدول الأصناف كود الإدارة فى جدول الإدارات.

الحقل Foreign key

هو حقل أصلاً primary key في جدول أخر يضاف إلي جدول أخر ليمثل بيانات الجدول الأصلي مثل حقل كود الإدارة في جدول الموظفين ومثل كود المنطقة فى جدول الموظفين ولتوضيح الحقلين انظر إلي الشكل(9-12).



الشكل(9-12)

اختيار بيانات من أكثر من جدول

ويتم ذلك عندما يحتوي جدول علي الحقل foreign key الذى هو primary key في جدول أخر كما أشرنا ونريد عرض البيانات الموجودة في الجدول الأصلي و التي تنتمي إلي هذا الحقل .

فمثلاً يحتوي جدول الموظفين علي بيانات الموظفين بما فيها كود الإدارة deptno بينما توجد بيانات الإدارات مثل اسم الإدارة وغيره في جدول الإدارات فنريد عرضها بشرط أن يساوي حقل الإدارة في جدول الموظفين حقل الإدارة في جدول الإدارات ويكتب الأمر كما بالشكل التالي:

```
Select EMP. deptno, Dept. Dname From EMP, Dept
Where EMP. DeptNO =Dept. DeptNO;
```

في هذا المثال تم اختيار الحقل deptno من الجدول EMP لذلك تم وضع . بين اسم الجدول و اسم الحقل.

وتم اختيار الحقل DName من الجدول Dept لذلك تم وضع . بينه وبين جدوله Dept، ثم كتابة from EMP, Dept لأنه بالفعل يختار بيانات من الجدولين وشرط الاختيار where هو أن كود الإدارة في جدول الموظفين يساوي كود الإدارة في جدول الإدارات

وبالتالي يتم عرض الاسم الأول والاسم الثاني من جدول EMP ويتم عرض اسم

الإدارة من جدول الإدارات الذي كود إدارته موجود في جدول الموظفين .

اكتب الأمر ونفذه تحصل على نتيجة التنفيذ كما بالشكل (10-12)

```
DEPTNO DNAME
------ --------------
    20 RESEARCH
    30 SALES
    30 SALES
    20 RESEARCH
    30 SALES
    30 SALES
    10 ACCOUNTING
    20 RESEARCH
    10 ACCOUNTING
    30 SALES
    20 RESEARCH

DEPTNO DNAME
------ --------------
    30 SALES
    20 RESEARCH
    10 ACCOUNTING
```

الشكل (10-12)

يمكن كتابة الأمر السابق بشكل أخر وهو كما يلي :

```
Select E.deptno, D. Dname
From EMP. E, Dept .D
Where E Dept NO=D. Dept NO;
```

في هذا المثال تم وضع الحرف E أمام اسم الجدول EMP.

وضع الحرف D أمام اسم الجدول Dept.

وبالتالي جميع التعاملات مع الجدولين تتم بالتعامل مع الحرفين D,E وهذا أسهل من كتابة أسماء الجداول كل مرة اكتب ونفذ الأمر وشاهد النتيجة .

ملحوظة هامة: عند كتابة جملة SELECT للاختيار من جدولين بدون الأمر Where تحصل على نتائج غريبة حيث يتم تكرار جميع سجلات الجدول الثاني مع كل سجل من الجدول الأول .

العلاقة الخارجية OUTER Join

تقوم هذه العلاقة بعرض البيانات التي ليس لها مقابل في الجدول الثاني عكس ما تم

فى الفقرة السابقة حيث كنا نعرض بيانات الإدارات من جدول الإدارات التى يوجد لها مقابل فى جدول الموظفين ولكن العلاقة الخارجية تمثل العكس فمثلاً لمعرفة : الإدارات التى لم يسكن عليها موظفين أى التى لا يوجد لها مقابل فى جدول الموظفين نستخدم هذه العلاقة كما فى السطور التالية:

```
Select EMP. deptno, Dept. Dname
From  Dept RIGHT OUTER JOIN  EMP;
```

أكتب ونفذ الأمر لتحصل على النتيجة كما فى الشكل (11-12).

```
DEPTNO DNAME
------- --------------
     10 ACCOUNTING
     10 ACCOUNTING
     10 ACCOUNTING
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     30 SALES
     30 SALES
     30 SALES

DEPTNO DNAME
------- --------------
     30 SALES
     30 SALES
     30 SALES
        OPERATIONS
```

الشكل(11-12)

نجد العبارة Right Outer لتعبر عن الطرف الذى تكون بعض بياناته لا تقابلها قيم محددة بل تقابلها القيمة NULL وهذا الطرف هو الجانب الأيمن.

أما إذا أردنا الجانب الأيسر فنجد العبارة Left Outer لتعبر عنه كما يتضح من المثال التالى:

```
Select EMP. deptno, Dept. Dname
From  emp LEFT OUTER JOIN  dept;
```

وتكون النتيجة كما بالشكل (12-12).

```
DEPTNO DNAME
------- --------------
     10 ACCOUNTING
     10 ACCOUNTING
     10 ACCOUNTING
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     30 SALES
     30 SALES
     30 SALES

DEPTNO DNAME
------- --------------
     30 SALES
     30 SALES
     30 SALES
        OPERATIONS
```

الشكل(12-12)

ربط الجدول مع نفسه Jonning a table to itself

أحياناً نحتاج لعرض بيانات من جدول واحد ولكن بشرط يربط بين حقلين من حقوله لتحقيق شرط معين ، واشهر مثال هو جدول الموظفين الذي يحتوي على كود الموظف وكود المدير وفي نفس الوقت هذا الموظف له مدير هو موظف له كود كموظف وفي نفس الوقت يسجل رقمه في حقل المدير ، ويظهر ذلك من الأمر التالي:

```
SELECT a.empid, a.location, a.date, b.location, b.date
FROM test_results a, test_results b
WHERE a.empid = b.empid
AND a.location <> b.location
AND a.date > sysdate-365
AND b.date > sysdate-365;
A.EMPID A.LOCATION A.DATE B.LOCATION B.DATE
------- ---------- --------- ---------- ---------
94839 St. John 04-NOV-97 Wendt 03-JAN-98
04030 Stridberg 27-JUN-97 Wendt 03-AUG-97
```

دوال المجموعات Group functions

دوال المجموعات هي دوال تنفذ على جميع سجلات الجدول أو مجموعة السجلات حسب شرط معين مثل دالة الجمع ()sum أو دالة المتوسط ()AVG وغيره.

وفي هذه الفقرة نتناول ما يلي:

- دوال المجموعات Group functions
- استعمال دوال المجموعات Using Group function
- الأمر Group BY
- تحديد البيانات باستعمال Having

الدوال:

• اسم الدالة: **AVG(x)**

الاستخدام: تستخدم لحساب متوسط قيمة حقل معين مثل متوسط مرتبات الموظفين.

• اسم الدالة: **COUNT(x)**

الاستخدام: تستخدم لحساب عدد سجلات جدول أو عدد سجلات حقل معين.

• اسم الدالة: **MAX(x)**

الاستخدام: تستخدم لمعرفة أقصى (اكبر) قيمة من مجموعة قيم الحقل مثل معرفة أعلى قيمة مرتب.

• اسم الدالة: **MIN(x)**

الاستخدام: تستخدم لمعرفة أقل قيمة من مجموعة قيم لحقل مثل معرفة أقل قيمة مرتب.

• اسم الدالة: **SUM(x)**

الاستخدام: تستخدم لمعرفة مجموع قيم حقل مثل مجموع مرتبات الموظفين او مجموع مرتبات موظفي إدارة.

• اسم الدالة: **VARIANCE(x)**

استخدام دوال المجموعات

• الدوال: SUM() ,AVG() ,MIN() ,MAX()

```
SQL> SELECT     AVG(sal), MAX(sal),
  2       MIN(sal), SUM(sal)
  3 FROM     emp
  4 WHERE   job LIKE 'SALES%';
```

في هذا المثال يتم استعمال مجموعة دوال على حقل Sal من جدول EMP.

اكتب هذا الأمر ونفذه تحصل على نتيجة التنفيذ كما بالشكل (13-12).

```
AVG(SAL)  MAX(SAL)  MIN(SAL)  SUM(SAL)
---------- ---------- ---------- ----------
      1400       1600       1250       5600
```

الشكل (13-12)

• الدالة: count()

```
Select Count (*),
from EMP;
```

يمكن استعمال الدالة Count() بثلاثة أشكال الأول بالعلامة *.

ويتم عد جميع السجلات وهذا أبطأ الاختيارات ،الصورة الثانية Count(1) .

وهو عدد الحقل الأول في الجدول ، والثالث هو Count() مع اسم الحقل empno.

اكتب الاوامر ونفذها تحصل على الشكل (14-12).

```
SQL>  Select Count (*)
  2  From emp;

  COUNT(*)
----------
        14

SQL>
```

الشكل(14-12)

• الأمر Group BY

يستعمل هذا الأمر لتقسيم سجلات جدول إلى مجموعات فمثلاً لعرض إجمالي مرتبات كل إدارة كلاً على حدة ويتضح ذلك من المثال التالي:

```
SELECT  deptno, AVG(sal)
  2 FROM    emp
  3 GROUP BY deptno;
```

في هذا المثال يتم استدعاء مجموعة من دوال المجموعات على حقل المرتب Salary وكذلك عرض حقل كود الإدارة DeptNO من جدول الموظفين بعد تقسيم السجلات إلى مجموعات حسب كود الإدارة، وبالتالي يتم عرض كود الإدارة وأمامه مجموع ومتوسط وعدد سجلات هذه الإدارة . اكتب الأمر ونفذه تحصل على الشكل (15-12) .

```
DEPTNO  AVG(SAL)
--------- ---------
10 2916.6667
20       2175
30 1566.6667
```

الشكل(15-12)

ملحوظة هامة:

في حالة استعمال GROUP BY لا يمكن اختيار حقل فردي بدون دالة غير الحقل الذي يتم عمل GROUP BY له.

فمثلاً في المثال السابق تم اختيار الحقل EMPNO وهو الحقل الذي تم عمل GROUP BY له، ولكن لا يمكن عرض الحقل EName أو غيره ونحن نختار تجميع سجلات الإدارات حسب كود الإدارة . لتوضيح هذه الملحوظة انظر المثال التالي:

```
SQL> SELECT  deptno, job, sum(sal)
  2 FROM    emp
  3 GROUP BY deptno;
```

اكتب هذه الأوامر ونفذه تحصل على رسالة خطأ كما في الشكل (16-12)

```
SQL> SELECT   deptno, job, sum(sal)
  2   FROM     emp
  3   GROUP BY deptno;
SELECT   deptno, job, sum(sal)
                 *
ERROR at line 1:
ORA-00979: not a GROUP BY expression
```

الشكل(16-12)

تم تنفيذ هذا المثال من خلال الـSQL Plus الموجودة في Oracle

ولتصحيح ذلك اكتب الصيغة التالية.

```
SQL> SELECT   deptno, job, sum(sal)
  2 FROM     emp
  3 GROUP BY deptno;
```

وعند تنفيذها نحصل على الشكل (17-12).

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900

الشكل(17-12)

والخطأ في هذا المثال هو محاولة عرض الوظيفة job في حين يتم التجميع باستعمال الحقل EMPNO.

استعمال ORDER BY مع GROUP BY

لنفرض أننا نريد عرض متوسطات مرتبات الإدارات كما سبق ولكن نريد عرض هذه المتوسطات مرتبة تنازلياً يتم ذلك باستعمال الأمر ORDER BY مع الأمر GROUP BY ويظهر ذلك من المثال التالي:

```
SQL> SELECT   job, SUM(sal) PAYROLL
  2 FROM     emp
  3 WHERE    job NOT LIKE 'SALES%'
  4 GROUP BY job
  6 ORDER BY SUM(sal);
```

في هذا المثال تم طلب عرض الوظيفة ومتوسط المرتب من جدول الموظفين وتجميع ذلك بشرط الوظيفة ثم طلب ترتيب ذلك حسب متوسط المرتب تنازلياً وذلك باستعمال order by اكتب الأمر ونفذه تحصل على الشكل (18-12).

JOB	PAYROLL
ANALYST	6000
MANAGER	8275

الشكل (18-12)

ـ استعمال الأمر Having

يستعمل هذا الأمر كبديل للأمر where الذي يستعمل في عدم وجود دوال مجموعات (Group function) ويؤدي نفس الغرض وباستعماله تستطيع وضع شروط لعرض البيانات مع Group BY مثل عرض بيانات الإدارات ذات المرتب أكبر من قيمة معينة ولتوضيح ذلك تابع المثال التالي:

```
SQL> SELECT   job, SUM(sal) PAYROLL
  2 FROM     emp
  3 WHERE    job NOT LIKE 'SALES%'
  4 GROUP BY job
  5 HAVING   SUM(sal)>5000
  6 ORDER BY  SUM(sal);
```

في هذا المثال تم إضافة الأمر HAVING بمعني WHERE.

إن متوسط المرتب (Sal) AVG اكبر من 5000 بحيث يتم عرض الإدارات التي متوسط مرتبتها أكبر من 5000.

اكتب الأمر ونفذه تحصل علي الشكل (19-12).

```
JOB       PAYROLL
--------- ---------
ANALYST      6000
MANAGER      8275
```

الشكل (19-12)

استعمال استعلامات فرعية SUB QUERIES

يمكن استعمال استعلام داخل استعلام أخر حسب الحاجة فيمكن أن يحتوي الشرط علي استعلام ، لهذا عدة أشكال منها:

الاستعلامات المتداخلة Nested subqueries

نفرض أنك تريد عرض بيانات موظف ولكن لا تعلم عنه معلومات غير أنه في نفس إدارة الموظف الذي كود الموظف له 7900.

تحقق ذلك بالشكل:

```
Select EMPNO, DeptNO, SAL FROM EMp
  Where DeptNO= (select deptNO from EMP WHERE EMPNO =7900);
```

في هذا المثال يتم عرض بيانات الموظف (كود، اسم، مرتب الموظف) الذي كود الإدارة له يساوي نتيجة الاستعلام الصغير .

الاستعلام الصغير يعيد كود الإدارة للموظف المعلوم كوده وهو 7900 وبالتالي الكود العائد من الاستعلام الصغير يوضع كطرف للشرط في الاستعلام الكبير .

اكتب هذا الأمر ونفذه تحصل على نتيجة التنفيذ كما بالشكل (20-12)

EMPNO	DEPTNO	SAL
7499	30	1600
7521	30	1250
7654	30	1250
7698	30	2850
7844	30	1500
7900	30	950

الشكل(20-12)

استعلامات متداخلة :

يمكن وضع استعلام داخل استعلام ، نفرض أنك لا تعلم كود الموظف في المثال السابق كل ما تعلمه أن الموظف الذي تريد عرض بياناته يوجد في نفس إدارة الموظف الذي له فاتورة INVOICE-NUMBER هي 5644.

اكتب الأمر التالي :

```
Select E.EMPNO, E.Sal from EMP
Where E.DEPTNO = (select DeptNo from EMP
Where EMPNO = (Select EMPNO from invoice
Where invoice_number = 5644));
```

في هذا المثال يوجد ثلاث استعلامات

الأول: يستعلم عن EMPNO، Sal من جدول الموظفين EMP ولكن الشرط أن يساوي كود الإدارة Dept NO نتيجة الاستعلام .

الثاني: يستعلم عن كود الإدارة من جدول الموظفين للموظف الذي كوده هو نتيجة الاستعلام الثالث .

الثالث: يستعلم عن كود الموظف من جدول الفواتير الذي رقم الفاتورة فيه هو 5644

وبالتالي الاستعلام الثالث يأخذ رقم الفاتورة ويعطي كود الموظف للاستعلام الثاني الذي يأخذ هذا الكود ويعطي كود الإدارة والاستعلام الأول يأخذ الكود هذا الكود ليعطي بيانات الموظف المطلوب الاستعلام عنه.

استعمال IN مع الاستعلامات الفرعية.

يمكن استعمال IN بدلاً من = إذا كان الحقل المطلوب مقارنته بمجموعة قيم وليس قيمة واحدة 0 ويظهر ذلك كما بالشكل :

```
>Select * from EMP Where EMPNO IN
(Select EMPNO from EMP where DeptNO = 20)
```

أكتب الأمر ونفه تحصل على النتيجة كما بالشكل

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7902	FORD	ANALYST	7566	03-DEC-81	3000		20

في هذا المثال يتم عرض بيانات الموظفين الذين تقع أكوادهم في القيمة العائدة من الاستعلام الصغير الذي يعيد أكواد الموظفين الذين كود الإدارة لهم 20.

استعمال ORDER BY مع الاستعلامات الفرعية

تستعمل ORDER BY لترتيب البيانات وتكتب في نهاية الاستعلام الكبير ويظهر ذلك كما بالشكل :

```
>select empno,ename from emp order by ename;
```

لاحظ أن الأمر ORDER BY عائد على الاستعلام الكبير الخارجي نتيجة ترتيب البيانات حسب كود الإدارة .

نفذ الامر لحصل على الشكل.

EMPNO	ENAME
7876	ADAMS
7499	ALLEN
7698	BLAKE
7782	CLARK
7902	FORD
7900	JAMES
7566	JONES
7839	KING
7654	MARTIN
7934	MILLER
7788	SCOTT

EMPNO	ENAME
7369	SMITH
7844	TURNER
7521	WARD

الأمر Insert

يستخدم هذا الأمر لإضافة بيانات إلى سجلات الجدول ويأخذ الصورة العامة التالية :

```
Insert Into Table-name { ( Columns - List ) }
        Values ( Value - List )
```

وفى هذه الصورة يتم إضافة القيم Value - List فى الحقول Columns - List فى الجدول Table-name ويتم تحقيق ذلك بأكثر من من صورة نوضحها فيما يلى :

الصورة الأولى :

```
Insert Into Employee Values ( 123, Ahmed , 500 eng )
```

فى هذه الصورة يتم إضافة هذه القيم إلى حقول الجدول Employee بترتيب حقول الجدول وفى هذه الصورة لم يتم تحديد الحقول وبالتالي يتم إضافة قيم للحقول كلها وبترتيب الحقول .

الصورة الثانية :

```
Insert Into Employee ( Emid , EmName , Dept )
        Values (30 , sAMY , tec )
```

وفى هذه الصورة تم إضافة بيانات إلى ثلاثة حقول وتم تحديد الحقول ثم القيم المقابلة لها.

الصورة الثالثة :

```
Insert Into Fixed - hist Select { Fixed - Data } . *
    From { Fixed - Data } *
```

و فيها يتم اختيار كل حقول الجدول Fixed - Data وإضافتها إلى الجدول Fixed - hist وبالتالى يتم إضافة جميع سجلات الجدول Fixed - Data إلى نهاية الجدول Fixed – hist.

الأمر Delete

يقوم هذا الأمر بحذف البيانات من الجدول أما كلها أو حسب الشرط الصورة العامة لها كما يلى

```
Delete From Table - name
```

وفيها يتم مسح جميع بيانات الجدول Table - name ولتوضيح ذلك تابع السطر التالى :

```
Delete From Employee
```

يؤدى هذا السطر إلى حذف جميع بيانات الجدول Employee

ويمكن مسح السجلات التى توافق شرط معين كما فى السطر التالى :

```
Delete From Employee Where Emid = 20
```

وفيها يتم مسح السجل الذى يأخذ الحقل Emid فيه القيمة 20 .

الأمر UpDate

يسمح هذا الأمر بتعديل قيمة حقل معين ويتضح ذلك من السطر التالى

```
UpDate Employee Set Salary = Salary + 200
```

وفيه يتم إضافة القيمة 200 إلى القيمة الموجودة فى الحقل Salary

ويمكن تعديل الحقول التى توافق شرط معين كما فى السطر التالى :

```
Up Date Employee Set Salary = 900 Where Salary < 900
```

وفى هذا السطر يتم وضع القيمة 900 فى الحقل Salary لجميع السجلات التى بها الحقل Salary أقل من 900 .

الفصل الثالث عشر

التعامل مع قواعد البيانات
Database Processing

Programming Concepts

في هذا الفصل

13

في هذا الفصل نبدأ الاستعمال الفعلي لـ VB.Net للتعامل مع قواعد البيانات وذلك من خلال النقاط التالية:

- الطرق المتاحة في VB.Net للتعامل مع قواعد البيانات
- التعامل مع قواعد البيانات من خلال Server Explore
- استخدام المعالجات DB Wizards
- استعمال أوامر الاتصال المباشر Direct Connection
- أدوات عرض البيانات DB Controls
- استعمال أوامر الاتصال غير المباشر Off Line DataBase
- علاقة قواعد البيانات بلغة XML

يتناول هذا الفصل الطرق المختلفة للتعامل مع قواعد البيانات من خلال لغات البرمجة وهى كما أشرنا من الموضوعات المهمة جدا فى عالم البرمجة فى الوقت الحاضر.

فى لغة VB.Net و C#

تعتبر طرق وقواعد واحدة مع وجدو الاختلاف الوحيد وهو طريقة كتابة الاوامر كما سبق وسوف نوضح ذلك أيضا.

يمكن التعامل مع قواعد البيانات من خلال Server Explore وفيها يتم التعامل مع قواعد البيانات من داخل بيئة التطوير كمطور وليس كمستخدم حيث تستطيع عن طريق هذه النافذة Server Explore فتح قواعد البيانات والتعامل مع عناصرها المختلفة.

استخدام المعالجات DB Wizards

يوفر VB.Net معالجات لانشاء النماذج التى تعتمد على قاعدة بيانات مع توفير العمليات الاساسية مثل عمليات الاضافة Insert والحذف Delete والتعديل Update

استعمال أوامر الاتصال المباشر Direct Connection

يوفر VB.Net مجموعة من الفصائل Classes التى توفر امكانية استعمال أوامر الاتصال المباشر Direct Connection بقاعدة البيانات حيث تستطيع فتح قاعدة البيانات واصدار الاوامر المختلفة لها مثل الفصيلة Connection class والفصيلة Command class وفصائل أخرى سوف نتناولها بالتفصيل.

أدوات عرض البيانات DB Controls

مجموعة من الأدوات التى يمكن ربطها بقاعدة البيانات لعرض البيانات فيها مثل الأداة الشهيرة Data Grid استعمال أوامر الاتصال غير المباشر Off Line DataBase.

يوفر VB.Net امكانية التعامل مع قاعدة البيانات بطريقة أوامر الاتصال غير المباشر Off Line DataBase وهذا يعنى عمل نسخة من البيانات فى ذاكرة الجهاز ثم القيام بالتعامل معها بالإضافة والحذف والتعديل فى انتظار أمر التحديث فيتم تحديث قاعدة البيانات الأصلية وهذا ما يسمى Off Line بحيث أننا لا نتعامل مع قاعدة البيانات مباشرة بل تعامل مع النسخة ، ويتم ذلك باستعمال فصيلة مشهورة فى VS.Net تسمى DataSet.

علاقة قواعد البيانات بلغة XML

لغة XML لغة حديثة تستخدم لتمثيل البيانات بما لها من ميزات عن لغة HTML ويوفر VS.Net تكامل كبير معها.

استعمال أدوات وفصائل وأوامر ADO.NET

وهي الطريقة الأخيرة والحديثة حتى لأن والموجودة في VB.NET بما توفره من تعديلات للفصيلة ADO وخصائص جديدة.

أدوات عرض البيانات DB Controls

يوجد مجموعة من الأدوات التي يمكن ربطها بقاعدة البيانات لعرض البيانات فيها مثل الأداة الشهيرة DataViewGrid وسوف نتناول ذلك فيما بعد .

استعمال أوامر الاتصال المباشر Direct Connection

يوفر VB.Net مجموعة من الفصائل Classes التي توفر امكانية استعمال أوامر الاتصال المباشر Direct Connection بقاعدة البيانات حيث تستطيع فتح قاعدة البيانات واصدار الاوامر المختلفة لها مثل الفصيلة Connection class والفصيلة Command class وفصائل أخرى سوف نتناولها بالتفصيل.

تعريفات

1ـ الفصيلة OleDbConnection

تستخدم هذه الفصيلة في تحديد مواصفات الاتصال Connection بقاعدة البيانات مثل اسم الجهاز وقاعدة البيانات وكلمة السر وغيره عن طريق الخاصية ConnectionString ثم فتح قاعدة البيانات باستعمال الدالة ()Open.

ويوجد من هذه الفصيلة شكلان:

- الأول هي OleDbConnection: وتستخدم عند الرغبة في لتعامل مع أي نوع من أنواع قواعد البيانات (Ms Access,Oracle,SQL Server)

وهي معرفة في NameSpace بالاسم OleDb داخل NameSpace بالاسم Data داخل

NameSpace بالاسم System لذلك للتعامل مع هذه الفصيلة لابد من كتابة السطر التالى فى أول البرنامج قبل تعريف الفصيلة.

فى VB.NET

```
Imports System.Data.OleDb
```

فى C#

```
using System.Data.OleDb
```

الفصيلة SqlConnection

وهى تستخدم عند الرغبة فى التعامل مع قاعدة بيانات من النوع SQL Server وقد أعدتها ميكروسوفت لتسهل التعامل مع قواعد البيانات الخاصة بها من النوع SQL Server حيث تعتبر هى النوع الافتراضى لديها.

ولاستعمال هذه الفصيلة لابد من كتابة السطر التالى فى أول البرنامج

```
Imports System.Data.SqlClient
```

فى لغة C#

```
using System.Data.SqlClient
```

2. الفصيلة OleDbCommand

وتستعمل فى تحديد جملة SQL التى تحدد البيانات التى ستتعامل معها مع تحديد متغير (هدف Object) فصيلة الاتصال بقاعدة البياناتConnection التى نتعامل معها وتأخذ صورتين مثل الفصيلة OleDbConnection.

الأول:**OleDbCommand**

وهى للتعامل مع أى نوع من أنواع قواعد البيانات.

والثانية **SqlCommand**

للتعامل مع قواعد البيانات من النوع SQl Server.

3. الفصيلة OleDbDataReader

تستخدم هذه الفصيلة لقراءة البيانات التى تم تحديدها باستخدام الفصيلة OleDbCommand

وذلك باستعمال الدالة ExecuteReader() العضوة في الفصيلة OleDbCommand وضع النتيجة في هدف الفصيلة OleDbDataReader ثم التعامل مع هذه البيانات وتأخذ صورتين مثل الفصيلة OleDbConnection.

- **الأولى OleDbDataReader**

كما ذكرنا للتعامل مع أي نوع من قواعد البيانات.

- **والثانية SqlDataReader**

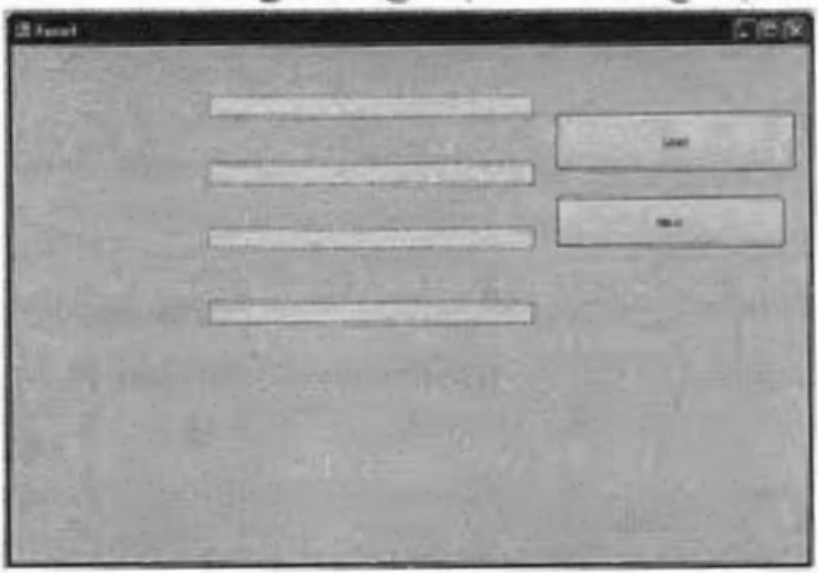
للتعامل مع قواعد البيانات من النوع SQl Server.

ويتم توضيح هذه النقاط بمثال كما في الخطوات التالية:

مثال

1- أبدأ تطبيق جديد من لنوع Windows Application كما سبق.

2- قم بتوقيع الأدوات وتصميم نموذج للتعامل مع البيانات كما في الشكل (1-13).



الشكل(1- 13)

3-اضغط بالماوس مرتين فوق النموذج Form ولكتب في أول الملف سطور فتح NameSpace للتعامل مع فصائل قواعد البيانات كما في الشكل (2-13).

فى VB.NET

```
1  Imports System.Data
2  Imports System.Data.OleDb
3
4
5  Public Class Form3
```

الشكل(2- 13)

فى لغة C#

```
1  using System;
2  using System.Drawing;
3  using System.Collections;
4  using System.ComponentModel;
5  using System.Windows.Forms;
6  using System.Data;
7  using System.Data.OleDb ;
8
9  namespace WindowsApplication14
```

- فى السطر رقم 2 تم استعمال الامر Imports لفتح الـ OleDb NameSpace كما أشرنا من قبل.

4- قم بكتابة سطور تعريف أهداف Objects من الفصيلة OleDbConnection للتعامل مع قاعدة البيانات والفصيلة OleDbDataReader لقراءة البيانات كما فى الشكل.

فى VB.NET

```
5  Public Class Form3
6      Inherits System.Windows.Forms.Form
7      Dim objConnection As New OleDbConnection()
8      Dim myReader As OleDbDataReader
9
```

فى لغة C#

```
14    public class Form1 : System.Windows.Forms.Form
15    {
16
17        OleDbConnection objConnection=new OleDbConnection();
18        OleDbDataReader myReader ;
19
```

5- اضغط زر الأمر المكتوب عليه Load Data والذى نستخدم فى فتح قاعدة البيانات وتحميل لبيانات أول مرة واكتب السطر التالى:

فى VB.NET

```
objConnection.ConnectionString = ("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA
SOURCE=c:\Northwind.mdb")
```

فى لغة C#

```
objConnection.ConnectionString=
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=c:\\Northwind.mdb";
```

- فى هذا السطر يتم تحديد مواصفات قاعدة البيانات التى نتعامل معها بتحديد قيمة المتغير ConnectionString لهدف الفصيلة OleDbConnection المعرف بالاسم ObjConnection وفيها يتم تحديد.

5-1 نوع قاعدة البيانات بالعبارة ;ROVIDER=Microsoft.Jet.OLEDB.4.0 حيث تعبر عن MsAccess ويتم تغييرها عند التعامل مع قاعدة بيانات أخرى.

5-2 تحديد قاعدة البانات التى نرغب فى التعامل معها وذلك بكتابة العبارة DATA SOURCE=c:\Northwind.mdb وفيها يتم استعمال الجملة Data source ثم تحديد اسم ومسار قاعدة البيانات وفى حالتنا هى c:\Northwind.mdb وهى قاعدة البانات التى تضيفها ميكروسوت كمثال داخل برنامج قواعد البيانات Ms Access.

ملحوظة

للتعامل مع قاعدة بيانات من النوع SQL Server يتم استبدال مواصفات فتح قاعدة البيانات فقط وجميع العمليات كما هي وتأخذ الصفات فتح قاعدة البيانات الصورة التالية:

```
Dim cn As New SqlClient.SqlConnection()
cn.ConnectionString = "data source=server;initial catalog=EMPLOYEE;user ID=sa"
```

- في هذه السطور تم تعريف المتغير cn من نوع SqlConnection.
- ثم تحديد مواصفات الاتصال بتحديد قيمة الخاصية ConectionString وفيها.
- Data Source تساوى القيمة Server وهو اسم الجهاز الخادم Server الموجود عليه قاعدة البيانات.
- Initial catalog تحدد اسم قاعدة البيانات وفى حالتنا هى EMPLOYEE.
- user ID اسم المستخدم المعرف فى SQl Server والمستخدم للدخول على قاعدة البيانات EMPLOYEE وهو فى حالتنا sa وهو المستخدم المعرف فى SQl Server وبالتالى تصبح سطور البرنامج هكذا:

```
Sub Main()

Dim conn1 As New SqlConnection()
conn1.ConnectionString = "Data Source=Azab;Initial Catalog=CSO;Persist
Security Info=True;User ID=sa;Password=azabazab"
Dim cmd As New SqlCommand
cmd.CommandType = CommandType.Text
cmd.CommandText = "SELECT CustomerID, CompanyName FROM Customers"

cmd.Connection = conn1
' Create a SqlParameter for each parameter in the stored procedure.

Dim reader As SqlDataReader
Dim previousConnectionState As ConnectionState = conn1.State
Try
If conn1.State = ConnectionState.Closed Then
conn1.Open()
End If
reader = cmd.ExecuteReader()
Using reader
While reader.Read
' Process SprocResults datareader here.
Console.Write(reader.GetValue(0))
```

```
Console.WriteLine("----" + reader.GetValue(1))

End While
End Using
Finally
If previousConnectionState = ConnectionState.Closed Then
conn1.Close()
End If
End Try

Console.ReadKey()
End Sub
```

6-اكتب سطور فتح قاعدة البيانات.

```
conn.ConnectionString = ("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA
SOURCE=c:\Northwind.mdb")
conn.Open()
MsgBox("Opened")

Dim strsql As String
strsql = "select * from Employees"

cmd.CommandType = CommandType.Text
cmd.CommandText = "SELECT * FROM Customers"

cmd.Connection = conn
' Create a SqlParameter for each parameter in the stored procedure.

Dim previousConnectionState As ConnectionState = conn.State
Try
If conn.State = ConnectionState.Closed Then
conn.Open()
End If
reader = cmd.ExecuteReader()
reader.Read()

TextBox1.Text = reader.GetValue(0)
TextBox2.Text = reader.GetValue(1)
TextBox3.Text = reader.GetValue(2)
TextBox4.Text = reader.GetValue(3)
Finally
If previousConnectionState = ConnectionState.Closed Then
conn.Close()
End If
End Try
```

فى لغة C#

```
158        private void button1_Click(object sender, System.EventArgs e)
159        {
160            objConnection.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;I
161            objConnection.Open ();
162            string strSQL;
163            strSQL="SELECT EmployeeID,LastName,FirstName,Title FROM Employees Or
164            OleDbCommand myCommand = new OleDbCommand(strSQL,objConnection);
165            myReader = myCommand.ExecuteReader();
166            myReader.Read();
167
168            textBox1.Text=myReader.GetInt32   (0)+"";
169            textBox2.Text =myReader.GetString (1);
170            textBox3.Text =myReader.GetString (2);
171            textBox4.Text =myReader.GetString (3);
172
```

فى هذه السطور

- فى السطر 112 تم تعريف مواصفات قاعدة البيانات كما سبق وشرحنا.
- فى السطر رقم 113 تم فتح قاعدة البيانات باستدعاء الدالة ()Open مع متغير وصلة قاعدة البيانات objConnection.
- فى السطر 115 تم الاعلان عن متغير strSQL من نوع string.
- فى السطر رقم 116 تم تحديد جملة الـ SQL المطلوب تنفيذها وذلك بوضعها فى المتغير strSQL.
- فى السطر 118 تم تعريف المتغير objCommand من نوع الفصيلة OleDbCommand التى تتعامل مع جملة الـ SQL وتنفذها مع ارسال جملة الـ SQL التى حددناها كمعامل وبالتالى أصبح المتغير objCommand يشير الى جملة الـ SQL المحددة.
- فى السطر رقم 119 تم استدعاء الدالة ExecuteReader مع المتغير objCommand وبالتالى تنفيذ جملة الـ SQL المحددة مسبقا ووضع النتيجة فى المتغير myReader

المعرف فى أول البرنامج من النوع OleDbReader والذى يقوم بقراءة بيانات من قاعدة بيانات.

- فى السطر رقم 120 يتم استدعاء الدالة Read() مع المتغير myReader والتى تقوم بقراءة سجل Record.
- فى السطور 122و123و124و125 يتم وضع قيم الحقول Fields المحددة (EmployeeID,FirstName,LastName,Title) فى مربعات النصوص TextBox1,TextBox2,TextBox3,TextBox4 وبالتالى تظهر بيانات أول سجل فى الشاشة.

7-اكتب سطر زر الامر الثانى الذى يعرض السجل التالى:

```
reader.Read()

TextBox1.Text = reader.GetValue(0)
TextBox2.Text = reader.GetValue(1)
TextBox3.Text = reader.GetValue(2)
TextBox4.Text = reader.GetValue(3)
```

فى لغة C#

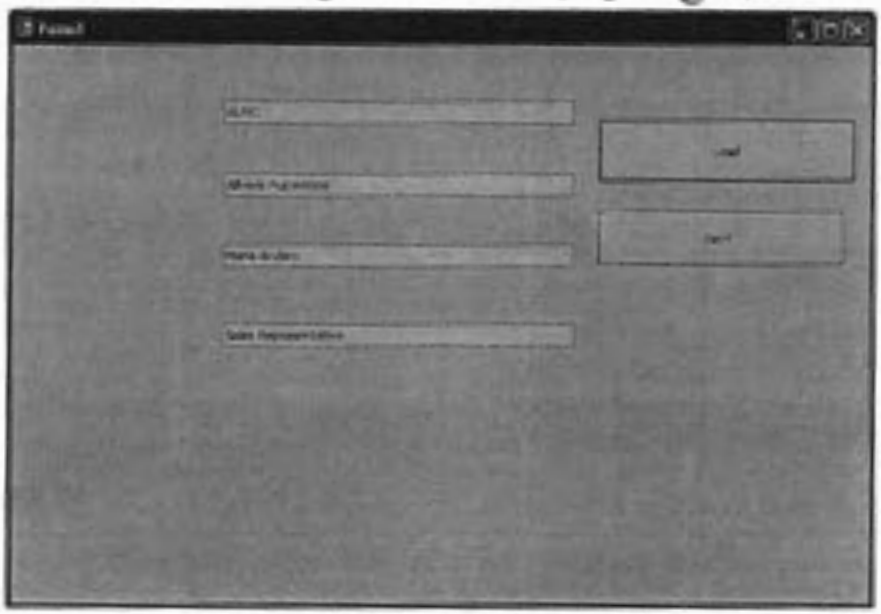
```
175        private void button2_Click(object sender, System.EventArgs e)
176        {
177            myReader.Read();
178
179            textBox1.Text=myReader.GetInt32   (0)+"";
180            textBox2.Text =myReader.GetString (1);
181            textBox3.Text =myReader.GetString (2);
182            textBox4.Text =myReader.GetString (3);
183
184        }
```

فى هذه السطور:

- فى السطر رقم 130 يتم استدعاء الدالة Read() مع المتغير myReader والتى تقوم بقراءة سجل Record.

- فى السطور 131و132و133و134 يتم وضع قيم الحقول Fields المحددة(EmployeeID,FirstName,LastName,Title) فى مربعات النصوص TextBox1,TextBox2,TextBox3,TextBox4 وبالتالى تظهر بيانات السجل التالى فى الشاشة كلما تم الضغط على هذا الزر.

8- نفذ البرنامج لتحصل على النتيجة التالية الشكل (3-13):



الشكل(3- 13)

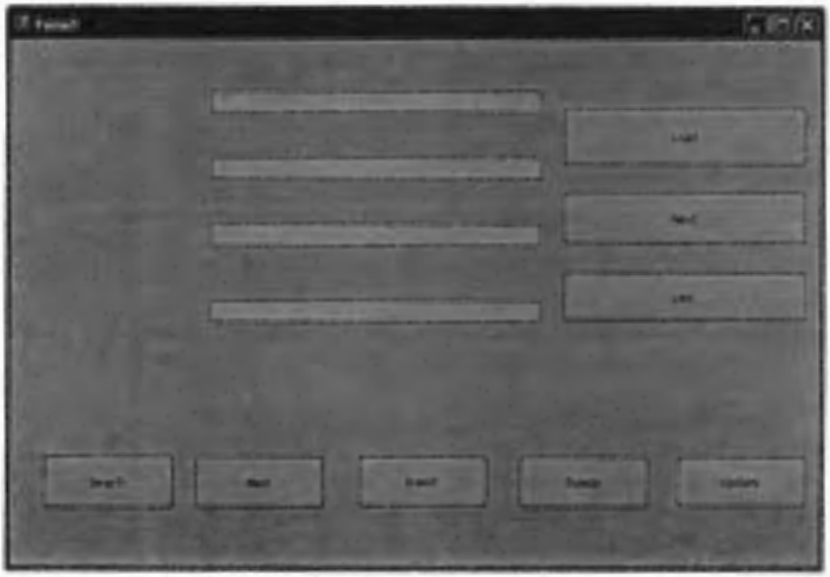
تحقيق عمليات قواعد البيانات

من الضرورى تحقيق العمليات الأساسية لقواعد البيانات وهى:

- الاضافة Insert والحذف Delete والتعديل Update والبحث ويتم ذلك بسهولة بتغيير جملة الـ SQL التى تمثل العملية المطلوبة.

وقد تم شرح جمل SQL المسؤلة عن هذه العمليات ولتوضيح كيفية تحقيق ذلك تابع معنا الخطوات التالية:

1-عد الى البرنامج السابق وقم باعادة تصميم نموذج البيانات الذى صممناه ليصبح كما فى الشكل (4-13).



الشكل(4- 13)

فى هذا الشكل نلاحظ اننا أضفنا أزرار أوامر Buttons للعمليات المطلوبة ويتبقى كتابة الأوامر التى تحققها كما يلى:

2- اكتب أوامر العمليات التالية.

زر البحث Search

يسمح هذا الزر للمستخدم بكتابة كود الموظف EmployeeID المطلوب البحث عنه فى مربع النص الأول TextBox1 ثم الضغط على الزر Search فتظهر بيانات هذا الموظف وتظهر سطور أوامر هذه العملية كالتالى:

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click

conn.ConnectionString = ("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA
SOURCE=c:\Northwind.mdb")
cmd.CommandType = CommandType.Text
Dim custIdTxtx As String
custIdTxtx = TextBox1.Text + ""

cmd.CommandText = "SELECT * FROM Customers where CustomerID='" +
custIdTxtx
cmd.Connection = conn
```

```
' Create a SqlParameter for each parameter in the stored procedure.
Dim previousConnectionState As ConnectionState = conn.State
Try
If Not conn.State = ConnectionState.Closed Then

conn.Close()

End If

If conn.State = ConnectionState.Closed Then
conn.Open()
End If
reader = cmd.ExecuteReader()
reader.Read()

TextBox1.Text = reader.GetValue(0)
TextBox2.Text = reader.GetValue(1)
TextBox3.Text = reader.GetValue(2)
TextBox4.Text = reader.GetValue(3)


Finally
If previousConnectionState = ConnectionState.Closed Then
conn.Close()
End If
End Try
End Sub
```

فى لغة C#

```
string strSQL;
strSQL="SELECT EmployeeID,LastName,FirstName,Title FROM
Employees WHERE EmployeeID="+textBox1.Text;
OleDbCommand myCommand=new
OleDbCommand(strSQL,objConnection);
myReader = myCommand.ExecuteReader();
myReader.Read();
textBox1.Text=myReader.GetInt32   (0)+"";
textBox2.Text =myReader.GetString (1);
textBox3.Text =myReader.GetString (2);
textBox4.Text =myReader.GetString (3);
myReader.Close ();
```

فى هذه السطور:

هذه هى نفس سطور أوامر زر أمر عرض السجل التالى ولكن الجديد هو:

1- وضع محتوى مربع النص TextBox1 وهو الرقم المطلوب البحث به فى المتغير srchtxt وذلك فى السطر رقم 190.

2-تعديل جملة الـ SQL باضافة جزء الشرط Where الذى يحدد السجلات المطلوب عرضها كما فى السطر 191.

مع جمع محتوى المتغير srchtxtعلى جملة السSQL وبالتالى يتم الحصول على السجلات التى فيها قيمة الحقل EmployeeID تساوى القيمة المطلوب البحث عنها والموجودة فى المتغير srchtxt والمأخوزة من مربع النص TextBox1 وباقى السطور كما سبق.

زر سجل جديد New

هذا الزر لا يحقق أحد عمليات قواعد البيانات فقط يؤدى الى مسح محتويات مربعات النصوص TextBox لتسمح للمستخدم بكتابة بيانات جديدة أو لكتابة كود موظف للبحث عنه ولذلك فأن سطوره بسيطة وهى كالتالى:

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = ""
    TextBox4.Text = ""
End Sub
```

```
260         private void button4_Click(object sender, System.EventArgs e)
261         {
262             textBox1.Text="";
263             textBox2.Text ="";
264             textBox3.Text ="";
265             textBox4.Text ="";
266         }
```

زر الاضافة Insert

يسمح هذا الزر للمستخدم بكتابة بيانات جديدة وحفظها بقاعدة البيانات كما بالسطور التالية:

```
conn.ConnectionString = ("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA
SOURCE=c:\Northwind.mdb")
cmd.CommandType = CommandType.Text

cmd.CommandText = "Insert Into Customers
(CustomerID,CompanyName,ContactName,ContactTitle) values
('code1','CSO','Omr','Mis')"
cmd.Connection = conn
' Create a SqlParameter for each parameter in the stored procedure.
Dim previousConnectionState As ConnectionState = conn.State
If Not previousConnectionState = ConnectionState.Closed Then
conn.Close()
End If
conn.Open()
cmd.ExecuteNonQuery()

conn.Close()
```

```
269        private void button5_Click(object sender, System.EventArgs e)
270        {
271            string str0GA;
272  str0GA="INSERT INTO Employees(EmployeeID,LastName,FirstName,Title) VALUES(99,'Omr','sss','dddd')
273            OleDbCommand myCommand = new OleDbCommand(str0GA,objConnection);
274            myCommand.ExecuteNonQuery ();
275
276
277        }
```

في هذه السطور:

- تم تغيير جملة الـ SQL كما في السطر 227 الى جملة Insert Into التي شرحناها من قبل والمسؤلة عن إضافة سجل جديد مع تثبيت القيم المطلوب إضافتها وعدم أخذها من مربعات النصوص من الشاشة وذلك للتجربة.

- فى السطر 229 تم استدعاء الدالة ExecuateNonQuery() التى تقوم بتنفيذ جمل SQL غير الاستعلامية أى التى لا تحتوى على الأمر Select مثل أوامر الاضافة Insert وأوامر الحذف Delete والتحديث Update وغيره.
- وبالتالى يقوم هذا الزر باضافة سجل جديد وحفظ القيم المحددة فى جملة Insert به ولكن المطلوب حفظ القيم التى يكتبها المستخدم فى مربعات النصوص TextBox لذلك قم بتعديل سطور أوامر الزر Insert كما يلى:

زر الاضافة بعد التعديل Insert

فى الفقرة السابقة قمنا بإضافة قيم محددة وذلك لتجربة عملية الإضافة فقط ولكن الطبيعى هو استقبال القيم المطلوب إضافتها من المستخدم ثم إضافتها لذلك قم بتعديل سطور عملية الاضافة Insert لتصبح كما فى السطور التالية:

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
        conn.ConnectionString = ("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA
SOURCE=c:\Northwind.mdb")
        cmd.CommandType = CommandType.Text
        Dim txt1, txt2, txt3, txt4, vals As String

        txt1 = "('" + TextBox1.Text + "'"
        txt2 = ",'" + TextBox2.Text + "'"
        txt3 = ",'" + TextBox3.Text + "'"
        txt4 = ",'" + TextBox4.Text + "')"

        vals = txt1 + txt2 + txt3 + txt4

        Dim sql1 As String
        sql1 = "Insert Into Customers
(CustomerID,CompanyName,ContactName,ContactTitle) values "
        Dim lastSQL As String
        lastSQL = sql1 + vals
        MsgBox(lastSQL)
        cmd.CommandText = lastSQL
        cmd.Connection = conn
        ' Create a SqlParameter for each parameter in the stored procedure.
        Dim previousConnectionState As ConnectionState = conn.State
        If Not previousConnectionState = ConnectionState.Closed Then
            conn.Close()
```

```
    End If
    conn.Open()
    cmd.ExecuteNonQuery()
    MsgBox("Record Inserted ..")

    conn.Close()
End Sub
```

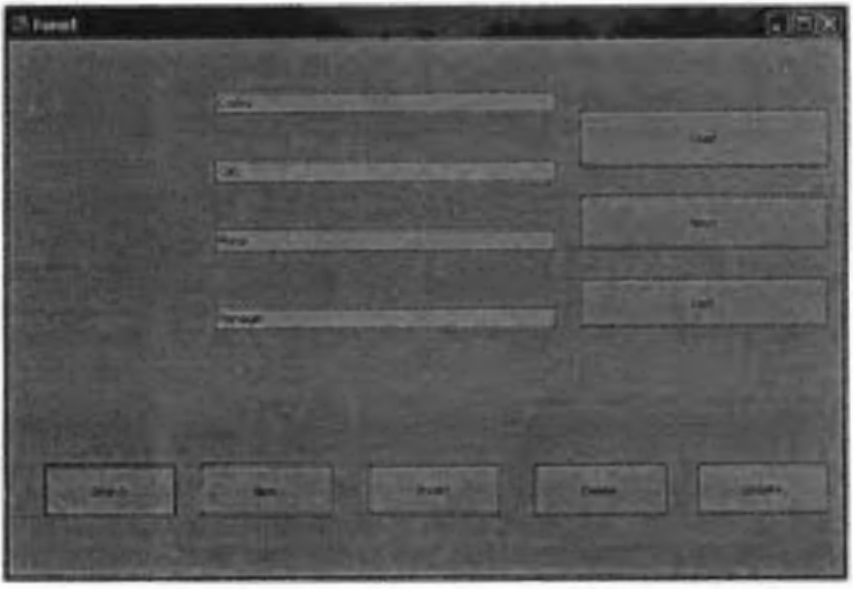
```
        private void button5_Click(object sender, System.EventArgs e)
        {
string str0QL,s2;
int EmployeeID_v;
string LastName_v;
string FirstName_v;
string Title_v;

EmployeeID_v=Int32.Parse (textBox1.Text) ;
LastName_v=textBox2.Text ;
FirstName_v=textBox3.Text ;
Title_v=textBox4.Text ;

str0QL="INSERT INTO Employees(EmployeeID,LastName,FirstName,Title)";
s2="VALUES("+EmployeeID_v+","+"'"+LastName_v+"'" +","+"'"+FirstName_v+"'"+","+"'"+Title_v+"'"+")";
str0QL+=s2;
    OleDbCommand myCommand = new OleDbCommand(str0QL,objConnection);
    myCommand.ExecuteNonQuery ();


        }
```

في هذه السطور:

- في السطر رقم 226 تم تعريف اربعة متغيرات من النوع string.
- في السطور 227و228و229و230 تم وضع قيم مربعات النصوص TextBox1,TextBox2,textBox3,TextBox4 وهي القيم التي كتبها المستخدم ليضيفها الى قاعدة البيانات في المتغيرات المعرفة.
- في السطر رقم 232 يتم كتابة جملة الـ SQL المطلوب تنفيذها وهي جملة Insert السابقة ولكن بتعديلها وجمع عليها قيم المتغيرات بدلاً من وضع قيم ثابتة.
- في السطر 234 يتم استعمال الدالة MsgBox() لعرض محتوى متغير جملة الـ SQL لمراجعتها ومعرفة نتيجة جمع المتغيرات على جملة الـ SQL.
- في السطر رقم 236 يتم استدعاء الدالة ExecuateNonQuery() لتنفيذ جملة الـ SQL

وإضافة السجل الجديد.

- نفذ البرامج عند هذه الخطوة واضغط الزر New ثم اكتب بيانات سجل جديد ثم اضغط بالماوس الزر Insert يتم اضافة السجل كما فى الشكل (5-13) .



الشكل(5- 13)

زر الحذف Delete

يؤدى الضغط على زر الأمر Delete إلى حذف بيانات الموظف الذى كتب كوده فى مربع النص الأول TextBox1 ويتم تحقيق ذلك بالأوامر التالية :

```
Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
        conn.ConnectionString = ("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA
SOURCE=c:\Northwind.mdb")
        cmd.CommandType = CommandType.Text
        Dim txt1 As String

        txt1 = "" + TextBox1.Text + ""

        Dim sql1 As String
        sql1 = "Delete from  Customers where CustomerID=" + txt1

        MsgBox(sql1)
```

```
    cmd.CommandText = sql1
    cmd.Connection = conn
    ' Create a SqlParameter for each parameter in the stored procedure.
    Dim previousConnectionState As ConnectionState = conn.State
    If Not previousConnectionState = ConnectionState.Closed Then
       conn.Close()
    End If
    conn.Open()
    cmd.ExecuteNonQuery()
    MsgBox("Record Deleted ..")

    conn.Close()
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = ""
    TextBox4.Text = ""
End Sub
```

فى لغة C#

```
private void button6_Click(object sender, System.EventArgs e)
{
    string strSQL;
    string EmployeeID_v;

    EmployeeID_v=textBox1.Text;
    strSQL="DELETE FROM  Employees WHERE EmployeeID="+EmployeeID_v;
    OleDbCommand myCommand = new OleDbCommand(strSQL,objConnection)
    myCommand.ExecuteNonQuery ();

    textBox1.Text="";
    textBox2.Text ="";
    textBox3.Text ="";
    textBox4.Text ="";

}
```

فى هذه السطور

يتم تغيير جملة الـ SQL من جملة Insert إلى جملة Delete وذلك كما فى السطر رقم 248 ثم تنفيذ هذه الجملة وبالتالى حذف السجل المكتوب رقمه فى مربع النص TextBox1.

زر تحديث البيانات Update

ويتيح لك هذا الزر إمكانية تعديل قيم بيانات موجودة وفيه يتم تغيير جملة الـ SQL إلى جملة Update كما فى السطور.

```
Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button7.Click
        conn.ConnectionString = ("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA
SOURCE=c:\Northwind.mdb")
        cmd.CommandType = CommandType.Text
        Dim txt1, txt2, txt3, txt4, vals As String

        txt1 = "" + TextBox1.Text + ""
        txt2 = "" + TextBox2.Text + ""
        txt3 = "" + TextBox3.Text + ""
        txt4 = "" + TextBox4.Text + ""

        vals = txt1 + txt2 + txt3 + txt4

        Dim sql1 As String
        sql1 = "update Customers set  CompanyName=" + txt2 +
",ContactName=" + txt3 + ",ContactTitle=" + txt4 + " where CustomerID=" +
txt1
        MsgBox(sql1)
        cmd.CommandText = sql1
        cmd.Connection = conn
        ' Create a SqlParameter for each parameter in the stored procedure.
        Dim previousConnectionState As ConnectionState = conn.State
        If Not previousConnectionState = ConnectionState.Closed Then
            conn.Close()
        End If
        conn.Open()
        cmd.ExecuteNonQuery()
        MsgBox("Record Updated ..")

        conn.Close()

    End Sub
```

فى لغة C#

```
string strSQL,s2;
int EmployeeID_v;
string LastName_v;
string FirstName_v;
string Title_v;

EmployeeID_v=Int32.Parse (textBox1.Text) ;
LastName_v=textBox2.Text ;
FirstName_v=textBox3.Text ;
Title_v=textBox4.Text ;

strSQL="UPDATE  Employees SET LastName='"+LastName_v+"',";
s2="FirstName='"+FirstName_v+"',Title='"+Title_v+"' WHERE EmployeeID="+EmployeeID_v;
strSQL+=s2;
OleDbCommand myCommand = new OleDbCommand(strSQL,objConnection);
myCommand.ExecuteNonQuery ();
```

كما ذكرنا يمكن تنفيذ هذا المثال مع قاعدة بيانات من النوع SQl Server وذلك بتغيير قيمة الخاصية ConnectionString كما أشرنا فى الملحوظة السابقة.

زر الأمر Last

اكتب السطور التالية :

```
Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button8.Click
    While (reader.Read)
        TextBox1.Text = reader.GetValue(0)
        TextBox2.Text = reader.GetValue(1)
        TextBox3.Text = reader.GetValue(2)
        TextBox4.Text = reader.GetValue(3)
    End While
End Sub
```

استعمال أوامر الاتصال غير المباشر Off Line DataBase

يوفر VB.Net امكانية التعامل مع قاعدة البيانات بطريقة أوامر الاتصال غير المباشر Off Line DataBase وهذا يعنى عمل نسخة من البيانات فى ذاكرة الجهاز ثم القيام بالتعامل معها بالاضافة والحذف والتعديل فى انتظار أمر التحديث فيتم تحديث قاعدة البيانات الأصلية وهذا ما يسمى Off Line بحيث اننا لا نتعامل مع قاعدة البيانات مباشرة بل تعامل مع النسخة ، ويتم ذلك باستعمال فصيلة مشهورة فى VS.Net تسمى DataSet وسوف نتعرض لهذه النقطة بالتفصيل فى فصل تالى.

علاقة قواعد البيانات بلغة XML

لغة XML لغة حديثة تستخدم لتمثيل البيانات بما لها من ميزات عن لغة HTML ويوفر VS.Net تكامل كبير معها فتستطيع فتح بيانات فى الصورة xml ووضعها فى قاعدة البيانات وكذلك حفظ بيانات من قاعدة البيانات فى الصورة xml وسوف نتعرض لهذه النقطة بالتفصيل فيما بعد.

فى لغة Java

ربط قواعد البيانات والجافا Java Data Base Connectivity (JDBC)

فى لغة الجافا Java يتم التعامل مع قواعد البيانات من خلال مجموعة من الفصائل Classes والـ interfaces وهذة الفصائل تسمى JDBC وهى اختصار لـ Java Data Base JDBC Connectivity.

وهذه الفصائل تستخدم لغة SQL للقيام بالاستعلامات فى برامج قواعد البيانات المختلفة.

وتشتمل مجموعة فصائل JDBC على الفصائل التى تمكننا من القيام بالمهام الشائعة عند التعامل مع قواعد البيانات مثل:

1. القيام بالربط بين الجافا وقاعدة البيانات.
2. إنشاء الجمل باستخدام SQL.

3. تنفيذ تلك الجمل داخل قاعدة البيانات (الاستعلامات).
4. رؤية نتيجة تنفيذ الاستعلامات في صورة سجلات.

وتوجد فصائل JDBC داخل الحزمة java.sql

وكما قلنا قبل ذلك فلكي نقوم بإنشاء تطبيقات قواعد البيانات مع لغات البرمجة فلابد من وجود واجهة Interface بين قاعدة البيانات ولغة البرمجة وتستخدم لغة الجافا الواجهة ODBC ولكن من خلال فصائل JDBC واصطلح على تسمية ذلك JDBC-ODBC BRIDGE.

الربط بين فصائل JDBC وواجهة قواعد البيانات ODBC

The JDBC-ODBC BRIDGE

ويقوم JDB-ODBC Bridge بالسماح لمشغلات JDBC drivers لكي تستخدم كأنها ODBC عن طريق تحويل استدعاء دوال JDBC إلى استدعاء وظائف ODBC.

ODBC

وODBC هي واجهة قد قامت ببنائها شركة مايكروسوفت لكي تسهل عملية بناء برامج تتصل ببرامج قواعد البيانات وفق معايير محددة وثابتة. وكل شركة منتجة لبرنامج من برامج قواعد البيانات تقوم بتوفير مشغل Driver ويتم دمجة في واجهة ODBC حتى يستطيع المطورون استخدام برامج قواعد البيانات وفق معايير موحدة.

وتوجد واجهة ODBC في لوحة التحكم Control Panel في ويندوز 95.98 وموجودة المجلد ضمن Administrative Tools الموجودة في لوحة التحكم ويندوز 2000 و ODBC بها مشغلات Drivers لقواعد البيانات الموجودة على جهازك فكل شركة من شركات قواعد البيانات تقوم بتوفير مشغل Driver خاص ببرنامج قاعدة بياناتها وذلك كما يتضح من الشكل (6-13).

الشكل (13-6)

استخدام JDBC-ODBC Bridge

لكى استخدم JDBC-ODBC Bridge يستلزم أن احدد ثلاثة أشياء :

1- مشغل JDBC-ODBC Bridge الموجود فى لغة جافا وهو : sun.jdbc.odbc.JdbcOdbcDriver

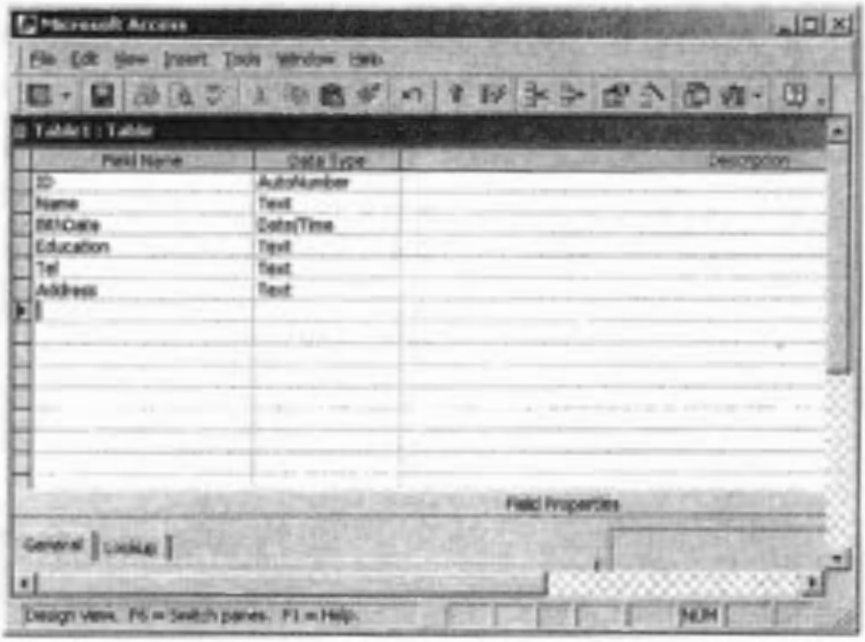
2- مشغل ODBC An ODBC Driver

3- قاعدة البيانات المرتبطة بالمشغل ODBC

وبعد أن أوضحنا بشيء من التفصيل تعال معى نرى من خلال مثال عملى كيفية تطبيق ذلك.

1- البرامج المطلوبة لتطبيق هذا المثال هو برنامج Access لإنشاء قاعدة البيانات. قم بتشغيل برنامج Access وقم بإنشاء قاعدة البيانات وأعطها اسم وليكن Javadb1 .

2- قم بإنشاء جدول Table جديد بالاسم Basicdata وقم بإنشاء أسماء الحقول Field Name وأنواعها DataTypes وذلك كما فى الشكل (7-13).

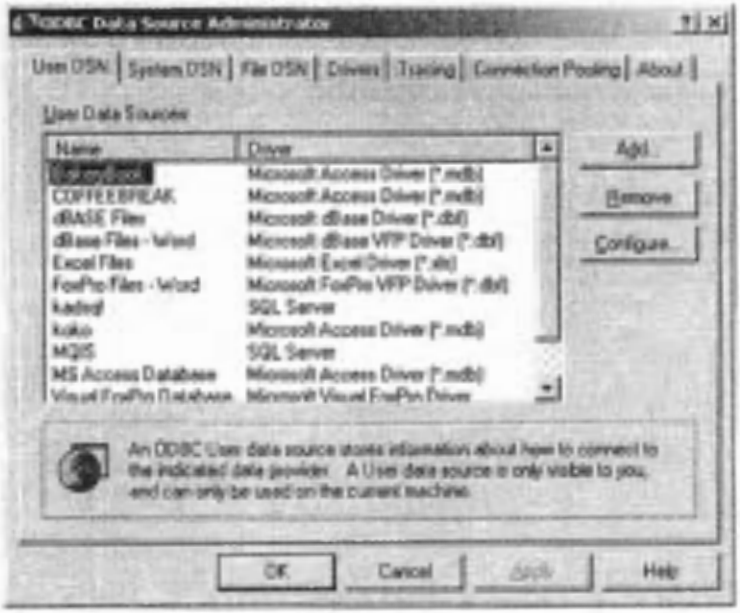


الشكل (13-7)

3- قم بفتح لوحة التحكم Control Panel واضغط مرتين على ODBC كما في الشكل (13-8).

الشكل (13-8)

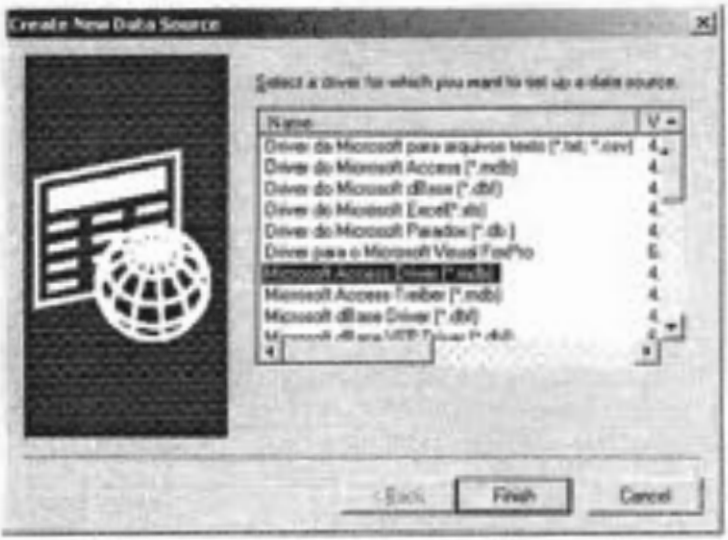
4- فيظهر الشكل (9-13).



الشكل (9-13)

5- اختار صفحة (تبويب) User DSN وهو أول تبويب على أقصى الشمال كما في الشكل.

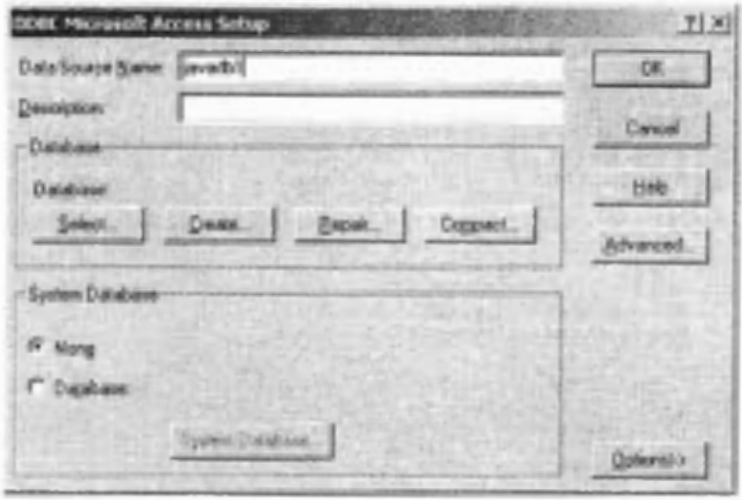
6- أنقر زر Add فيظهر الشكل (10-13).



الشكل (10-13)

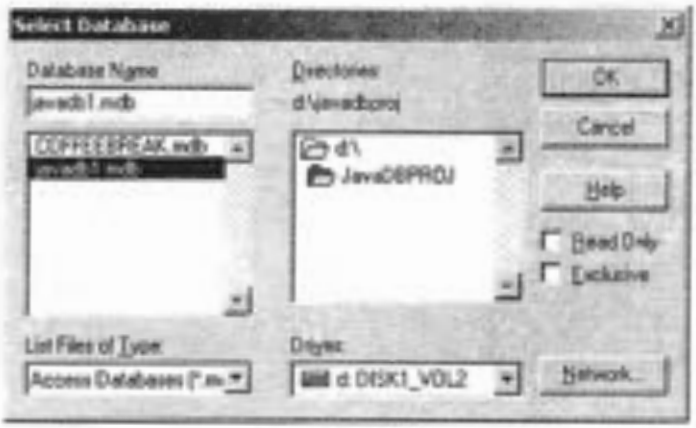
7- اختر Microsoft Access Driver (mdb) كما فى الشكل ثم اضغط Finish.

8- يظهر لك الشكل (11-13).



الشكل (11-13)

يطلب منك فى الحقل الأول أن تكتب اسم الوصلة وليكن Javadb1 وليس بالضرورة أن يكون نفس اسم قاعدة البيانات التى أنشأناها ولكننا هنا كتبنا نفس الاسم لكى للسهولة ثم اضغط على زر Select لكى نختار قاعدة البيانات التى أنشأناها فيظهر الشكل (12-13).



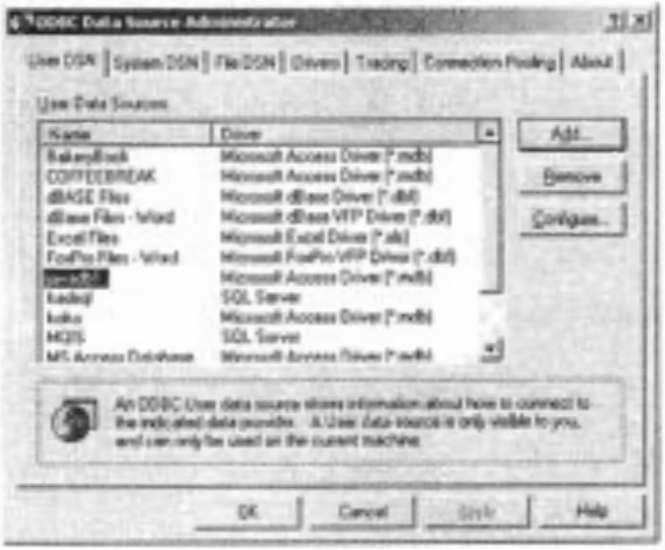
الشكل (21-13)

فقم بالذهاب إلى المسارpath الذى حفظت فيه قاعدة البيانات Javadb1 فتجد مثلا أننى قمت بحفظ قاعدة البيانات فى المسار التالى:

D:>JavaDBProJ \Javadb1.Mdb

9- بعد ذلك اضغط على الزر OK فيظهر مربع الحوار الأخير فاضغط على الزر OK ايضاً.

10- فتجد الشكل (13-13)



الشكل (13-13)

وقد ظهر به اسم Javadb1 الذى أنشأناه بعد ذلك اضغط على الزر OK.

لاحظ أن فى الخطوات من 3 إلى 7 قمنا بتنفيذ خطوتين هما:

1- تحديد ODBC Driver وفى هذا المثال اخترنا مشغل برنامج قاعدة البيانات Access.

2- تحديد قاعدة البيانات المرتبطة بالمشغل الذى اخترناه والتى هى Javadb1.

بذلك انتهى جزء إنشاء قاعدة البيانات ثم إضافة وصلة باسمها فى ODBC يقى لنا جزء كود الجافا java وهى أسهل من الجزء السابق فقط هناك ثلاثة خطوات رئيسية أقوم بها.

1- إنشاء متغير من نوع String وذلك لكى احتفظ فيه بعنوان الوصلة التى أنشأتها فى ODBC كالتالى:

```
String URL = " jdbc:odbc:Javadb1";
```

2-أقوم بتحميل المشغل الموجود فى الجافا لكى أقوم بالربط بقاعدة البيانات كالأتى:

```
Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
```

3- بعد تحميل المشغل نستطيع إنشاء وصلة Connection بقاعدة البيانات التى أنشأناها كالأتى:

```
Connection Con= DriverManager.getConnection, (URL, " ", " ")
```

حيث نجد أنها تأخذ ثلاثة معاملات.

1. عنوان وصلة قاعدة البيانات التى أنشأناها واعطيناه URL نظر لأن URL يحتوى على العنوان كما فى الخطوة 1.
2. المعامل الثانى يحتوى على اسم من مستخدم لقاعدة البيانات وحيث أننا لم نحدد ذلك تركناه فارغ " ".
3. المعامل الثالث كلمة السر لمستخدم قاعدة البيانات وأيضا تركناه فارغ " ".

بعد هذه الخطوات الثلاثة نستطيع أن نفعل ما نريد فى قاعدة البيانات فتعال معى نرى ذلك تفصيلياً كما فى نص البرنامج التالى:

```
1. import java.sql.*;
2. class Javadb1 {
3.  public static void main (String arg[ ] )
4. {
5.       String URL = " Jdbc: odbc:javadb1";
6.       try{
7.       Class.forName = (" sun.jdbc.odbc.JdbcOdbcDriver");
8. Connection Con= DriverManager.getConnection (URL, " ", " ");
9. Statement stmt = Con. CreatStatement();
10. String qry1 ="select * from Basicdata";
11. ResultSet  rcord= stmt.executeQuery (qry1);
12. While (rcord. next( ) ){
13. System.out.println(rcord.getString     ("ID")     +     ";"     +     rcord.
```

```
    getString("Name")+","+rcord.getString("Birthdata")+","                +
    rcord.getString("Education")+","  +  rcord.getString  ("Tel")+","  +
    rcord.getString ("Address"));
14. }
15. stmt.close();
16. }catch (Exception e){
17. System.out.println("Error " + e. to string());
18.    }
19. }
20. }
```

فى هذا المثال تم الآتى:

- فى السطر رقم 1 أوضحنا أننا نريد استخدام الحزمة java.sql لأنها تحتوى على الفصائل والـInterfaces مثل ....,RecordSet,Connection.
- فى السطر رقم 5 تم تخزين عنوان الوصلة Connection فى المتغير URL.
- فى السطر رقم 6 قمنا بإنشاء معالج استثناء Exception Handling لأن الدوال التى ستستخدمها تلقى استثناءات.
- فى السطر رقم 7 قمنا بتحميل Load المشغل Driver الموجود فى لغة جافا.
- فى السطر رقم 8 قمنا بإنشاء وصلة وذلك باستخدام Connection Interface وأعطيناها المعاملات (المتغير URL)الذى سبق وأنشأناه فى السطر 5 والذى يحتوى على عنوان الوصلة.
- قمنا بإنشاء هدف Instance من Statement Interface وذلك لفتح قناة مع قاعدة البيانات واستخدامها فى تنفيذ أوامر SQL وذلك من كما فى السطر 10.
- قمنا بكتابة أمر استعلام Query وذلك باستخدام الأمر ()stmt.executeQuery والذى يقوم بإرجاع سجلات فكان لابد من استخدام هدف Object (rcord)من نوع ResultSet لاستقبال هذه السجلات.
- فى السطر 12 أردت ان ادخل الهدف rcord فى Loop ولكن ما معنى ()rcord.next ؟
- عندما نقوم بإرسال استعلام إلى قاعدة البيانات يكون هناك مؤشر cursor يشير إلى ما قبل السجل الأول فتقوم الدالة next بإعلامه أن يتحرك إلى السجل الأول.

- بعد ذلك يقوم السطر 13 بعرض نتائج الاستعلام وذلك بأن يجعل الهدف rcord يقوم باستدعاء الدالة getString والتى تأخذ معامل واحد وهو أما اسم الحقل أو رقم الحقل فمثلا الحقل "ID" هو الحقل رقم 1 فبدلا من اكتب "ID" يمكن اكتب 1 وهكذا يتضح من السطر أننى أريد أن أعرض سجل مكون من حقول ID , Tel , Address Education, Birth Data, Name.
- فى السطر 16 أقوم بإغلاق قناة التعامل مع قاعدة البيانات أى بشكل آخر أوضح أننى انتهيت ولم أعد أريد تنفيذ أوامر أخرى مع قاعدة البيانات.
- فى السطر 17-19 أقوم بمعالجة الاستثناء لو حدث خطأ نتج عن عدم وجود قاعدة البيانات أو أن الربط لم يتم أو لأى استثناء قد يحدث.

هذا المثال يوضح كيفية عرض نتائج استعلام فى بيئة سطر الأوامر Command Prompt وسنتعرض لاحقا لمثال من خلال البيئة الرسومية GUI ولكن قبل ذلك يهمنا التعرض بشئ من التفصيل لـ ResultSet Interface.

الدوال المعرفة فى ResultSet Interface

يوجد العديد من الدوال المعرفة فى ResultSet Interface للتعامل مع البيانات الموجودة فى قاعدة البيانات فمثلا عند التعامل مع حقل يحوى التاريخ فى قاعدة البيانات فلابد من وجود دالة للتعامل مع مثل هذا النوع من البيانات واليك بعض هذة الدوال.

```
getDate(String)
getDouble(String)
getFloat(String)
getInt(String)
getLong(String)
getString(String)
```

ويتوقف استخدامك للدالة على نوع البيانات الموجودة فى حقول قاعدة البيانات.

كما أن ResultSet Interface يحتوى على الدوال التى تساعدك على التحرك Navigate فى قاعدة البيانات وهى.

afterLast() التحرك الى ما بعد السجل الأخير فى قاعدة البيانات.

beforeFirst() التحرك الى ما قبل السجل الأول فى قاعدة البيانات.

()first التحرك إلى السجل الأول فى قاعدة البيانات.

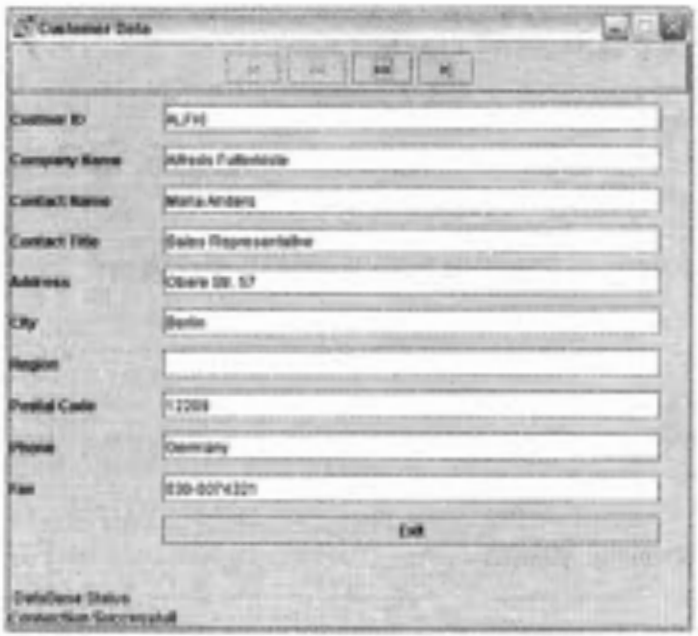
()Last التحرك إلى السجل الأخير فى قاعدة البيانات.

()previous التحرك إلى ما قبل السجل الحالى فى قاعدة البيانات.

()next التحرك إلى السجل التالى فى قاعدة البيانات.

والآن جاء موعدنا مع مثال يستخدم واجهة المستخدم الرسومية GUI وفى هذا المثال سوف نستخدم قاعدة البيانات NorthWind المدمجة فى برنامج قاعدة البيانات Access وستجدها على القرص المدمج مع الكتاب لذلك قم بعمل الخطوات من 3-10 السابق ذكرها لعمل وصلة فى ODBC وذلك بفتح لوحة التحكم ثم الوصول الى ايقونة ODBC ثم قم بعمل وصلة واعطها اسم وليكن NorthWind و اربطها مع قاعدة البيانات NorthWind التى غالبا ستجدها فى الدليل Folder الذى يحتوى ملفات Ms Office.

وسوف نتعامل مع جدول واحد من قاعدة البيانات NorthWind وهو جدول العملاء Custmers وهذا الجدول يحتوى على عشرة حقول تمثل بيانات العملاء والشكل التالى يبين البرنامج بعد بناء الواجهة الرسومية.



والشكل كما هو واضح مكون من 10 JTextField و12 JLabel و5 Jbutton و 4 Jpanel

واليك الكود.

يجب ترجمة وتنفيذ هذا المثال باستخدام JDK1.4

```
1. ●/
2. 2 ● TestDB.java
3. ●
4. Created on January 25, 2001, 6:17 PM
5. /●
6. ●● /
7. ●
8. /●
9. public class TestDB extends javax.swing.JFrame {
10. ●● /Creates new form TestDB /●
11. public TestDB (){
12. initComponents();
13. setupPosition();    / /Method to put Form in screen center
14. setupIcon();   / /Method to cahnge Form Icon
15. setSize(550,500); //to set Form Size
16. startDB();    / /to satrt DB upon Intializing
17. }
18. / /Implementing setupIcon
```

```
19. private void setupPosition()
20. {
21. java.awt.Dimension
      screen=java.awt.Toolkit.getDefaultToolkit().getScreenSize();
22. java.awt.Dimension frm=this.getPreferredSize();
23. if(frm.height>screen.height)
24. frm.height=screen.height;
25. if(frm.width>screen.width)
26. frm.width=screen.width;
27. setLocation((screen.width-550)/2,(screen.height-500)/2);
28. setTitle("Customer Data");
29. }
30. private void setupIcon()
31. {
32. java.awt.Image
      img=java.awt.Toolkit.getDefaultToolkit().getImage("e\\:icn.jpg");
33. setIconImage(img);
34. }
35. private String url="jdbc:odbc:NorthWind";
36. //Statement stmt=null;
37. java.sql.Statement stmt=null;
38. java.sql.Connection con=null;
39. java.sql.ResultSet rcord=null;
40. //Implementing of the DataBase
41. private void startDB()
42. {
43. try
44. {
45. Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
46. con=java.sql.DriverManager.getConnection(url,"kadry","18111970");
47. stmt=con.createStatement( java.sql.ResultSet.TYPE_SCROLL_INSENSITIVE,
48. java.sql.ResultSet.CONCUR_UPDATABLE);
49. rcord=stmt.executeQuery("SELECT * FROM Customers");
```

```
50. if(rcord.next())
51. {
52. if(rcord.isFirst())
53. {
54. jButton1.setEnabled(false);
55. jButton2.setEnabled(false);
56. }
57. jTextField1.setText(rcord.getString(10));
58. jTextField2.setText(rcord.getString(2));
59. jTextField3.setText(rcord.getString(3));
60. jTextField4.setText(rcord.getString(4));
61. jTextField5.setText(rcord.getString(5));
62. jTextField6.setText(rcord.getString(6));
63. jTextField7.setText(rcord.getString(7));
64. jTextField8.setText(rcord.getString(8));
65. jTextField9.setText(rcord.getString(9));
66. jTextField10.setText(rcord.getString(10));
67. }
68. jLabel6.setText("Connection Successful");
69. {catch(Exception e){
70. jLabel6.setText("Connection Not Successfull"+e.toString());
71. }
72. }
73. /** /This method is called from within the constructor to
74. initialize the form.
75. WARNING: Do NOT modify this code. The content of this method is
76. always regenerated by the Form Editor.
77. /*/
78. private void initComponents(){
79. jPanel1 = new javax.swing.JPanel();
80. jButton1 = new javax.swing.Jbutton();
81. jButton2 = new javax.swing.Jbutton();
82. jButton3 = new javax.swing.Jbutton();
83. jButton4 = new javax.swing.Jbutton();
```

```
84. jPanel3 = new javax.swing.Jpanel();
85. jLabel1 = new javax.swing.Jlabel();
86. jLabel2 = new javax.swing.Jlabel();
87. jLabel4 = new javax.swing.Jlabel();
88. jLabel3 = new javax.swing.Jlabel();
89. jLabel7 = new javax.swing.Jlabel();
90. jLabel71 = new javax.swing.Jlabel();
91. jLabel72 = new javax.swing.Jlabe();
92. jLabel73 = new javax.swing.Jlabel();
93. jLabel74 = new javax.swing.Jlabel();
94. jLabel75 = new javax.swing.Jlabel();
95. jPanel2 = new javax.swing.Jpanel();
96. jLabel5 = new javax.swing.Jlabel();
97. jLabel6 = new javax.swing.Jlabel();
98. jPanel4 = new javax.swing.Jpanel();
99. jTextField1 = new javax.swing.JtextField();
100.    jTextField2 = new javax.swing.JtextField();
101.    jTextField3 = new javax.swing.JtextField();
102.    jTextField4 = new javax.swing.JtextField();
103.    jTextField5 = new javax.swing.JtextField();
104.    jTextField6 = new javax.swing.JtextField();
105.    jTextField7 = new javax.swing.JtextField();
106.    jTextField8 = new javax.swing.JtextField();
107.    jTextField9 = new javax.swing.JtextField();
108.    jTextField10 = new javax.swing.JtextField();
109.    jButton5 = new javax.swing.Jbutton();
110.    setResizable(false);
111.    addWindowListener(new java.awt.event.WindowAdapter(){
112.    public void windowClosing(java.awt.event.WindowEvent evt(){
113.    exitForm(evt);
114.    }
115.    });
116.    jPanel1.setBorder(new
    javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
117.    jPanel1.setPreferredSize(new java.awt.Dimension(40, 40));
118.    jButton1.setText;(">|")
119.    jButton1.addActionListener(new java.awt.event.ActionListener(){
120.    public void actionPerformed(java.awt.event.ActionEvent evt){
121.    jButton1ActionPerformed(evt);
122.    }
123.    });
124.    jPanel1.add(jButton1);
```

```
125.    jButton2.setText(">>");

126.    jButton2.addActionListener(new java.awt.event.ActionListener (){
127.    public void actionPerformed(java.awt.event.ActionEvent evt ){
128.    jButton2ActionPerformed(evt);
129.    }
130.    });
131.    jPanel1.add(jButton2);
132.    jButton3.setText("<<");

133.    jButton3.addActionListener(new java.awt.event.ActionListene (){
134.    public void actionPerformed(java.awt.event.ActionEvent evt(){
135.    jButton3ActionPerformed(ev);
136.    }
137.    });
138.    jPanel1.add(jButton3);
139.    jButton4.setText("|<");

140.    jButton4.addActionListener(new java.awt.event.ActionListene(){
141.    public void actionPerformed(java.awt.event.ActionEvent evt} (

142.    jButton4ActionPerformed(evt);
143.    }
144.    });
145.    jPanel1.add(jButton4);
146.    getContentPane().add(jPanel1, java.awt.BorderLayout.NORTH);
147.    jPanel3.setLayout(new
        java.awt.FlowLayout(java.awt.FlowLayout.CENTER, 5, 10));

148.    jPanel3.setPreferredSize(new java.awt.Dimension(100, 100));
149.    jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.LEF);
150.    jLabel1.setText("Custner ID");
151.    jLabel1.setPreferredSize(new java.awt.Dimension(100, 23));
152.    jPanel3.add(jLabel1);
153.    jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
154.    jLabel2.setText("Company Name");
155.    jLabel2.setPreferredSize(new java.awt.Dimension(100, 23));
156.    jPanel3.add(jLabel2);
157.    jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
158.    jLabel4.setText("Contact Name");
159.    jLabel4.setPreferredSize(new java.awt.Dimension(100, 23));
160.    jPanel3.add(jLabel4);
161.    jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
```

```
jLabel3.setText("Contact Title");
jLabel3.setPreferredSize(new java.awt.Dimension(100, 23));
jPanel3.add(jLabel3);
jLabel7.setText("Address");
jLabel7.setPreferredSize(new java.awt.Dimension(100, 23));
jPanel3.add(jLabel7);
jLabel71.setText("City");
jLabel71.setPreferredSize(new java.awt.Dimension(100, 23));
jPanel3.add(jLabel71);
jLabel72.setText("Region");
jLabel72.setPreferredSize(new java.awt.Dimension(100, 23));
jPanel3.add(jLabel72);
jLabel73.setText("Postal Code");
jLabel73.setPreferredSize(new java.awt.Dimension(100, 23));
jPanel3.add(jLabel73);
jLabel74.setText("Phone");
jLabel74.setPreferredSize(new java.awt.Dimension(100, 23));
jPanel3.add(jLabel74);
jLabel75.setText("Fax");
jLabel75.setPreferredSize(new java.awt.Dimension(100, 23));
jPanel3.add(jLabel75);
getContentPane().add(jPanel3, java.awt.BorderLayout.WEST);
jPanel2.setLayout(new java.awt.BorderLayout());
jLabel5.setFont(new java.awt.Font("Dialog", 0, 12));
jLabel5.setText(" DataBase Status");
jPanel2.add(jLabel5, java.awt.BorderLayout.CENTER);
jLabel6.setForeground(new java.awt.Color(255, 0, 0));
jLabel6.setText("jLabel6");
jPanel2.add(jLabel6, java.awt.BorderLayout.SOUTH);
getContentPane().add(jPanel2, java.awt.BorderLayout.SOUTH);
jPanel4.setLayout(new
    java.awt.FlowLayout(java.awt.FlowLayout.CENTER, 5, 10));
jTextField1.setPreferredSize(new java.awt.Dimension(400, 23));
jTextField1.addActionListener(new java.awt.event.ActionListener (){

public void actionPerformed(java.awt.event.ActionEvent evt){
jTextField1ActionPerformed(evt);
}
});
jPanel4.add(jTextField1);
jTextField2.setPreferredSize(new java.awt.Dimension(400, 23));
jPanel4.add(jTextField2);
jTextField3.setPreferredSize(new java.awt.Dimension(400, 23));
```

```
203.    jPanel4.add(jTextField3);
204.    jTextField4.setPreferredSize(new java.awt.Dimension(400, 23));
205.    jPanel4.add(jTextField4);
206.    jTextField5.setPreferredSize(new java.awt.Dimension(400, 23));
207.    jPanel4.add(jTextField5);
208.    jTextField6.setPreferredSize(new java.awt.Dimension(400, 23));
209.    jPanel4.add(jTextField6);
210.    jTextField7.setPreferredSize(new java.awt.Dimension(400, 23));
211.    jPanel4.add(jTextField7);
212.    jTextField8.setPreferredSize(new java.awt.Dimension(400, 23));
213.    jPanel4.add(jTextField8);
214.    jTextField9.setPreferredSize(new java.awt.Dimension(400, 23));
215.    jPanel4.add(jTextField9);
216.    jTextField10.setPreferredSize(new java.awt.Dimension(400, 23));
217.    jPanel4.add(jTextField10);
218.    jButton5.setText("Exit");
219.    jButton5.setPreferredSize(new java.awt.Dimension(400, 23));
220.    jButton5.addActionListener(new java.awt.event.ActionListener (){
221.    public void actionPerformed(java.awt.event.ActionEvent evt){
222.    jButton5ActionPerformed(evt);
223.    }
224.    });
225.    jPanel4.add(jButton5);
226.    getContentPane().add(jPanel4, java.awt.BorderLayout.CENTER);
227.    pack();
228.    }
229.    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt){
230.     // //Add your handling code here:
231.    System.exit(0);
232.    {
233.    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt){
234.     // //Add your handling code here:
235.    try{
236.    if(rcord.last())
237.    {
238.    jButton1.setEnabled(true);
239.    jButton4.setEnabled(false);
240.    if(rcord.isLast())
241.    {
242.    jButton4.setEnabled(false);
```

```
243.    jButton3.setEnabled(false);
244.    jButton2.setEnabled(true);
245.    jButton1.setEnabled(true);
246.    }
247.    jTextField1.setText(rcord.getString(1));
248.    jTextField2.setText(rcord.getString(2));
249.    jTextField3.setText(rcord.getString(3));
250.    jTextField4.setText(rcord.getString(4));
251.    jTextField5.setText(rcord.getString(5));
252.    jTextField6.setText(rcord.getString(6));
253.    jTextField7.setText(rcord.getString(7));
254.    jTextField8.setText(rcord.getString(8));
255.    jTextField9.setText(rcord.getString(9));
256.    jTextField10.setText(rcord.getString(10));
257.    {
258.    //stmt.close;()
259.    {catch(Exception e){jLabel6.setText("Failed to move Last
    "+e.toString());
260.    }
261.    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){
262.    //Add your handling code here:
263.    try{
264.    if(rcord.previous())
265.    {
266.    jButton1.setEnabled(true);
267.    jButton4.setEnabled(true);
268.    if(rcord.isFirst())
269.    {
270.    jButton1.setEnabled(false);
271.    jButton2.setEnabled(false);
272.    }
273.    jTextField1.setText(rcord.getString(1));
274.    jTextField2.setText(rcord.getString(2));
275.    jTextField3.setText(rcord.getString(3));
276.    jTextField4.setText(rcord.getString(4));
277.    jTextField5.setText(rcord.getString(5));
278.    jTextField6.setText(rcord.getString(6));
279.    jTextField7.setText(rcord.getString(7));
280.    jTextField8.setText(rcord.getString(8));
281.    jTextField9.setText(rcord.getString(9));
282.    jTextField10.setText(rcord.getString(10));
```

```
283.    {
284.     //stmt.close;()
285.    {catch(Exception e){(Label6.setText("Failed to move Previous
    "+e.toString());
286.    }
287.    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt){
288.     //Add your handling code here:
289.    try{
290.    if(rcord.next())
291.    {
292.    jButton1.setEnabled(true);
293.    jButton4.setEnabled(true);
294.    if(rcord.isLast())
295.    {
296.    jButton4.setEnabled(false);
297.    jButton3.setEnabled(false);
298.    jButton1.setEnabled(true);
299.    jButton2.setEnabled(true);
300.    {
301.    jTextField1.setText(rcord.getString(1));
302.    jTextField2.setText(rcord.getString(2));
303.    jTextField3.setText(rcord.getString(3));
304.    jTextField4.setText(rcord.getString(4));
305.    jTextField5.setText(rcord.getString(5));
306.    jTextField6.setText(rcord.getString(6));
307.    jTextField7.setText(rcord.getString(7));
308.    jTextField8.setText(rcord.getString(8));
309.    jTextField9.setText(rcord.getString(9));
310.    jTextField10.setText(rcord.getString(10));
311.    jButton2.setEnabled(true);
312.     //jTexField5.setText(rcord.getString(4));
313.    {
314.     //stmt.close;()
315.    }catch(Exception e){jLabel6.setText("Failed to move Next
    "+e.toString());
316.    }
317.    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt)
```

```
318.      / /Add your handling code here:
319.     {
320.     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
321.      / /Add your handling code here:
322.     try{
323.     if(rcord.first())
324.     {
325.     if(rcord.isFirst())
326.     {
327.     jButton1.setEnabled(false);
328.     jButton2.setEnabled(false);
329.     jButton3.setEnabled(true);
330.     jButton4.setEnabled(true);
331.     }
332.     jTextField1.setText(rcord.getString(1));
333.     jTextField2.setText(rcord.getString(2));
334.     jTextField3.setText(rcord.getString(3));
335.     jTextField4.setText(rcord.getString(4));
336.     jTextField5.setText(rcord.getString(5));
337.     jTextField6.setText(rcord.getString(6));
338.     jTextField7.setText(rcord.getString(7));
339.     jTextField8.setText(rcord.getString(8));
340.     jTextField9.setText(rcord.getString(9));
341.     jTextField10.setText(rcord.getString(10));
342.     {
343.      / /stmt.close;()
344.     }catch(Exception e){jTextField1.setText("Failed to move First
    "+e.toString());}
345.     }
346.      ** /Exit the Application /*
347.     private void exitForm(java.awt.event.WindowEvent evt){
348.     System.exit(0);
349.     {
350.     ** /
351.     @param args the command line arguments
```

```
352.    /**
353.    public static void main(String arg[]){
354.    new TestDB().show();
355.    }
356.     // /Variables declaration - do not modify
357.    private javax.swing.JPanel jPanel4;
358.    private javax.swing.JButton jButton5;
359.    private javax.swing.JPanel jPanel3;
360.    private javax.swing.JPanel jPanel2;
361.    private javax.swing.JButton jButton4;
362.    private javax.swing.JButton jButton3;
363.    private javax.swing.JPanel jPanel1;
364.    private javax.swing.JButton jButton2;
365.    private javax.swing.JButton jButton1;
366.    private javax.swing.JTextField jTextField10;
367.    private javax.swing.JTextField jTextField9;
368.    private javax.swing.JTextField jTextField8;
369.    private javax.swing.JTextField jTextField7;
370.    private javax.swing.JTextField jTextField6;
371.    private javax.swing.JTextField jTextField5;
372.    private javax.swing.JTextField jTextField4;
373.    private javax.swing.JTextField jTextField3;
374.    private javax.swing.JTextField jTextField2;
375.    private javax.swing.JTextField jTextField1;
376.    private javax.swing.JLabel jLabel7;
377.    private javax.swing.JLabel jLabel6;
378.    private javax.swing.JLabel jLabel5;
379.    private javax.swing.JLabel jLabel4;
380.    private javax.swing.JLabel jLabel3;
381.    private javax.swing.JLabel jLabel75;
382.    private javax.swing.JLabel jLabel2;
383.    private javax.swing.JLabel jLabel74;
384.    private javax.swing.JLabel jLabel1;
385.    private javax.swing.JLabel jLabel73;
386.    private javax.swing.JLabel jLabel72;
387.    private javax.swing.JLabel jLabel71;
388.     // /End of variables declaration
389.    }
```

- هذا الكود به الكثير من الدوال والفصائل التي قد تعرضنا لها خلال الفصول السابقة ولكن مايهمنا هنا هو الكود المظلل حيث أنه يحتوى على الكود الخاص بانشاء والتعامل مع قاعدة البيانات وهذا ماسنشرحة الآن.
- فى السطور من 39-43 قمنا بإنشاء متغيرات للتعامل مع قاعدة البيانات وهذه المتغيرات هى.
- URL وهو متغير من نوع String لتحفظ فية عنوان الوصلة.
- Stmt وهو هدف من نوع Statement.
- con وهو هدف من نوع Connection.
- rcord وهو هدف من نوع ResultSet.
- وقد اعلنا عن هذة المتغيرات والاهداف خارج نطاق اى دالة حتى نستطيع استخدامهم من خلال اى دالة او منشأ Block فى الفصيلة.
- فى السطور من 45-79 نبنى الدالة ()startDB وقد استدعيناها فى دالة البناء الـ Constructor فى السطر 18 حتى يتم تنفيذها حال انشاء الهدف من الفصيلة TestDB.
- فى هذة الدالة(()startDB) قمنا بشحن الاهداف السابق اعلانها فى السطور من 39-43 فقمنا بتحميل المشغل كما فى السطر 49.
- فى السطر 50 شحن الهدف con بعنوان وصلة قاعدة البيانات url,username,password.
- فى السطر 51-52 شحن الهدف stmt ولكن مايهمنا هذة المرة هو وجود المعاملين.

ResultSet.TYPE_SCROLL_INSENSITIVE,

ResultSet.CONCUR_UPDATABLE

فهذان المعاملان هما اللذان يمنحانا القدرة على التحرك فى قاعدة البيانات Navigating أى أننا بدونهما نستطيع التحرك فى السجلات إلى الأمام فقط.

حاول حذفهما ثم اعد ترجمة البرنامج وتنفيذة وحاول التحرك للخلف او اول او اخر سجل فى قاعدة البيانات وانظر ماذا ترى!

- فى السطر 54 نقوم بتنفيذ استعلام فى الجدول Customers وسوف يحتفظ الهدف rcord بنتيجة هذا الاستعلام.

- فى السطر 55 أقوم بفحص ماذا كان مؤشر قاعدة البيانات يشير إلى السجل الأول.
- فى السطور من 57 -61 اقوم باختبار ما إذا كان المؤشر يشير فعلا للسجل الأول فإذا كان فعلا يشير إليه فيقوم بجعل زرى التحرك (أول سجل >| و سجل سابق للحالى<<)غير فعالين.
- فى السطور من 62-71 نقوم بسحب pull البيانات من قاعدة البيانات وعرضهم من خلال أدوات الكتابة jTextField .
- فى السطر 73 نقوم بعرض رسالة تفيد نجاح الاتصال بقاعدة البيانات من خلال أداة العنوان jLabel6 الموجودة أسفل الفورم وتعرض الرسائل باللون الأحمر.
- فى السطور 75-78 نقوم بمعالجة أى استثناء قد يحدث ونقوم بجعل jLabel6 يعرض رسالة تحمل الاستثناء الحادث.

لاحظ :

اننا جعلنا jLabel6 يعمل كشريط حالة Status Bar

- فى السطر 79 قوس إنهاء الدالة.
- وبذلك عند انشاء هدف من الفصيلة كما فى السطر413 يتم استدعاء الدالة startDB() .
- فى السطور من 284-311 كود زر التحرك إلى أخر سجل وفى السطور 293-297 نقوم فى حالة إذا كان المؤشر يشير إلى السجل الأخير بجعل زرى التحرك (لأخر سجل |< والسجل التالى <<)غير فعالان.
- فى السطور من 313-383 كود زر التحرك إلى السجل السابق <<
- فى السطور 320-324 نقوم فى حالة اذا كان المؤشر يشير الى السجل الاول بجعل زرى التحرك(لاول سجل >| والسجل السابق <<)غير فعالان.
- فى السطور من 340-370 كود زر التحرك إلى السجل التالى << وفى السطور 353-374 نقوم فى حالة إذا كان المؤشر يشير إلى السجل الاخير بجعل زرى التحرك (لأخر سجل |< والسجل التالى <<)غير فعالان.
- فى السطور من 376-402 كود زر التحرك إلى السجل الأول >|

- فى السطور 387-381 نقوم فى حالة إذا كان المؤشر يشير إلى السجل الأول بجعل زرى التحرك(الأول سجل <| والسجل السابق <<)غير فعالان.
- لاحظ أننا وضعنا الكود الخاص بكل زر بين try-catch لمعالجة أى استثناء قد يحدث وإذا حدث استثناء يتم عرضة من خلال إداة العنوان lLabel6.

وبهذا نكون قد انتهينا من فصل قواعد البيانات وقد تعرضنا لقدر بسيط فى التعامل مع قواعد البيانات ولنا جولات أخرى فى التعامل مع قواعد البيانات فى الجزء الثانى من الكتاب.

فى Oracle

يتم استعمال أوامر SQL التى شرحناها مباشرة لتحقيق عمليات قواعد البيانات ولا تحتاج إلى المقدمات والأوامر التى استعملناها فى اللغات الأخرى.
لذلك هى من أسهل اللغات لإعداد تطبيقات قواعد البيانات.

الفصل الرابع عشر

تقارير التطبيقات

في هذا الفصل Programming Concepts 14

في هذا الفصل نتناول موضوع تقارير التطبيقات وما له من أهمية وخاصة في تطبيقات قواعد البيانات وذلك من خلال النقاط التالية

- أهمية التقارير
- أنواع التقارير
- خطوات تصميم برنامج به تقارير في لغة VB.NET ,C#
- خطوات تصميم تقارير في Oracle
-

أهمية التقارير

تعتبر التقارير من أهم أجزاء برنامج قواعد البيانات ، فهي الصورة التي يراها المستفيد من البرنامج والتي تخرج فيها مخرجات البرنامج ، فمثلاً عند تصميم برنامج لمتابعة حسابات العملاء لابد من توفير مجموعة من التقارير منها :

- تقرير بحسابات العملاء خلال فترة.
- تقرير بحسابات عميل خلال فترة.
- تقرير بأرصدة العملاء وتقرير برصيد عميل معين ، وهكذا .

وهذه التقارير هي التي تعطي قيمة للبرنامج ، لأن البرنامج بدون تقارير يعتبر بلا فائدة لأنه ينفذ العمليات ولا يعطي النتائج ، وبالتالي يصبح وسيلة لحفظ البيانات فقط .

وكانت عملية تصميم التقارير قبل برمجة النوافذ تتطلب جهد من المبرمج ، حيث يقوم المبرمج بتصميم التقرير باستخدام أوامر البرمجة كما يهتم بضبط الهوامش وغيرها من العمليات الصعبة ، ولكن مع ظهور النوافذ ظهرت برامج منفصلة لتصميم التقارير ثم ربطها بالبرنامج ،ومن هذه البرامج برنامج مولد التقرير المسمى Crystal Report الذى يوفر معظم العمليات المطلوبة لتنظيم التقارير مع إمكانية ربطه بالبرنامج والتقرير الذى يتم تصميمه باستخدام Crystal Report يمكن استدعائه من برنامج مصمم باستخدام لغة VC++ أو باستخدام لغة Visual Basic مع مراعاة الاصدار فإذا سبق لك كتابة برامج باستخدام لغة VB فبالطبع تعرف كيفية استعمال Crystal Report فى تصميم التقارير المطلوبة .

أنواع التقارير

التقارير REPORTS

جزء مهم جدا في التطبيقات ، فالتقارير هي نتيجة عمل البرنامج (تطبيق) بدون تقارير بالتأكيد ليس للبرنامج قيمة ، فمثلاً ما الفائدة من إعداد برنامج ميزانية لا يخرج ويطبع تقارير الميزانية ، ما الفائدة من برنامج شئون عاملين وحفظ بياناتهم بدون تقارير عن العاملين مرتباتهم وإجازتهم وغيره.

ولذلك توجد أشكال مختلفة من التقارير منها .

1. التقارير الجدولية TABULAR

وهي التقارير المشهورة وهي عبارة عن قائمة LIST من البيانات مثل بيانات العملاء أو بيانات الطلبة أو غيره وتأخذ الشكل (1-14) .

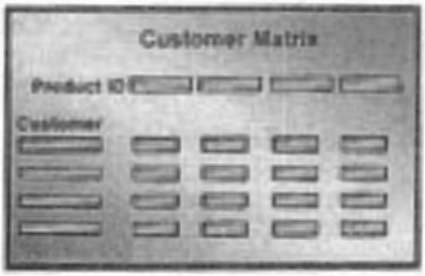
الشكل (1-14)

2. العناوين البريدية LABEL

وهي أحد أشكال التقارير عبارة عن إخراج البيانات في شكل عنوان بريدي LABEL مثل البيانات التي تأتي على الملصقات عن الجلدات بالتأكيد وصلت مجلة من الخارج ورأيت كيف تكتب بيانات على أوراق صغيره تلصق على المجلة هذه هي الملصق البريدي .

3. تقارير في شكل مصفوفة MATRIX REPOTS

وهي طرق لعرض بيانات في شكل مصفوفة MATRIX لمقابلتها ببعض مثل عرض أسماء المخازن أفقيا وعرض قيمة المخزون رأسيا مع السنوات ويأخذ الشكل (2-14).



الشكل (2-14)

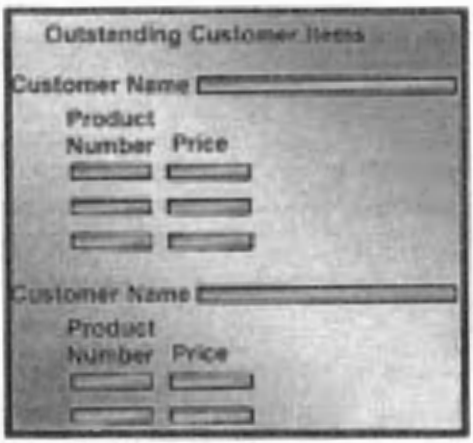
4. تقارير في رسومات بيانية

وهي عرض البيانات في شكل رسومات بيانيه.

وكل هذه الأنواع وغيرها الكثير سوف نتناولها بالتفصيل حيث يوفر مطور التقارير REPORT BUILDER جميع الإمكانيات المطلوبة لإعداد جميع أشكال التقارير.

5. تقارير تجميعية MASTER/DETAILS

وتأخذ الشكل (3-14)



الشكل (3-14)

– في هذا الشكل يتم ظهور التقرير لك في شكله النهائي .

في لغة VB.NET ,C#

خطوات تصميم برنامج به تقارير

1. تصميم البرنامج المطلوب وفي هذه الخطوة يتم تصميم البرنامج وتحقيق عملياته وترك عملية التقارير لآخر خطوة .
2. تصميم التقارير المطلوبة وحفظها في فهرس البرنامج .
3. إضافة أداة استدعاء التقارير إلى البرنامج Crystal Report Control.

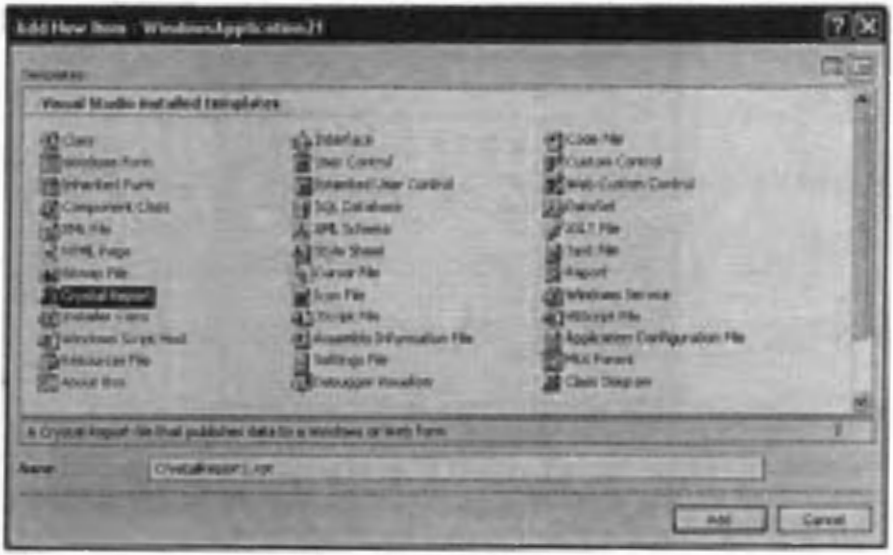
4. ربط أداة التقارير بالتقرير المطلوب إظهاره مع ضبط الخصائص .

5. كتابة أوامر استدعاء التقرير فى البرنامج.

ولتوضيح هذه الخطوات نتابع المثال التالى :

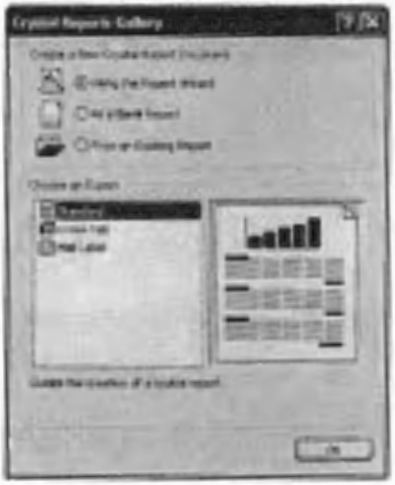
1- قم بإنشاء برنامج جديد من النوع Windows Application بالخطوات المعتادة

2- اختر القائمة Project من القائمة الرئيسية ومنها اختر Add New Item لإضافة تقرير للبرنامج تحصل على مربع حوار الاضافة Add New Itm كما فى الشكل (4-14) .



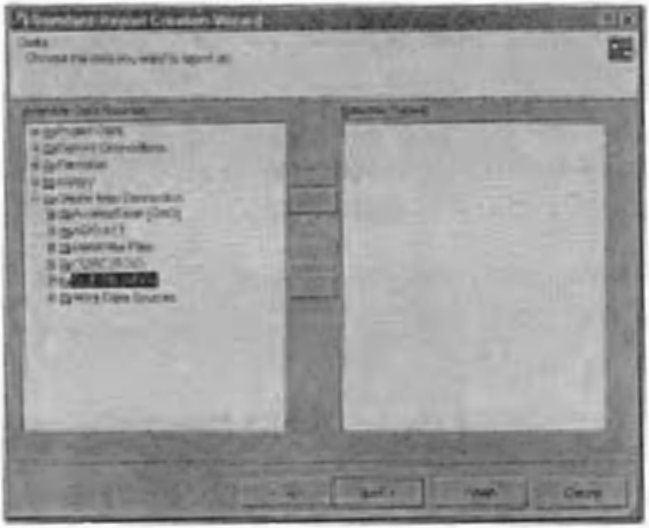
الشكل (4-14)

من مربع حوار الإضافة اختر Crystal Report كما فى الشكل ثم اضغط الزر Open تحصل على شاشة بأنواع التقارير التى يمكن تصميمها كما بالشكل (5-14).



المشكل (5-14)

4- اترك الاختيار الافتراضي Standard كما هو وهو التقرير التقليدي ثم اضغط الزر Ok لحصل على شاشة لتحديد مصدر بيانات التقرير كما بالشكل (6-14).

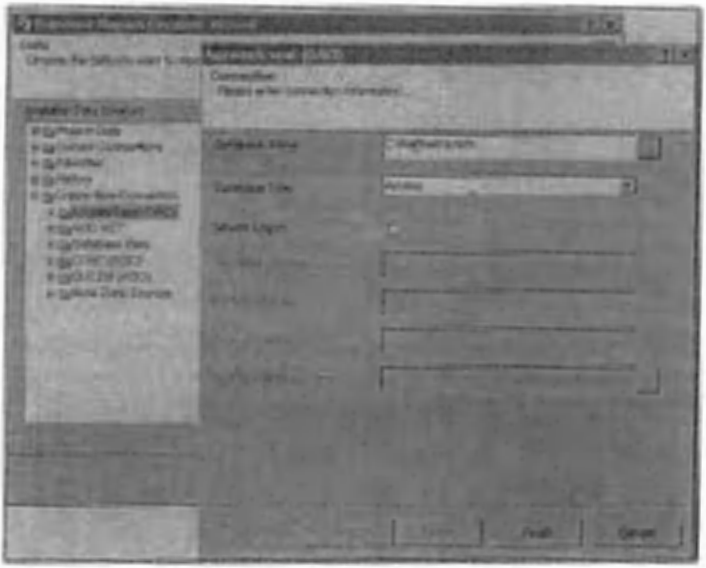


المشكل (6-14)

5- الشكل يعرض لك مشغلات قواعد البيانات DataBase Drivers لتحديد نوع قاعدة البيانات التي ترغب بناء التقرير عليها وفق هذا المثال نقوم بتجربة التقارير

على قاعدة البيانات NorthWind التى أشرنا اليها من النوع Ms Access لذلك حدد الاختيار Microsoft Jet OLE DB Provider الذى يشغل قاعدة بيانات من النوع Access كما فى الشكل السابق.

- ثم اضغط الزر Next تحصل على الشكل (7-14):



الشكل (7-14)

ملحوظة:

أضاف الاصدار 2005 امكانية اعداد تقرير مباشرة على ملفات MS Access

- ثم اضغط الزر Next تحصل على الشكل (8-14)

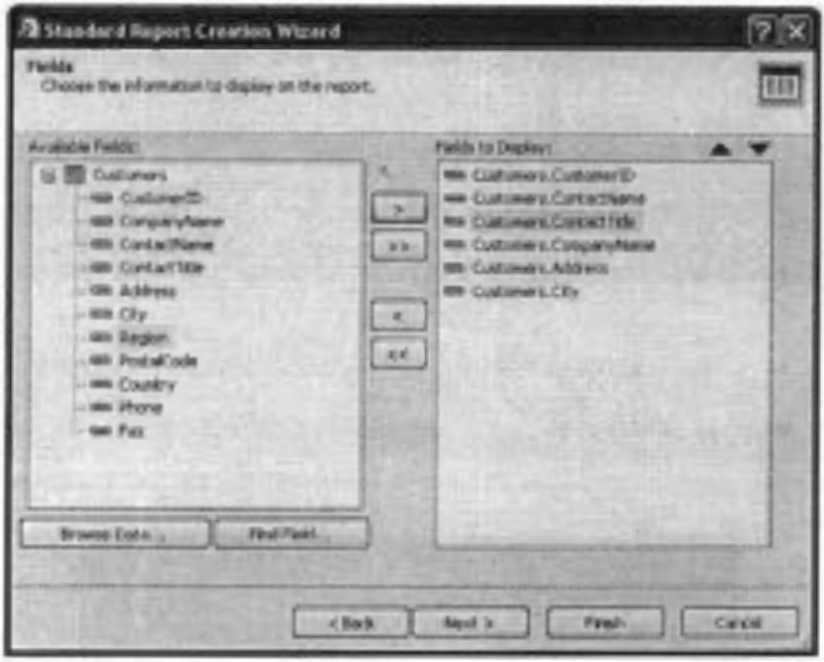
الشكل (14-8)

6- فى هذا الشكل يتم تحديد ملف قاعدة البيانات الذى نرغب التعامل معه لذلك اختر الملف NorthWind.Mdb بالضغط على الزر .. كما فى الشكل ثم اضغط على الزر Next لحصل على الشكل (14-9).

الشكل (14-9)

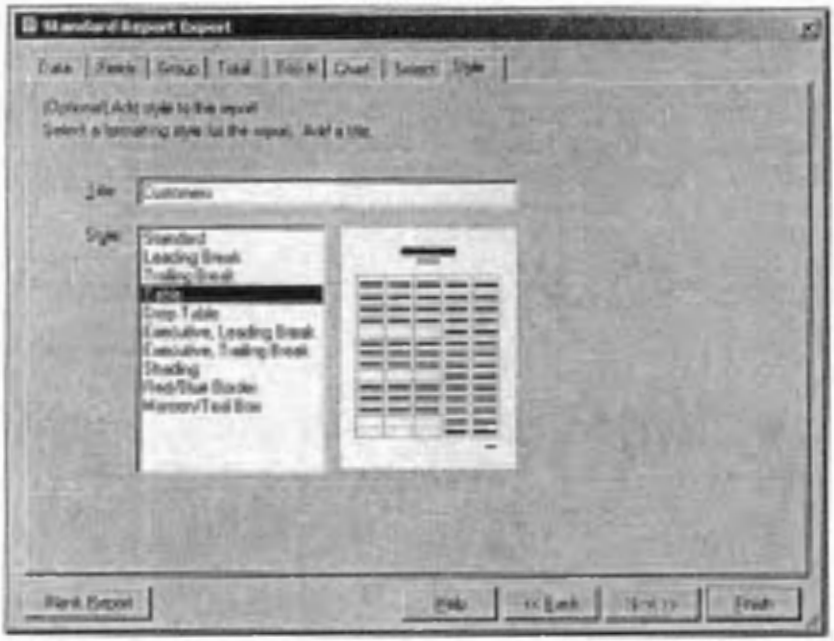
فى هذا الشكل يتم عرض قاعدة البيانات المفتوحة مع عرض جداولها ، اختر الجدول Customers بالضغط عليه مرتين كما فى الشكل السابق.

7- اضغط تبويب Fields وفيها يتم عرض الجداول لتحديد الحقول المطلوب إضافتها للتقرير ، اختر جدول Customers ثم اضغط الزر > All لإضافة جميع حقول الجدول (أو أضف حقل بحقل) كما بالشكل (10-14).



الشكل (10-14)

1. اضغط تبويب Total :4 وفيه يتم تحديد الحقول التى يصلح تجميعها تلقائياً فى هذه الحالة يصلح تجميع لتعطى حقول مجاميع . و لاحظ أسفل الحقل TOTAL FIELD يوجد عدة دوال مثل SUM,AVERAGE,.... جرب اختيار ما تشاء.
2. اضغط تبويب Select :5 وفيها يتم تحديد شرط معين بحيث تظهر السجلات التى توافق هذا الشرط فقط.
3. اضغط تبويب Style :6 وهى الخطوة الأخيرة تحصل على الشكل (11-14).



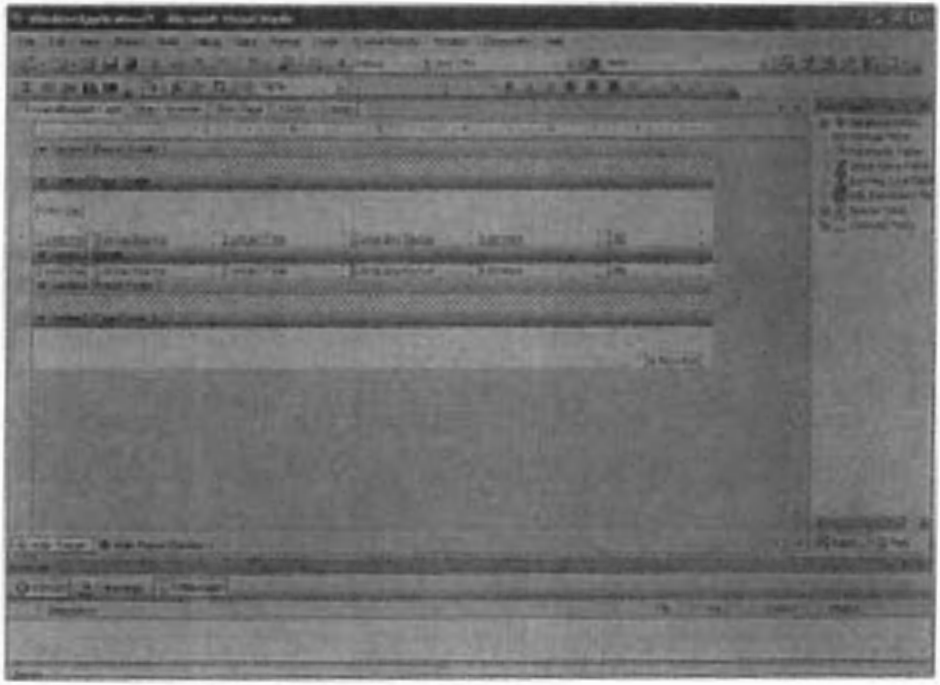
الشكل (11-14)

1- ويتم في هذا الشكل تحديد عدة نقاط .

- الشكل المطلوب للتقرير.
- عنوان التقرير.

اكتب عنوان للتقرير وليكن Customers Report وذلك في خانة الكتابة المقابلة للعنوان Title.

- من مربع الاختيار (أسفل العنوان Style) اختار الشكل المطلوب للتقرير جرب كل الاختيارات تلاحظ ظهور شكل تقريبي للتقرير بجوار مربع الاختيارات أختر النوع Table ليعطي الشكل التقليدي للتقرير ثم اضغط الزر Finish تحصل على الشكل (14-12).



الشكل (12-14)

الشكل (12-14)عبارة عن التقرير الذى تم تصميمه بالخطوات السابقة والشروط التى حددناها فمثلاً نلاحظ :

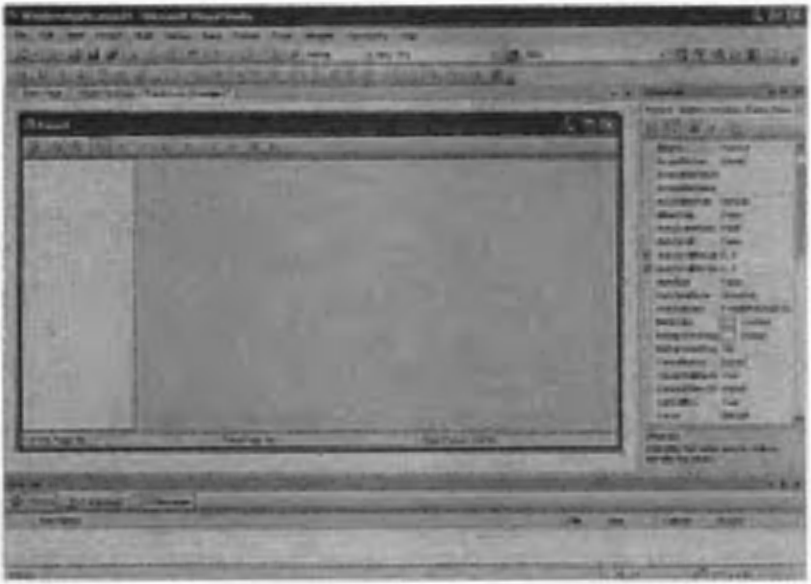
- عرض جميع حقول الملف.
- ترتيب البيانات أبجدي حسب الرقم (NO1).
- إضافة العنوان الذى حددناه كعنوان للتقرير.
- إضافة خانة للمجموع (Total) للحقل إذا حددت ذلك.
- عرض التقرير بالشكل الجدولى الذى حددناه فى خطوة Style هذه هى المواصفات التى حددناها ولكن بعد عرض التقرير يمكنك التعديل فيه .

يمكنك أجراء التعديلات التى تراها مثل تغير نوع الخط ، إضافة حقل نصى أو حقل بيانات وغير ذلك ولكن قبل الإسهاب فى شرح إمكانيات برنامج التقارير نكمل أولاً خطوات البرنامج ثم نعاود شرح برنامج التقارير .

13. أحفظ هذا التقرير وذلك بالاسم REPORT فى الفهرس REPORT الخاص بالبرنامج

ثم انهى برنامج التقارير Crystal Report

- وقع اداة CrystalReportViewer من مربع الأدوات Tool Box وغير حجمها كما فى الشكل (13-14).



الشكل (13-14)

اعرض خصائص الأداة CrystalReportViewer واختر منها الخاصية Report Source واختر منها Browse تحصل على مربع حوار فتح ملف لتحديد ملف التقرير الذى ترغب فى عرضه حدد ملف التقرير الذى أعددناه من قبل كما فى الشكل (14-14).

الشكل (14-14)

حدد الملف ثم اضغط الزر Open بهذا تنتهى الخطوات المطلوبة لإنشاء التقرير وربطه ويبقى مشاهدة النتيجة لذلك قم بتشغيل البرنامج تحصل على التقرير كما فى الشكل (15-14).

الشكل (15-14)

جرب استعمالها بتكبير وتصغير محتوياتها ثم إغلاقها بالنقر على زر Close تعود إلى البرنامج مرة أخرى .

بهذا تنتهى تصميم خطوات برنامج يستدعى تقرير والتى خصصناها فى أول الدرس وهذا هو أبسط شكل ولكن معظم البرامج أكثر تعقيداً من ذلك فمعظم البرامج تحتاج إلى أكثر من تقرير فما عليك إلا إضافة قائمة اختيارات بها أسماء التقارير المطلوبة وربطها واستدعائها .

إمكانيات برنامج التقارير Crystal Report

فى هذه الفقرة نعود إلى برنامج التقارير وإمكانياته

أنواع التقارير .

يوفر برنامج التقارير مجموعة من التقارير تظهر عند إنشاء تقرير جديد ولمشاهدة ذلك أختر New من قائمة File تحصل على مربع حوار بانواع التقارير نعرضها كما يلى :

التقرير التقليدى (Standard)

ويقوم بتصميم الشكل التقليدى للتقرير وهو عرض البيانات أما فى شكل جدول أو مجموعات كما فى المثال السابق .

النوع (Form)

وهو عرض البيانات فى شكل نموذج Form ويفيد فى حالة الرغبة فى تصميم شهادة أو دعوة أو كارت أو كارتيه أو ايصال مرتب.

النوع (Form Letter)

وهو يفيد فى تجهيز خطابات تأخذ بياناتها من قاعدة بيانات ،نفرض ان لديك قاعدة بيانات عملاء به 5000 عميل وترغب فى ارسال خطاب تهنئة اليهم جميعا تستخدم هذا النوع من التقرير فى كتابة بيانات خطاب التهنئة مع تحديد حقل الاسم والعنوان من جدول العملاء من قاعدة البيانات .

النوع (Cross Tab)

وفيه يتم عرض البيانات بشكل مختصر وهو مفيد فى حالة الرغبة فى معرفة معلومات سريعة مثل أرصدة الاصناف فى المخازن حيث يقوم بعرض المخازن والأصناف والأرصدة والمجاميع الأفقية والرأسية .

النوع (Mail Label)

وهو إخراج البيانات فى شكل ملحقات بريدية (Table) فمثلاً عند الرغبة فى طباعة شكل كارنيهات للعاملين يتم تصميم شكل الكارنيه فى شكل Lable وعند عرض البيانات تخرج فى هذا الشكل ، وبالمثل عند الرغبة فى تصميم شكل ايصال المرتبات للموظفين .

النوع (Drill Down)

وفيه يتم عرض البيانات فى إجمالى وتفاصيل Master /Detail مثل بيانات الفواتير وإذون الوارد وإذون المنصرف.

مساعدات التصميم

يوفر برنامج التقارير مجموعة كبيرة من المساعدة المطلوبة أثناء عملية التصميم مثل إمكانية رسم مستطيلات ورسم خطوات وغيره يمكنك الاستفادة منها.

اعداد تقارير بشروط

يمكن إعداد تقارير بشروط مثل عرض موظفي إدارة معينة أو عرض جميع الموظفين أصحاب المرتبات أعلى من قيمة معينة وهكذا ويتم ذلك من خلال الخاصية SelectionFormula التي تحدد شرط عرض البيانات.

```
CrystalReportViewer1.SelectionFormula = "CustId>125"
```

في هذه الصيغة يتم عرض بيانات العملاء الذين هم أكواد أكبر من 125.

ويمكن بتغيير هذه المعادلة ان نتحكم في استخراج التقارير المطلوبة.

بناء تقرير يعتمد على شرط

في الفقرة السابق شرحنا كيفية بناء واستدعاء تقرير بدون شروط ولكن لا يخلو تطبيق من تقارير بشروط مثل عرض بيانات جهة بمعلومية رقم الجهة أو بمعلومية اسم الجهة أو عرض بيانات تم ادخالها (مثل الواردات) خلال فترة محددة ولتوضيح ذلك تابع الخطوات التالية:

1- عد إلى البرنامج السابق وأضف مربع نص TextBox1 إلى Form1 ثم قم بتغيير سطور أوامر استدعاء التقرير لتصبح كما يلي:

```
        Dim C As New Form2()
        Dim R As New CrystalReport1()
        Dim t As String
        t = TextBox1.Text
        R.CrystalReportViewer1.SelectionFormula = "{Algebat.GobaCode} =" & Int32.Parse(t)
        R.CrystalReportViewer1.ReportSource = R
        R.Show()
```

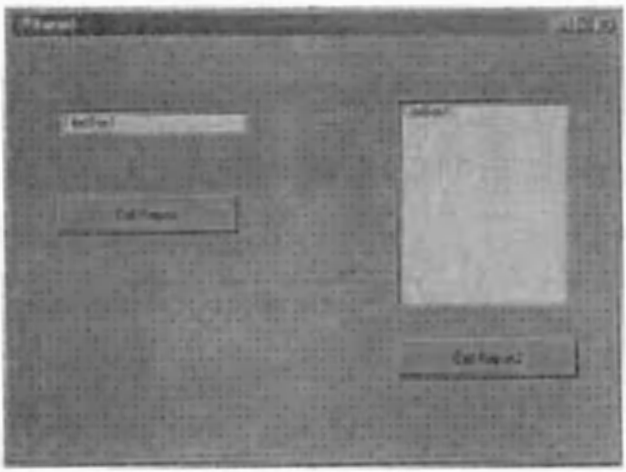
2- نفذ البرنامج واكتب رقم الجهة وليكن 11 ثم اضغط الزر Call Report لتحصل على شاشة التقارير وبها بيانات الجهة 11 كما في الشكل (16-14).

الشكل (14-16)

بناء تقرير يعتمد على اختيار من مجموعة اختيارات

فى الفقرة السابقة شرحنا كيفية بناء تقرير على شرط محدد وهو عرض بيانات جهة بمعلومية كود الجهة وفى هذه الفقرة نشرح بناء تقرير بناء على اختيار اسم جهة من قائمة اختيارات Combo Box ولتحقيق ذلك تابع الخطوات التالية

1- عد إلى البرنامج السابق وقم بإضافة List Box وزر أمر ليصبح التصميم كما فى الشكل (14-17).



الشكل (14-17)

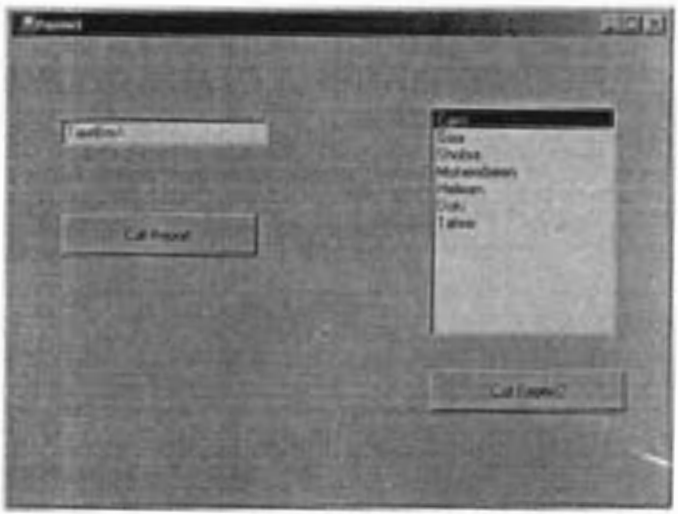
2- اكتب سطور ملء مربع الاختيارات ListBox1 كما فى السطور التالية.

```
        Dim cn As New SqlClient.SqlConnection()
        cn.ConnectionString = "data source=mssbi;initial catalog=invensorg;user ID=sa"
        Dim cmd1 As New SqlCommand()
        cmd1 = cn.CreateCommand
        Dim dr As SqlDataReader
        Dim dare As OleDbDataAdapter = New OleDbDataAdapter()
        cn.Open()
        Dim strcmd As String
        Dim ii As Integer
        strcmd = "SELECT * from Algebar"
        cmd1.CommandText = strcmd
        dr = cmd1.ExecuteReader(CommandBehavior.CloseConnection)
        While dr.Read()
            ListBox1.Items.Add(dr.Item("DeluName"))
        End While
```

- اضغط زر استدعاء التقرير الجديد Call Report2 مؤثين واكتب سطور استدعاء التقرير بالحدث Click كما فى السطور التالية.

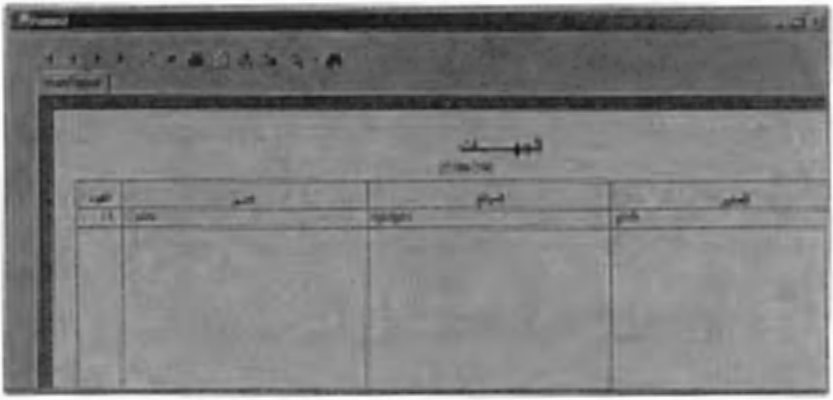
```
        Dim f As New Form2()
        Dim R As New CrystalReport1()
        Dim Del As String
        Del = ListBox1.SelectedItem
        f.CrystalReportViewer1.SelectionFormula = "{Algebar.DeluName} = '" & Del & "'"
        f.CrystalReportViewer1.ReportSource = R
        f.ShowDialog()
```

3- نفذ البرنامج لحصل على شاشة استدعاء التقرير وبها أسماء الجهات وزر استدعاء التقرير كما فى الشكل (18-14).



الشكل (18-14)

4- اضغط الزر Call Report2 بعد تحديد اسم الجهة Cairo تحصل على تقرير ببيانات هذه المنطقة كما في الشكل (19-14).



الشكل (19-14)

في Oracle

يوفر Oracle أداة منفصلة لإنشاء التقارير تسمى Report Builder توفر جميع إمكانيات التقارير حيث توفر.

تصميم وتشغيل التقارير DESIGNING & RUNNING REPORTS

- الأشكال العامة للتقارير.
- كيفية تشغيل التقارير.
- طرق عرض التقارير.
- عرض التقارير في بيئة العرض REPORT VIEWER.
- الخصائص العامة وأساسيات مطور التقارير REPOT BUILDER CONCEPTS.
- أجزاء مطور التقارير REPORT BUILDER COMPONENTS.
- تشغيل مطور التقارير REPORT BUILDER.
- لأجزاء الأساسية في مطور التقارير MAIN OBJECTS.
- استعمال معالج التقارير في إنشاء التقارير USING REPORT WIZARD.
- إنشاء تقرير جدولي TABULAR بسيط.

- طرق إنشاء استعلام QUERY الخاص بالتقارير.
- إضافة حقول تلخيص SUMMARY FIELDS للتقرير.
- أشكال أخرى للتقارير.
- الأشكال العامة للتقارير.

في هذه الفقرة وهذه الخطوات أردت فقط تقديم فكرة لك عن مطور التقارير REPORT BUILDER وكيفية استعماله في إنشاء تقرير ولكن يمكنك الرجوع الى كتاب منفصل حول مطور التقارير REPORT BUILDER.

خطوات تشغيل مطور التقارير REPORT BUILDER 9I,10 G

في هذا الفقرة نتناول بشكل سريع مطور التقارير Oracle 9I,1 G Report builder وذلك بشرح كيفية تشغيله والشاشات المختلفة Report wizard وذلك من خلال النقاط التالية :

- مقدمة عن Report builder 9I,10G.
- خطوات استعمال معالج التقارير Report wizard.
- الجديد في بناء التقارير في Report builder 9I,1 G.

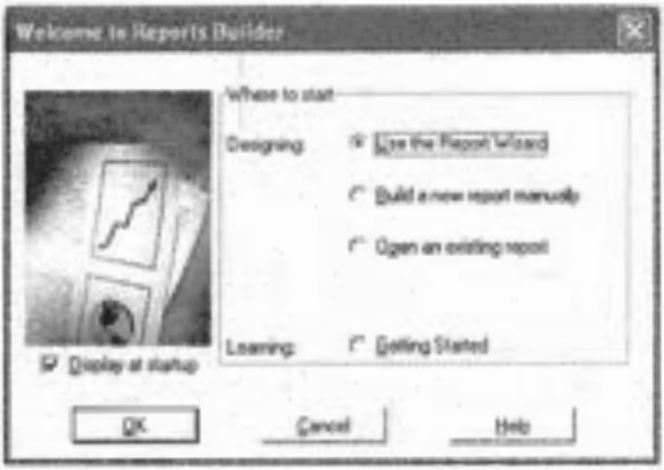
مقدمة عن Report builder 9I,10G

كما سبق وعرضنا Form builder 9I,10G ووجدنا كيف أصبح من السهل إنشاء نموذج يعمل في بيئة الإنترنت كذلك من الواضح في Report builder 9I,1 G سهولة أعداد تقرير يعرض في بيئة الإنترنت في متصفح صفحات الإنترنت Explore أو Netscape أو غيره وسوف يتضح ذلك من خطوات أعداد تقرير.

استعمال معالج التقارير Report wizard

تشبه خطوات معالج التقارير في هذا الإصدار بدرجة كبيرة معالج التقارير Report wizard في الإصدار السابق مع بعض الاختلافات البسيطة ويظهر ذلك كما يلي :

1- قم بتشغيل مطور التقارير Report builder بالخطوات المعتادة تحصل على أول شاشة كما بالشكل (20-14).



الشكل (14-20)

لاحظ أن هذا الشكل نفس الشاشة الموجودة في الإصدار السابق بها نفس الاختيارات وهي :

1- بناء تقرير باستعمال المعالج Report wizard.

2- بناء تقرير يدويا بدون استعمال المعالج Manually.

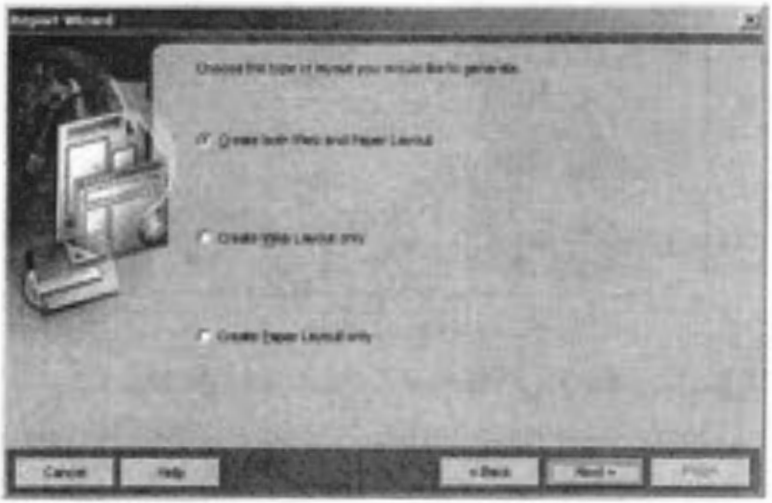
3- فتح ملف تقرير موجود بالفعل Open an existing Report.

اترك الاختيار الأول كما هو واضغط الزر OK بالماوس.

2- تحصل على شاشة الترحيب كما بالشكل (14-21) اضغط زر Next للمتابعة.

الشكل (14-21)

3- نحصل علي شاشة جديدة من شاشات معالج التقارير وهي بالفعل جديدة عن الإصدار السابق وبها اختيارات جديدة كما بالشكل (22-14).



الشكل (22-14)

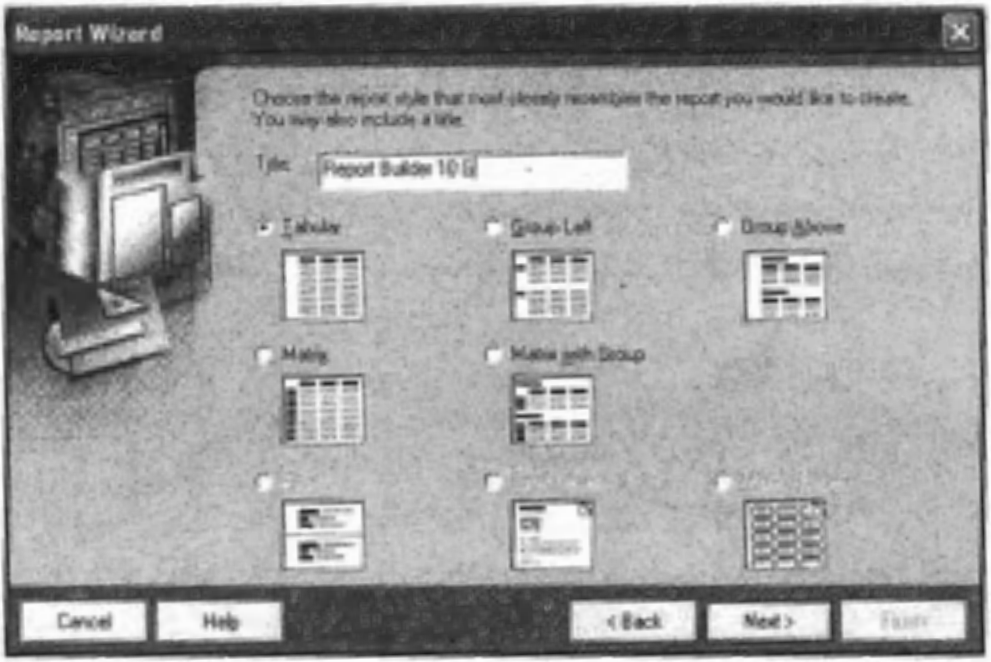
في الشكل (22-14) توجد 3 اختيارات.

الأول : يؤدي أي إنشاء تقرير للإنترنت و أخر للطباعة علي الورق وهما Web layout و Paper.

الثاني : تقرير للعرض علي صفحات الإنترنت فقط Web layout only.

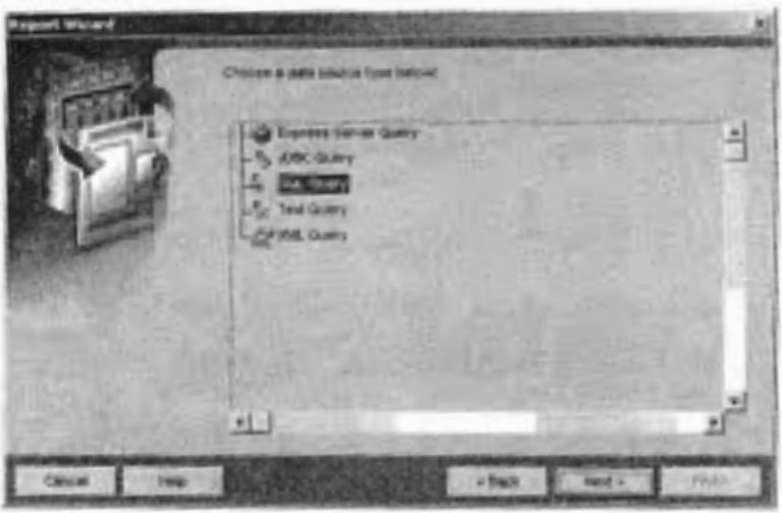
الثالث : تقرير يطبع علي الورق فقط Paper layout.

اترك الاختيار الأول كما هو واضغط بالماوس علي زر Next نحصل علي شاشة بأنواع التقارير المتاحة بشكل جديد كما بالشكل (23-14).



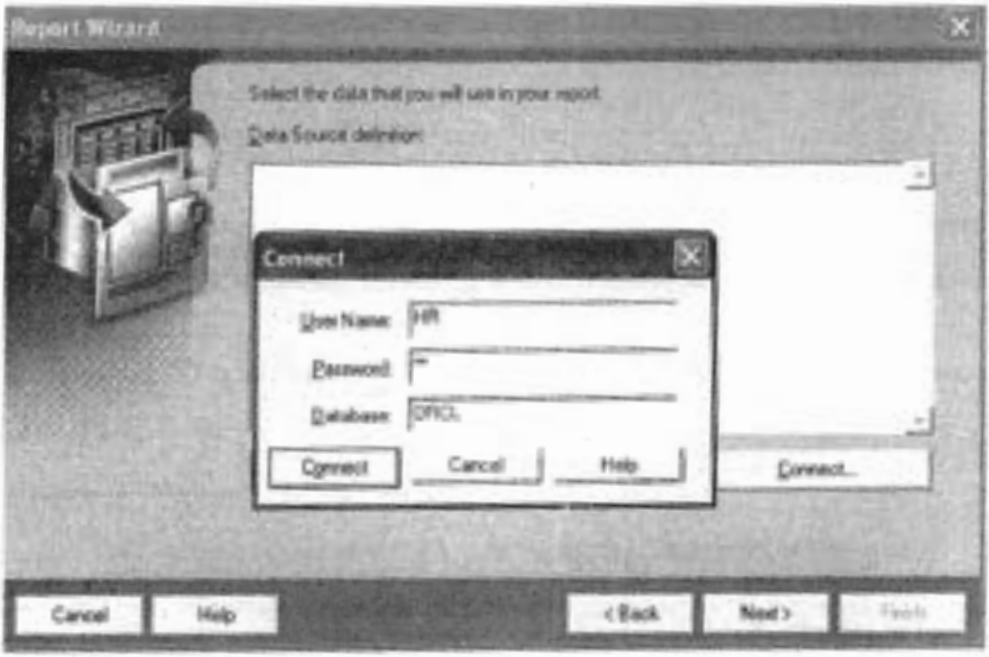
الشكل (14-23)

في الشكل 17 توجد خانة لعنوان التقرير اكتب عنوان للتقرير كما سبق وراجع الأنواع المختلفة للتقارير والتي شرحناها فيما سبق اترك النوع الأول Tabular كما هو واضغط زر Next تحصل علي الشاشة التالية كما بالشكل (14-24).



الشكل (14-24)

5- في شكل 18 تجد العنوان Choose a data source type ومعناه اختر نوع مصدر بيانات التقرير والنوع الافتراضي والتقليدي هو جملة استعلام SQL Query لذلك اترك الاختيار الافتراضي كما هو واضغط بالماوس زر Next وتعتبر هذه الشاشة جديدة عن الإصدار السابق حيث سمحت بتعدد مصادر البيانات للتقارير ، بعد اختيار SQL Query والضغط علي زر Next تحصل علي شاشة الاتصال بقاعدة بيانات كما بالشكل (25-14).



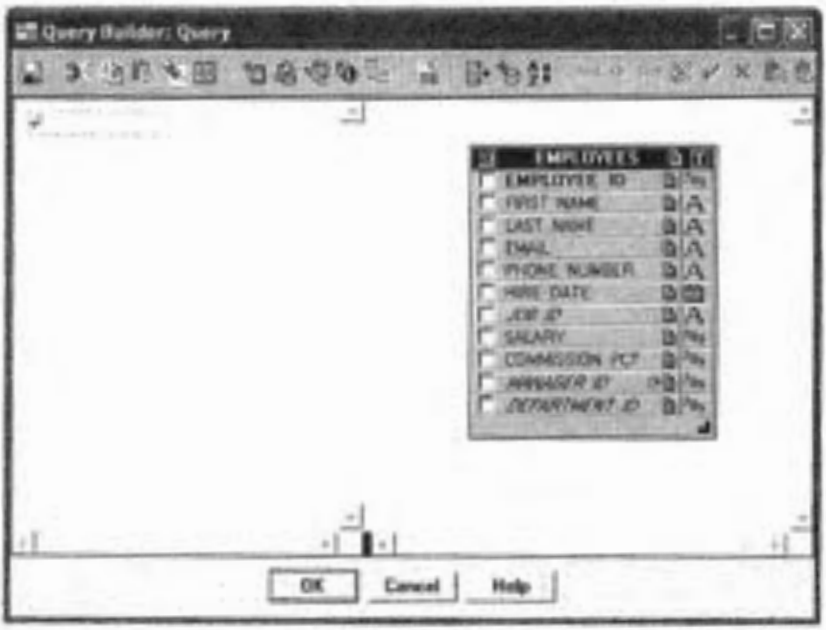
الشكل (25-14)

6- في شكل 19 اكتب اسم وكلمة سر المستخدم الذي تريد التعامل مع بياناته وإنشاء تقرير عليها ، في هذا المثال اكتب المستخدم المخصص للتدريب Scott/tiger ، وORCL هي اسم Service الذي أعددتها علي الجهاز للوصول إلي قاعدة البيانات ويمكنك إنشاء وصلة Service لديك باسم أخر.

بعد ذلك اضغط زر Connect للاتصال بقاعدة البيانات والحصول علي جدول هذا المستخدم كما بالشكل (26-14).

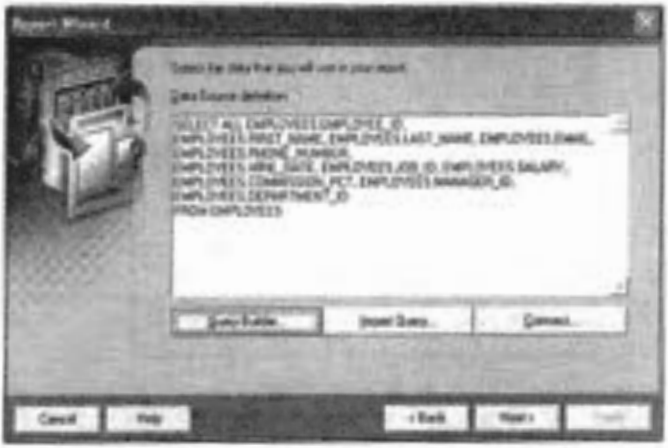
الشكل (14-26)

في شكل 20 تظهر جداول المستخدم Scott/tiger لتحديد الجداول التي تريد إنشاء استعلام عليها ، اختر الجدول EMP فقط ثم اضغط زر Include ثم close تحصل على شاشة الحقول كما بالشكل (14-27).



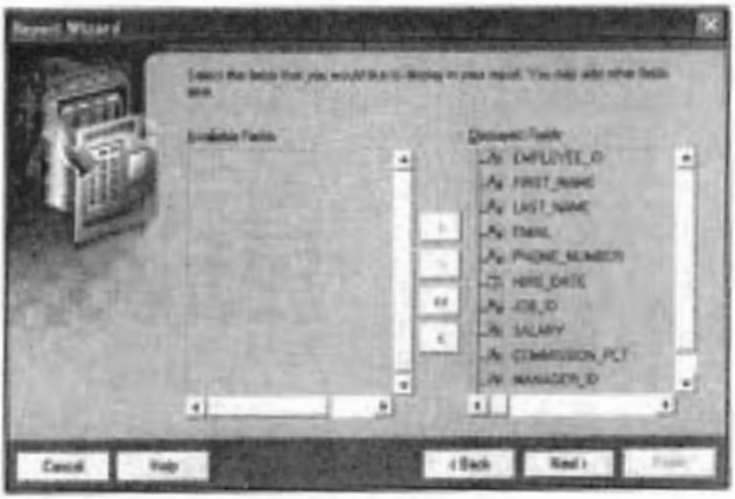
الشكل (14-27)

8- في الشكل 29-21 يتم تحديد الحقل المطلوب إضافته لاستعلام التقرير اختر جميع الحقول كما بالشكل ثم اضغط بالماوس زر OK تلاحظ كتابة الاستعلام المقابل اختيارك حقول هذا الجدول كما بالشكل (28-14).



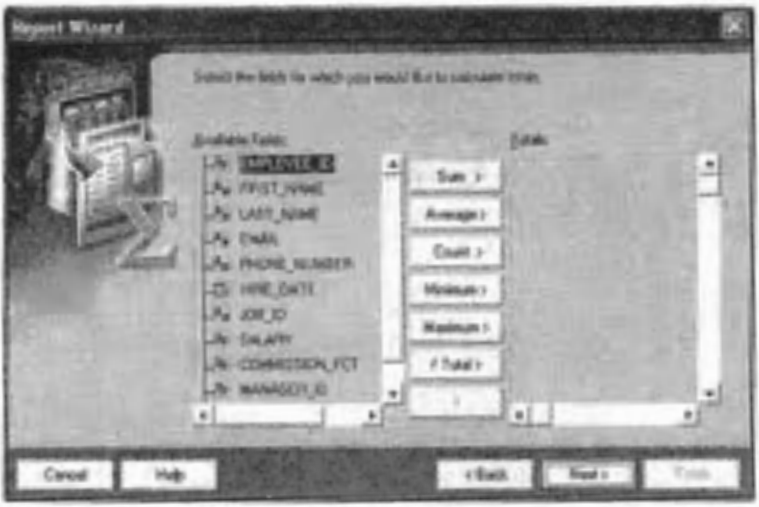
الشكل (28-14)

9- اضغط زر Next بعد إنشاء استعلام التقرير تحصل على شاشة بها حقول لاستعلام المتاحة وعليك تحديد الحقول المطلوب إظهارها بالتقرير حيث يمكنك اختيار جميع حقول الجدول في الاستعلام ولكن عند إظهارها كلها بالتقرير ، اختر جميع الحقول كما بالشكل (29-14).



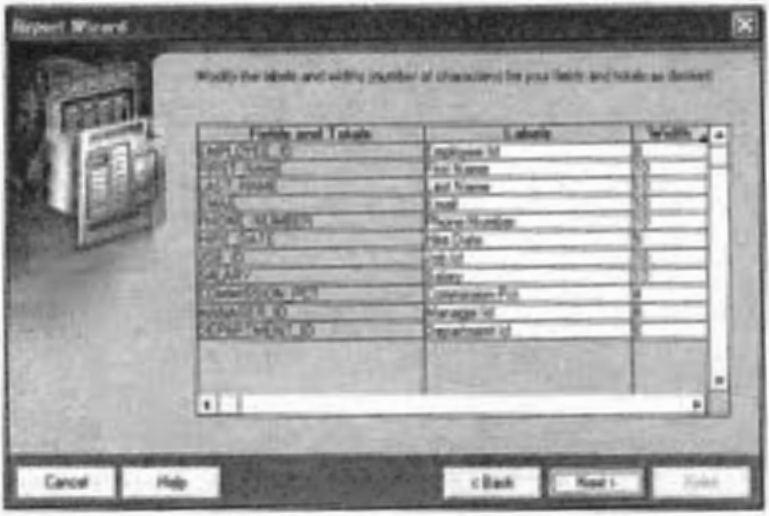
الشكل (29-14)

1- بعد اختيار جميع الحقول للظهور بالتقرير اضغط زر Next تحصل علي شاشة إنشاء حقول الملخصات Summary وهو كما شرحنا حقول تحقيق بعض الدوال الحسابية مثل Count(), AUG(), SUM() ويظهر ذلك كما الشكل (30-14).



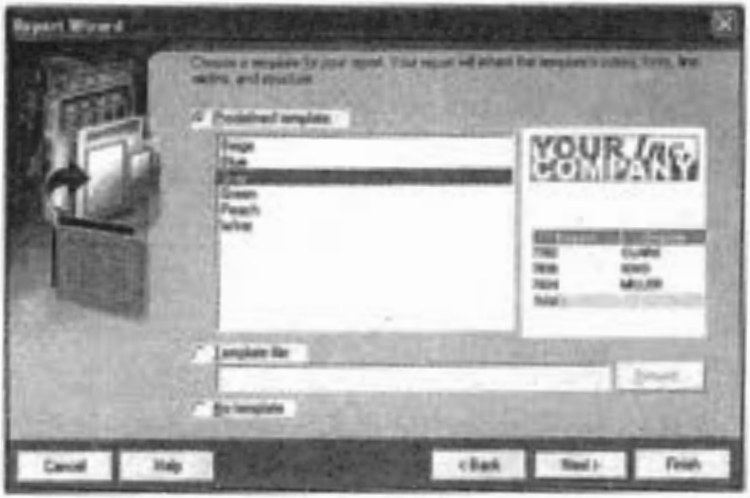
الشكل (30-14)

11- اختر حقول الملخصات كما تشاء حسب الدوال التي تريدها ثم اضغط زر Next تحصل علي شاشة تحديد عناوين الحقول Labels وكذلك مساحة عرض كل حقل علي الشاشة كما بالشكل (31-14).



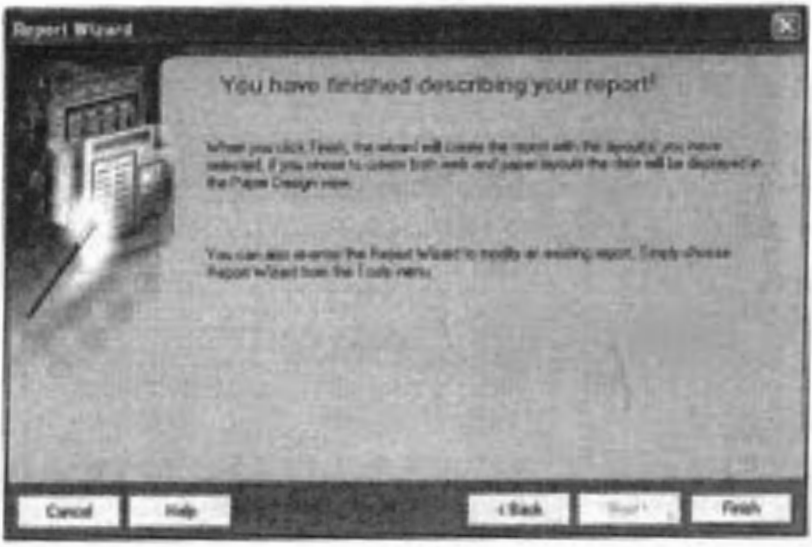
الشكل (31-14)

12- في هذا الشكل يمكنك تغيير عناوين الحقول إلي العربية أو ما يناسب التقرير وكذلك تغيير عرض width الحقول بما يناسب التقرير ، غير ما تشاء ثم اضغط زر Next لتحصل علي شاشة بأشكال التقرير المعروف مسبقا كما بالشكل (32-14).



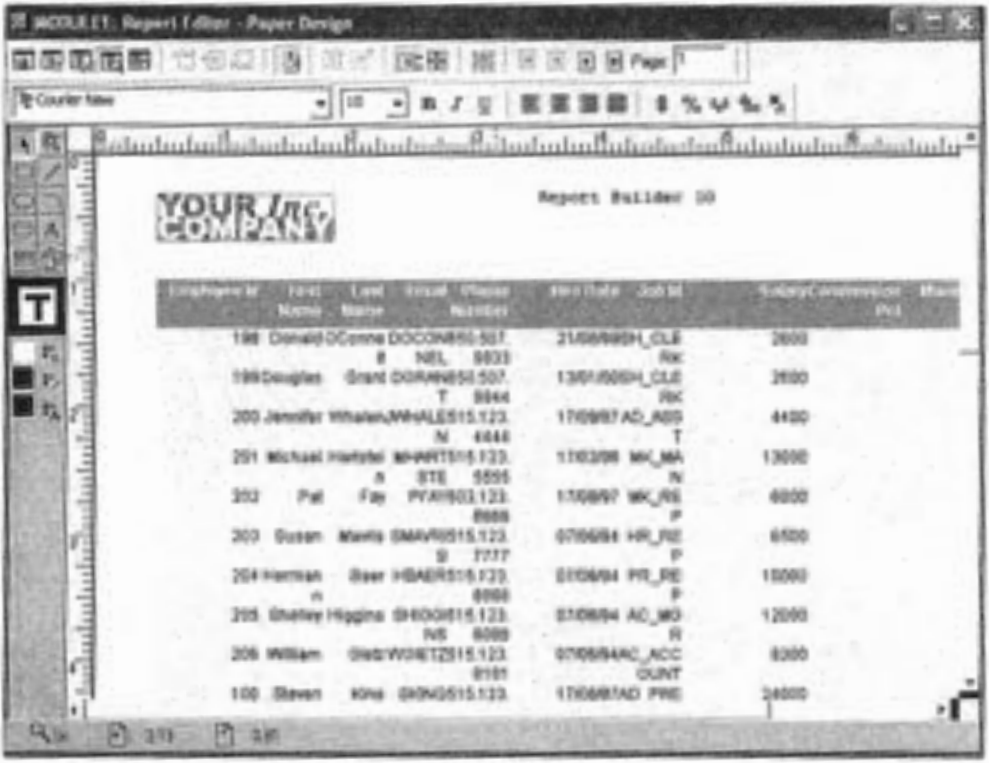
الشكل (32-14) : في هذا الشكل توجد الاختيارات الموجودة في نفس شاشة الإصدار السابقة يمكنك الرجوع إليها مع بعض اختلافات الشكل وأشكال وأسماء قوالب التقارير .

اختر الشكل الذي تريد ثم اضغط زر next تحصل علي أخر شاشة تخبرك بأنهاء الخطوات كما بالشكل (33-14).



الشكل (33-14)

اضغط زر Finish لإنهاء خطوات المعالج وعرض التقرير الذي يتم إعداده طبقا للمواصفات التي حددتها في خطوات المعالج كما بالشكل (34-14).



الشكل (34-14)

14- الشكل (34) يعرض شكل التقرير كما حددنا في المواصفات ولعرض التقرير في مستعرض صفحات الإنترنت Explore اضغط بالماوس زر العرض في بيئة عارض صفحات الانترنت تحصل علي التقرير في المستعرض قد يقابلك أسئلة بالعمل OFF LINE اختر TRY AGIN تحصل علي الشكل (35-14).

الشكل (35-14)

وهذا الشكل يعرض التقرير فى بيئة الإنترنت Browser.

بهذا تم تناول Report builder 9i,10G بشكل سريع وسوف نتناوله بتفصيل أكثر من كتاب أخر .

القسم السادس

الفصل الخامس عشر

انشاء مكتبات
Class Lib,Custom Control

Programming Concepts فى هذا الفصل

15

فى هذا الفصل نتناول موضوع من الموضوعات المتقدمة وهو انشاء مكتبات فصائل لتطبيقات النوافذ Windows Applications والتى تسمى Class Library وكذلك انشاء أدوات Tools جديدة وذلك من خلال النقاط التالية :-

- ما هى المكتبة class Lib
- انشاء تطبيق مكتبة Class Lib بسيط
- انشاء تطبيق Windows App يستخدم المكتبة Class Lib1
- انشاء تطبيق مكتبة Class Lib2 متقدم
- انشاء تطبيق Windows App يستخدم المكتبة Class Lib2
- ما هى Custom controls
- انشاء تطبيق أداة Custom Control بسيطة MyTimer
- انشاء تطبيق يستعمل الاداة الجديدة MyTimer
- انشاء تطبيق أداة Custom Control متقدمة LoginDialog
- انشاء تطبيق يستعمل الاداة الجديدة LoginDialog

ما هى المكتبة class Lib

من المفاهيم المتقدمة هو إمكانية إنشاء مكتبة فصائل class lib تحتوى على مجموعة من الفصائل classes كثيرة الاستعمال بالنسبة لك أو بالنسبة لفريق العمل ووضعها فى مكتبة تترجم لل ملف بالامتداد dll (Dynamic Link Lib) ثم استعمال هذه المكتبة من قبل أى برنامج جديد لك أو لفريق البرمجة وهذا الأسلوب يوفر الكثير حيث يوفر اعادة كتابة هذه الفصائل classes أكثر من مرة ، كما يوفر إعادة ترجمة هذه الفصائل أكثر من مرة وتصحيح الأخطاء حيث تقوم بترجمتها مرة واحدة وتصحيح الاخطاء مرة واحدة ثم استعمالها كما تشاء كما أنها توفر صورة غير نصية من المكتبة DLL بحيث يمكن استعمالها دون الاطلاع على نص الأوامر أو التعديل فيها

انشاء تطبيق مكتبة Class Lib بسيط

لتوضيح هذه الفكرة سوف نقوم بانشاء مكتبة فصائل class lib بسيطة تحتوى على فصيلة واحدة class بها دالة Method واحدة للتجربة فقط على نقوم باعداد مكتبة فصائل class Lib متقدمة فيما بعد ولتوضيح ذلك تابع الخطوات التالية:

1-قم باعداد تطبيق جديد من النوع ClassLibrary كما فى الشكل (1-15)

شكل (1-15)

2- اترك الاسم الافتراضي Classlibrary1 كما هو ثم اضغط الزر Ok تحصل على ملفات تطبيق المكتبة وبها الملف class1.vb الذى يحتوى فصيلة class بالاسم class1مفتوحة للكتابة.

3- اكتب تعريف دالة Method بسيطة لعرض رسالة تفيد انها تستدعى من مكتبة الفصائل ClassLib1 كما فى السطور الموضحة فى الشكل (2-15).

فى لغة VB.NET

```
Public Class Class1
    Public Sub Hello()
        MsgBox("Hello from Classlibrary1")
    End Sub
End Class
```

شكل (2-15)

فى لغة C#

```
1  using System;
2
3  namespace ClassLibrary2
4  {
5      /// <summary>
6      /// Summary description for Class1.
7      /// </summary>
8      public class Class1
9      {
10         public Class1()
11         {
12             //
13             // TODO: Add constructor logic here
14             //
15         }
16         public string Hello()
17         {
18     return "Hello";
19         }
20
21     }
22 }
23
```

فى هذه السطور :

- فى السطر رقم 1 تم تعريف فصيلة class بالاسم Class1.
- فى السطر رقم 2 تم تعريف برنامج فرعى من النوع Sub (Method) كدالة عضوة للفصيلة Class1.
- فى السطر رقم 3 تم استدعاء الدالة ()Msgbox وبين أقواسها رسالة تفيد انها تعرض من مكتبة الفصائل ClassLib.

4- قم بترجمة تطبيق المكتبة ClassLibrary1 وذلك باختيار ClassLibrary1 Build من الاختيار Build من القائمة الرئيسية تحصل على رسالة تفيد ترجمة التطبيق بدون أخطاء (Build: 1 succeeded, 0 failed, 0 skipped).

قبل الانتقال الى الخطوة الثانية وهى استعمال هذه المكتبة تأكد من ترجمة وإنشاء ملف ClassLibrary1.DLL وذلك بفتح المجلد MyDocuments من سطح المكتب DeskTop ثم فتح المجلد Visual Studio Projects تجد مجلد بالاسم ClassLibrary1 افتحه تجد مجلد بالاسم bin افتحه تحصل على الملف ClassLibrary1.dll كما فى الشكل (3-15).

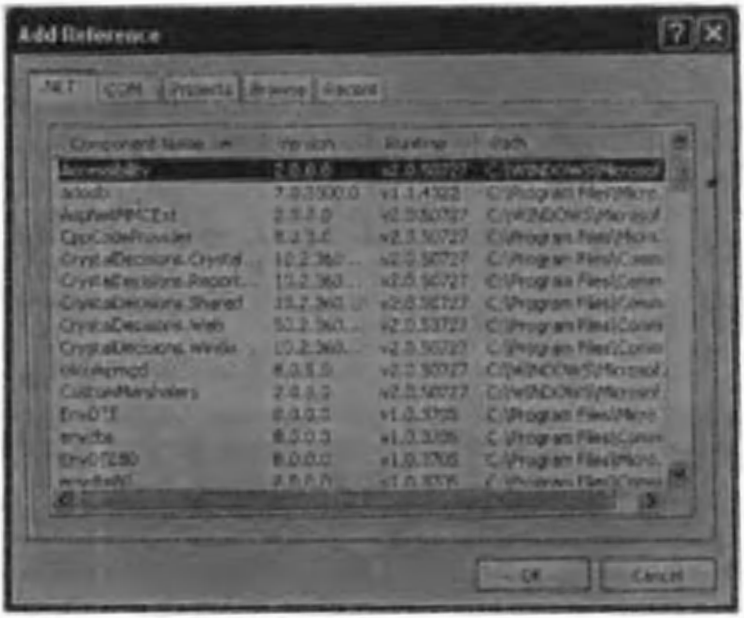
شكل (3-15)

انشاء تطبيق Windows App يستخدم المكتبة ClassLibrary1

لانشاء تطبيق يستعمل المكتبة التى تم إنشائها ClassLibrary1 اتبع الخطوات التالية:

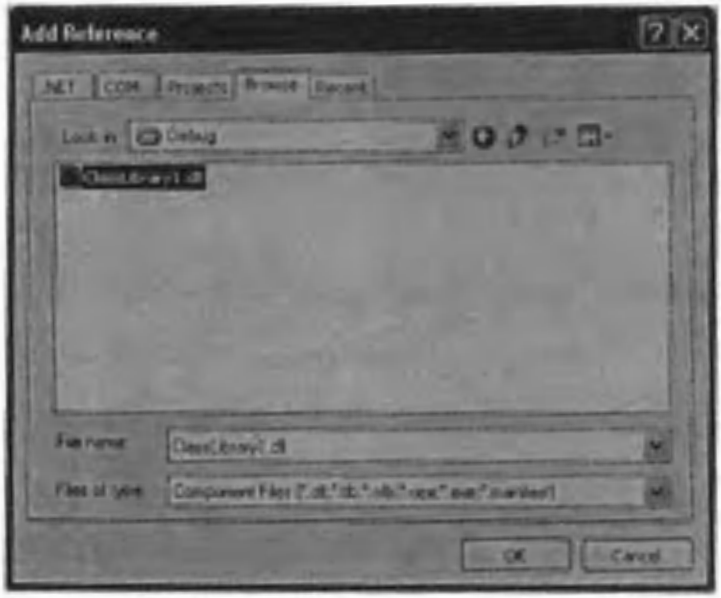
1- قم بانشاء تطبيق جديد من النوع Windows Application كما سبق.

2- اختر Project من القائمة الرئيسية ومنها اختر Add Reference يظهر مربع حوار Add Reference للاشارة الى ملف المكتبة كما فى الشكل (4-15).



شكل (15-4)

3- اضغط الصفحة Projects (Tab) ثم اضغط الزر Browse واختر الملف ClassLibrary1 كما في الشكل (15-5)



شكل (15-5)

4- اضغط الزر Open فيتم اضافة ملف المكتبة الى مجموعة العناصر المختارة Selected Components أسفل مربع الحوار ،اضغط الزر Ok تعود الى بيئة التصميم.

5- وقع زر أمر Button وغير عنوانه الى Hello واضغطه مرتين واكتب به سطور أوامر استدعاء الدالة Hello المعرفة فى الفصيلة class1 المعرفة فى مكتبة الفصائل ClassLibrary1 التى قمنا بإعدادها وقم بعمل Reference إليها وتتضح هذه السطور فى الشكل (6-15).

```
Dim obj1 As New ClassLibrary1.Class1()
obj1.Hello()
```

شكل (6-15)

فى لغة C#

```
106         private void button1_Click(object sender, System.EventArgs e)
107         {
108             ClassLibrary2.Class1 obj1=new ClassLibrary2.Class1 ();
109             textBox1.Text =obj1.Hello ();
110
111         }
```

فى هذه السطور

- فى السطر رقم 59 تم الإعلان عن المتغير Obj1 كهدف (object) من الفصيلة Class1 المعرفة داخل المكتبة ClassLibrary1.
- فى السطر رقم 60 تم استدعاء الدالة ()Hello العضوة فى الفصيلة Class1.

وبالتالى عند تنفيذ البرنامج والضغط على الزر Hello تنفذ الدالة Hello وتحصل على الرسالة كما فى الشكل (7-15).

شكل (7-15)

انشاء تطبيق مكتبة Class Lib2 متقدم

فى الفقرات السابقة قمنا بإعداد مكتبة Class Library بسيطة وكذلك تطبيق يستعمل هذه المكتبة وذلك لتوضيح كيفية انشاء مكتبة فصائل Class Library وكيفية استعمالها ولكن بالطبع لن نقوم بإعداد مكتبة لا تحتوى إلا على Class بسيط بهذا الشكل وكذلك لاتحتوى إلا على دالة تقوم بعرض رسالة ولكن الطبيعى إننا نقوم باعداد مكتبة class Library ذات قيمة ،ولتوضيح ذلك نقوم باعداد مكتبة Class Library تحتوى على فصيلتين Classes الأولى MathOperations وتوفر دوال لتحقيق العمليات الحسابية الأساسية وهى الجمع والطرح والضرب والقسمة من خلال مجموعة الدوال الأعضاء Member Methods التى تحقق هذه العمليات والثانية AdvcOperation التى توفر بعض العمليات مثل عملية الأس ()Power .

ولتحقيق ذلك تابع الخطوات التالية:

1 -قم باعداد تطبيق جديد من النوع ClassLibrary كما فى سبق.

2- اترك الاسم الافتراضى ClassLibrary2 كما هو ثم اضغط الزر Ok تحصل على ملفات تطبيق المكتبة وبها الملف class1.vb الذى يحتوى فصيلة class بالاسم class1 مفتوحة للكتابة.

3- قم بتغيير اسم الفصيلة class1 الى MathOperations ثم اكتب تعريف الدوال Methods التى تحقق العمليات البسيطة كما فى السطور الموضحة فى الشكل (8-15).

```
Public Class MathOperation
Dim No1 As Double
Dim No2 As Double
Dim Res As Double

Public Function MySum(ByVal v_No1 As Double, ByVal v_No2 As Double)
No1 = v_No1
No2 = v_No2
Res = No1 + No2
Return Res
End Function
Public Function MySub(ByVal v_No1 As Double, ByVal v_No2 As Double)
No1 = v_No1
No2 = v_No2
Res = No1 - No2
Return Res
End Function
Public Function MyMul(ByVal v_No1 As Double, ByVal v_No2 As Double)
No1 = v_No1
No2 = v_No2
Res = No1 * No2
Return Res
End Function

Public Function MyDiv(ByVal v_No1 As Double, ByVal v_No2 As Double)
No1 = v_No1
No2 = v_No2
Res = No1 / No2
Return Res
End Function

End Class
```

الشكل(8-15)

فى لغة C#

```
    public class MathOperations
    {
        double No1,No2,No3;

        public MathOperations()
        {
            // TODO: Add constructor logic here
        }
    public  double MySum(double v_No1,double v_No2)
        {
        No1=v_No1;
        No2=v_No2;
        No3=No1+No2;
return No3;
        }
public double MySub(double v_No1,double v_No2)
        {
            No1=v_No1;
            No2=v_No2;
            No3=No1-No2;
            return No3;
        }

public double MyMul(double v_No1,double v_No2)
        {
            No1=v_No1;
            No2=v_No2;
            No3=No1*No2;
            return No3;
        }
public double MyDiv(double v_No1,double v_No2)
        {
            No1=v_No1;
            No2=v_No2;
            No3=No1/No2;
            return No3;
        }
        }
```

فى هذه السطور:

- فى السطر رقم 1 تم تعريف فصيلة class بالاسم MathOperations.
- فى السطور رقم 2و3و4 تم الاعلان عن ثلاثة متغيرات من النوع Double|hx No1,No2,No3.
- من السطر رقم 5 الى السطر رقم 28 تم انشاء 4 دوال Methods أعضاء فى الفصيلة لتحقيق العمليات الحسابية (/,*,-,+).
- فى السطر رقم 5 تم انشاء الدالة ()MySum التى تقوم باستقبال معاملين (Paramters) فى المتغيرين v_No1,v_No2.

- فى السطرين رقم 6و7 تقوم الدالة بوضع قيم المعاملات فى متغيرى الفصيلة No1,No2.
- فى السطر رقم 8 ثم تقوم بجمع القيمتين ووضع النتيجة فى المتغير Res.
- فى السطر رقم 9 يتم إعادة النتيجة باستعمال الأمر Retuen مع المتغير Res.
- وبالمثل يتم إعداد الدوال MySub(),MyDiv(),MyMul().

4- قم بتعريف فصيلة Class جديدة بعد إنتهاء الفصيلة الأولى (End Class) وذلك بالاسم AdvcOperation وقم بكتابة دوال هذه الفصيلة كما هو موضح فى السطور التالية كما بالشكل (9-15).

```
Public Class AdvcOperation
Dim b As Integer
Dim p As Integer
Dim res As Integer

Public Function MyPower(ByVal v_b As Integer, ByVal v_p As Integer)
Dim i As Integer
res = 1
b = v_b
p = v_p
For i = 1 To p
res = res * b
Next
Return res
End Function

Public Function MyPower(ByVal v_factNo As Integer)
Dim i As Integer
res = 1
For i = 1 To v_factNo
res = res * i
Next
Return res
End Function
End Class
```

شكل (9-15)

فى لغة C#

```
47      public class AdvcOperation
48      {
49  public int b,p,res;
50          public int MyPower(int v_b,int v_p)
51          {
52
53              b=v_b;
54              p=v_p;
55              int i;
56              for (i=1 ;i<=p ;i++)
57          res=res*b;
58              return res;
59          }
60          public int MyFact(int v_factNo)
61          {
62              int i;
63              res=1;
64              for (i=1 ;i<=v_factNo ;i++)
65                  res=res*i;
66              return res;
67          }
68      }
```

فى هذه السطور:

- فى السطر رقم 31 تم تعريف فصيلة class بالاسم AdvcOperation.
- فى السطور رقم 32و33و34 تم الاعلان عن ثلاثة متغيرات من النوع Double.
- فى السطر رقم 35 إلى السطر رقم 42 تم انشاء دالة Methods بالاسم MyPower() كدالة عضوة فى الفصيلة لحساب قيمة رقم مرفوع لأس (Power).
- فى السطر رقم 37 تم استعمال For Loop لتكرار ضرب الاساس b فى القيمة Res التى تأخذ أول مرة القيمة 1 ثم تأخذ نتيجة الضرب فاذا كان المطلوب ايجاد قيمة الرقم 5 مرفوع للاس 4 يتم ضرب الرقم 5 أول مرة فى 1 ثم فى 5 ثم فى 25 وهكذا

حتى تنتهي قيمة الاس ثم تقوم بوضع النتيجة في المتغير Res واعادتها عند استدعاء الدالة.

- بالمثل يتم انشاء دالة أخرى بالاسم ()MulFact يمكنك مراجعة سطورها من السطر رقم 43 الى السطر رقم 50.

5- قم بترجمة تطبيق المكتبة ClassLibrary2 وذلك باختيار Build ClassLibrary2 من الاختيار Build من القائمة الرئيسية لتحصل على رسالة تفيد ترجمة التطبيق بدون أخطاء (Build: 1 succeeded, 0 failed, 0 skipped).

قبل الانتقال الى الخطوة الثانية وهي استعمال هذه المكتبة تأكد من ترجمة وإنشاء ملف ClassLibrary2.DLL وذلك بفتح المجلد MyDocuments من سطح المكتب DeskTop ثم فتح المجلد Visual Studio Projects تجد مجلد بالاسم ClassLibrary2 افتحه تجد مجلد بالاسم bin افتحه تحصل على الملف ClassLibrary1.dll كما سبق.

إنشاء تطبيق Windows App يستخدم المكتبة ClassLibrary2

لانشاء تطبيق يستعمل المكتبة التي تم انشائها ClassLibrary2 اتبع الخطوات التالية:

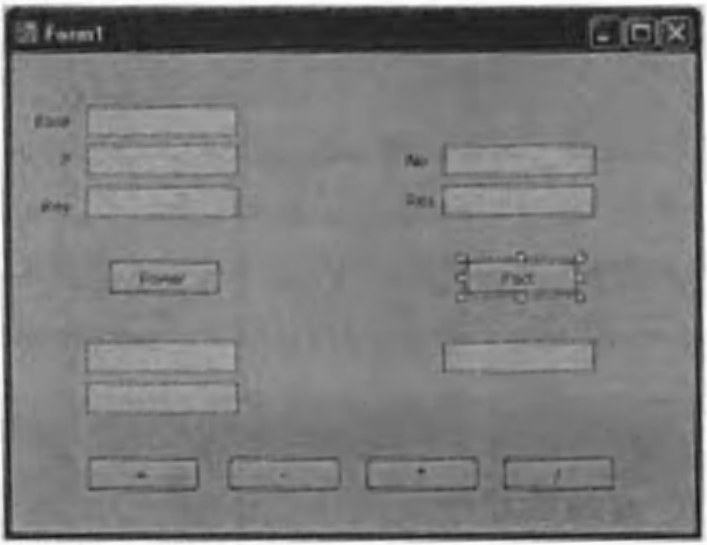
1- قم بإنشاء تطبيق جديد من النوع Windows Appliaction كما سبق.

2- اختر Project من القائمة الرئيسية ومنها اختر Add Reference يظهر مربع حوار Add Reference للاشارة الى ملف المكتبة كما سبق.

3- اضغط الصفحة Projects (Tab) ثم اضغط الزر Browse واختر الملف ClassLibrary2 كما سبق.

4- اضغط الزر Open فيتم اضافة ملف المكتبة الى مجموعة العناصر المختارة Selected Components اسفل مربع الحوار ، اضغط الزر Ok تعود الى بيئة التصميم.

5- قم بتصميم النموذج Form1 ليصبح كما في الشكل (10-15).



شكل (15-10)

6- اضغط مرتين الزر Power واكتب به سطور أوامر استدعاء الدالة MyPower() المعرفة فى الفصيلة AdvcOperation المعرفة فى مكتبة الفصائل ClassLibrary2 التى قمنا بإعدادها وقم بعمل Reference اليها وتتضح هذه السطور فى الشكل (15-11).

```
Dim AdvcOpObj As New ClassLibrary2.AdvcOperation()
Dim r As Integer
r = AdvcOpObj.MyPower(Val(TextBox3.Text), Val(TextBox4.Text))
TextBox5.Text = r
```

شكل (15-11)

فى لغة C#

```
164        private void button1_Click(object sender, System.EventArgs e)
165        {
166            int r;
167            ClassLibrary2.AdvcOperation AdvcOpObj =new ClassLibrary2.AdvcOperation ();
168            r=AdvcOpObj.MyPower  (Int32.Parse (textBox3).Int32.Parse (textBox4));
169            textBox5.Text =r+"";
170        }
```

فى هذه السطور :

- فى السطر رقم 236 تم تعريف متغير هدف (Object) بالاسم AdvcOpObj من الفصيلة AdvcOperation المعرفة بالمكتبة ClassLibrary2.

- فى السطر رقم 238 تم استدعاء الدالة (MyPower() مع متغير الهدف مع ارسال المعاملات ووضع النتيجة فى المتغير r.
- فى السطر رقم 239 تم وضع قيمة المتغير r فى مربع النص TextBox3.

7- اضغط مرتين الزر Fact واكتب به سطور أوامر استدعاء الدالة (MyFact() المعرفة فى الفصيلة AdvcOperation المعرفة فى مكتبة الفصائل ClassLibrary2 التى قمنا بإعدادها وقم بعمل Reference إليها وتتضح هذه السطور فى الشكل (12-15).

```
242        Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
243            Dim AdvcOpObj As New ClassLibrary2.AdvcOperation()
244            Dim r As Integer
245            r = MathOpObj.MyFact(Val(TextBox4.Text))
246            TextBox3.Text = r
247        End Sub
```

شكل (12-15)

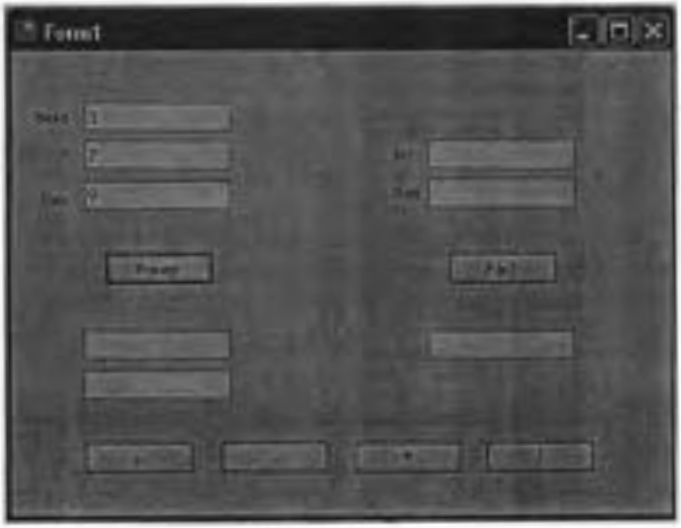
فى لغة C#

```
156            private void button1_Click(object sender, System.EventArgs e)
157            {
158                int r;
159        ClassLibrary2.AdvcOperation AdvcOpObj =new ClassLibrary2.AdvcOperation ();
160        r=AdvcOpObj.MyFact (Int32.Parse (textBox4));
161        textBox3.Text =r+"";
162            }
```

فى هذه السطور:

- فى السطر رقم 242 تم تعريف متغير هدف (Object) بالاسم AdvcOpObj من الفصيلة AdvcOperation المعرفة بالمكتبة ClassLibrary2.
- فى السطر رقم 245 تم استدعاء الدالة (MyFact() مع متغير الهدف مع ارسال المعاملات ووضع النتيجة فى المتغير r.
- فى السطر رقم 246 تم وضع قيمة المتغير r فى مربع النص TextBox3.

8- نفذ البرنامج عند هذه المرحلة وجرب كتابة رقم كأساس ورقم كأس ثم اضغط الزر Power ولاحظ النتيجة وجرب كتابة رقم فى خانة No ثم اضغط الزر Fact ولاحظ النتيجة كما فى الشكل (13-15).



شكل (13-15)

9- قم بكتابة سطور أوامر الأزرار +،-،*،/ كما هو موضح فى السطور التالية كما بالشكل (14-15).

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        Dim mathop As New ClassLibrary2.MathOperation()
        Dim r As Double
        r = mathop.MySum(Val(TextBox6.Text), Val(TextBox7.Text))
        TextBox8.Text = r
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        Dim mathop As New ClassLibrary2.MathOperation()
        Dim r As Double
        r = mathop.MySub(Val(TextBox6.Text), Val(TextBox7.Text))
        TextBox8.Text = r
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
        Dim mathop As New ClassLibrary2.MathOperation()
        Dim r As Double
        r = mathop.MyMul(Val(TextBox6.Text), Val(TextBox7.Text))
        TextBox8.Text = r
    End Sub
```

```
Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
    Dim mathop As New ClassLibrary2.MathOperation()
    Dim r As Double
    r = mathop.MyDiv(Val(TextBox6.Text), Val(TextBox7.Text))
    TextBox8.Text = r
End Sub
```

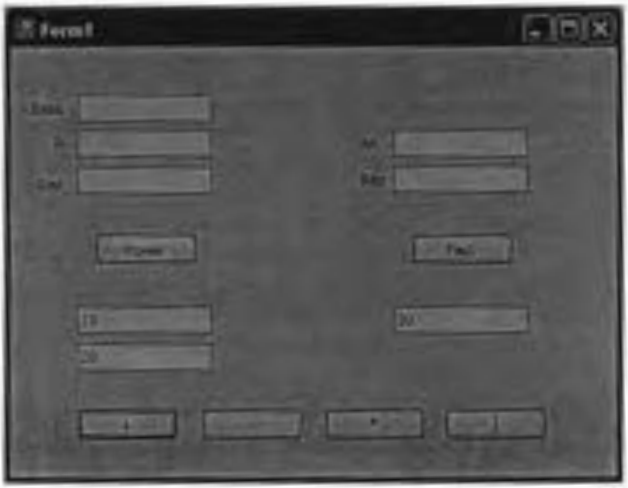
شكل (14-15)

فى لغة C#

```
private void button3_Click(object sender, System.EventArgs e)
{
ClassLibrary3.MathOperations mo=new ClassLibrary3.MathOperations ();
    double r;
    r=mo.MySum (double.Parse (textBox6),double.Parse (textBox7));
    textBox8.text=r;
}
private void button4_Click(object sender, System.EventArgs e)
{
ClassLibrary3.MathOperations mo=new ClassLibrary3.MathOperations ();
    double r;
    r=mo.MySub (double.Parse (textBox6),double.Parse (textBox7));
    textBox8.text=r;
}
private void button5_Click(object sender, System.EventArgs e)
{
ClassLibrary3.MathOperations mo=new ClassLibrary3.MathOperations ();
    double r;
    r=mo.MyMul (double.Parse (textBox6),double.Parse (textBox7));
    textBox8.text=r;
}
private void button6_Click(object sender, System.EventArgs e)
{
ClassLibrary6.MathOperations mo=new ClassLibrary3.MathOperations ();
    double r;
    r=mo.MyDiv (double.Parse (textBox6),double.Parse (textBox7));
    textBox8.text=r;
}
```

حاول فهم شرح هذه السطور من شرح السطور السابقة.

10- نفذ البرنامج وجرب الأوامر الجديدة تحصل على النتيجة كما فى الشكل (15-15).



شكل (15-15)

ما هى Custom controls

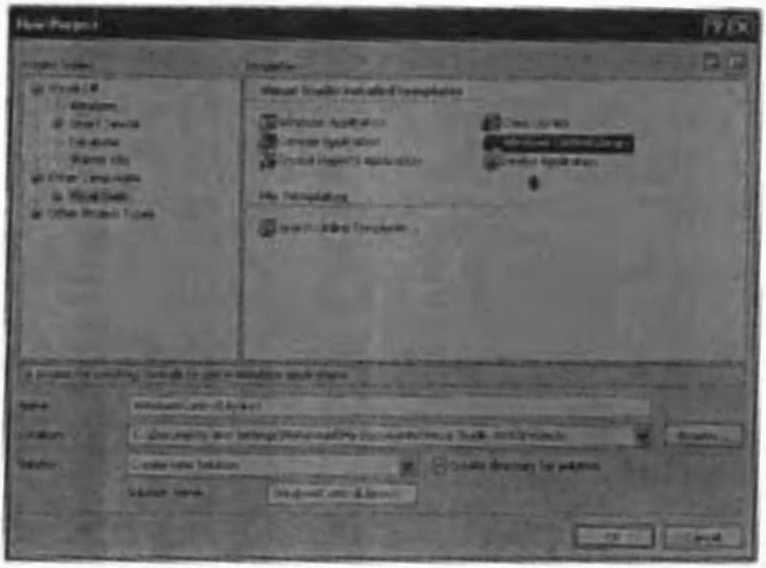
هى فكرة إنشاء أدوات Tools (ويطلق عليها Controls) مثل الأدوات الموجودة بمربع الأدوات ToolBox ولكن تكون خاصة بنا وتحقق عمليات ووظائف غير الأدوات المعرفة باللغة ، والأدوات المعرفة باللغة تسمى Built In Controls لإنها مبنية باللغة وليس لها دخل بها إلا الاستعمال.

بينما الأدوات التى نحن بصددها يقوم مستخدم اللغة (المبرمج) بإنشائها لذلك تسمى User defind Controls أو Custom controls أو WindowsControlLibrary ويمكنك أن تجد الآلف الكثيرة من هذه الأدوات التى يقوم المبرمجين بإعدادها ونشرها على شبكة الإنترنت إذا بحثت عن الامتداد OCX وهى مفيدة جداً حيث إنها عبارة عن برامج كاملة فى صورة أدوات بسيطة تضاف إلى شريط الأدوات ، ولتوضيح أهميتها تابع معنا الفقرات التالية:

انشاء تطبيق أداة Custom Control بسيطة MyTimer

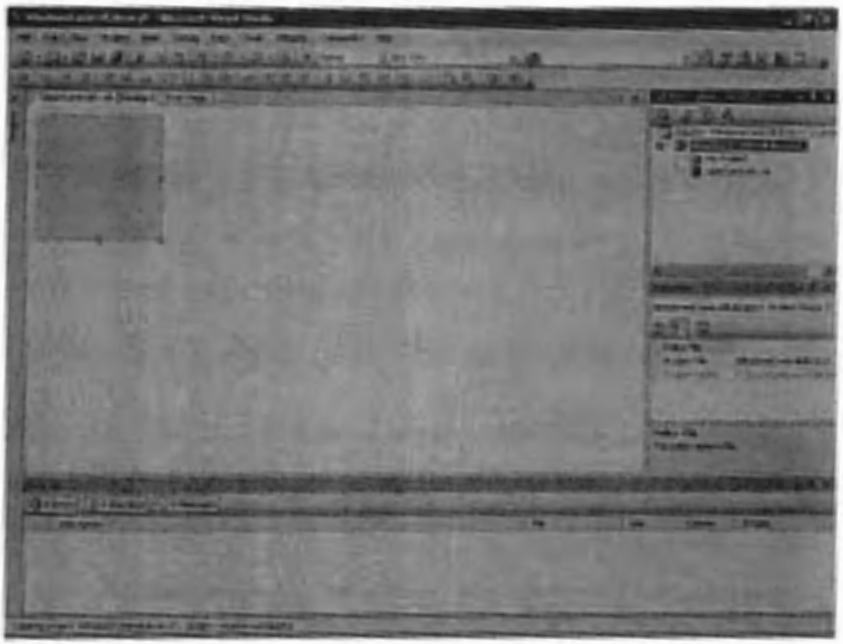
الخطوة الأولى هى شرح بالخطوات كيفية انشاء أداة Control بسيطة تقوم بعرض الوقت والتاريخ الحالى تلقائيا ولتحقيق ذلك تابع الخطوات التالية:

1- قم بانشاء تطبيق جديد من النوع WindowsControlLibrary وغير اسمه الى MyTimer كما فى الشكل (16-15).



شكل (15-16)

2-اضغط الزر Ok تدخل الى بيئة التصميم وتحصل على نموذج Form صغير وبسيط يشابه النموذج Form الخاص بتطبيق Windows Application كما في الشكل (15-17).



شكل (15-17)

3- قم بتوقيع أداة Label وأداة Timer واضبط خاصية Enabled لأداة الـ Timer1 على القيمة True والخاصية Interval على القيمة 1000.

4- اضغط أداة الـ Timer بالماوس مرتين واكتب بها أوامر عرض الوقت والتاريخ كما هو موضح في الشكل (18-15).

```
Public Class MyTimer
    Inherits System.Windows.Forms.UserControl

 Windows Form Designer generated code

    Private Sub Timer1_Tick(ByVal sender As System.Object
        Label1.Text = Now
    End Sub
End Class
```

شكل (18-15)

في لغة #C

```
        private void timer1_Tick(object sender, System.EventArgs e)
        {
            label1.Text=(Int32.Parse(label1.Text)+1)+"";

        }
    }
```

مع ملاحظة تغيير اسم الفصيلة إلى MyTimer.

5- قم بترجمة تطبيق الأداة وذلك باختيار Build MyTimer من الاختيار Build من القائمة الرئيسية وتأكد من عدم وجود أخطاء.

6- اذهب إلى مجلد التطبيق MyTimer وتأكد من وجود الملف MyTimer.dll داخل الفهرس bin داخل الفهرس MyTimer داخل الفهرس Visual Studio Projects داخل الفهرس My Documents كما في الشكل (19-15).

شكل (15-19)

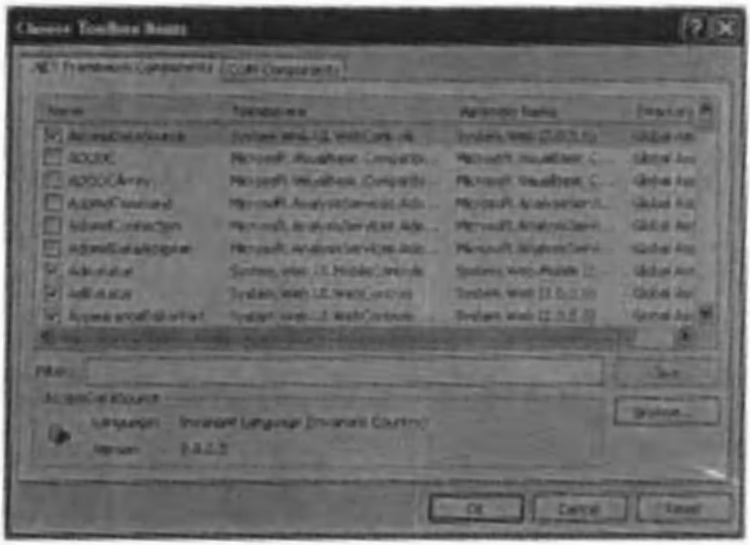
بهذا تم إعداد الأداة ويبقى استعمالها.

انشاء تطبيق يستعمل الاداة الجديدة MyTimer

هذه هى الخطوة الثانية فى التعامل مع الأدوات Windows Control Library وهى إعداد التطبيق الذى يستعمل هذه الأداة ويتم ذلك كما فى الخطوات التالية:

1- قم بإنشاء تطبيق من النوع Windows Application كما سبق.

2- اضغط بالطرف الأيمن للماوس فوق مربع الأدوات ToolBox تحصل على قائمة اختيارات Menu اختر منها Custom ToolBox يظهر مربع حوار Customize ToolBox كما بالشكل (20-15).



شكل (15-20)

3- اضغط الصفحة (tab) .NET Framework Components. ثم اضغط الزر Browse لتحميل ملف الأداة MyTimer.Dll الذى انشأناه فى الفقرة السابقة.

4- يظهر مربع حوار فتح ملف حدد مجلد الملف MyTimer وحدد الملف MyTimer.DLL كما فى الشكل (21-15).

شكل (21-15)

5- اضغط الزر Open يتم اضافة الأداة الى مجموعة الأدوات المختارة والعودة إلى مربع حوار Cusomize ToolBox.

6- اضغط الزر Ok تعود إلى بيئة التصميم وقد تم إضافة الأداة الجديدة MyTimer فى أخر مربع الأدوات ToolBox.

7- قم بسحب وتوقيع الأداة الجديدة MyTimer على النموذج Form لحصل على الأداة التى صممناها وتراها تعمل تلقائيا فى بيئة التصميم وتعرض الوقت والتاريخ كما فى الشكل (22-15).

شكل (22-15)

8- نفذ البرنامج تلاحظ أن الأداة تعمل كما كانت تعمل فى بيئة التصميم بهذا تم إنشاء الأداة مرة واحدة ثم استعمالها أكثر من مرة.

فى لغة Java

المكتبات Packages

هى فكرة ممتازة وقوية فى لغات البرمجة.

وهى فكرة بناء مكتبات من الفصائل classes ، حيث يتم وضع عدد ضخم من الفصائل classes ولذلك فإننا نضع كل مجموعة من الفصائل classes المتشابهة مع بعضها البعض في حزمة واحدة.

- إذن المكتبات Packages هى عبارة عن مجلدات (Directories) تنظم الفصائل classes بحيث أن كل مجموعة فصائل classes لها وظائف مشتركة فى لغة الجافا Java يتم وضعها فى حزمة Package معينة.
- وقد تعاملنا فيما مضى مع المكتبات Packages الخاصة بلغة الجافا Java مثل:

```
import javax.swing.*;
```

وفيها المكتبات package المسماه (javax.swing) هي حزمة تحتوي على الفصائل classes الخاصة بأدوات الـ swing وبالمثل باقي المكتبات packages التي تعاملنا معها من قبل.

وهذا الأسلوب لا تخلو منه لغة برمجة وبدونه تصبح جميع الفصائل مع بعضها بدون تنظيم ولا تقسيم ، ولكن عمليا يتم وضع جميع الفصائل التى تؤدى وظائف متشابهة معا ، مثل فصائل الرسم Graphics والأدوات Components كما أشرنا .

ولكن هذه المكتبات تأتى مبنية فى لغة الجافا فكيف نستطيع بناء حزم خاصة بنا ولماذا؟ نريد أن ننشئ المكتبات Packages الخاصة بنا للأسباب الآتية:

- عندما تنشئ نظاماً معقداً فيفضل وضع مجموعة الوظائف الخاصة بجزء معين من النظام فى حزمة package خاصه بها فمثلاً قد يكون النظام يتعامل مع شبكات أو رسومات أو قاعدة بيانات أو واجهة مستخدم ولذلك يفضل إنشاء حزمة package لكل وظيفة على حدة بأن تضع حزمة package للشبكات و حزمة package للرسومات وهكذا.
- من الممكن إعادة استخدام المكتبات Packages التى تنشأها فى برامج أخرى فإذا كانت هناك حزمة Package أنشأتها توفر وظيفة عامة فستطيع إعادة استخدامها مباشرة.

مثال آخر

```
import java.util.*;
```

فى هذا المثال يتم استيراد (الاشارة الى) جميع فصائل classes (*) المكتبة util الموجودة فى الفهرس java وبالتالى يمكن استعمال الفصائل classes الموجودة بها كما فى السطر.

```
Date today = new Date();
```

فى هذا السطر يتم تعريف متغير هدف بالاسم today من نوع الفصيلة Date وبالتالي اذا لم يتم استيراد المكتبة التى تحتوى على الفصيلة Date لا يتعرف المترجم عليها ويعطى خطأ.

ويمكن استيراد (الإشارة إلى) فصيلة محددة class (اسم الفصيلة) موجودة بالمكتبة كما فى السطر.

```
import java.util.Date;
```

فى هذا السطر تم تحديد استيراد الفصيلة Date من المكتبة java.util.

مع ملاحظة أن الإشارة إلى جميع فصائل المكتبة لا يزيد من حجم البرنامج.

ملحوظة:

لايمكن استيراد أو الإشارة إلى جميع مكتبات فهرس محدد فمثلا من غير المقبول كتابة السطر .

```
 import java.*
or
 import java.*.*
```

للإشارة إلى جميع مكتبات الفهرس java.

ملحوظة:

يجب مراعاة عدم الاشارة الى مكتبات Packages تحتوى على فصائل classes بنفس الاسم والا سيحدث تداخل Conflict بينهما كما فى السطور التالية:

```
import java.util.*;
import java.sql.*;
```

المشكلة فى هذه السطور هو احتواء المكتبتين على فصيلة بالاسم Date مما يحدث تداخل.

ولو افترضنا انك تريد استعمال كلا الفصيلتين Date من كلا المكتبتين يتم الاشارة اليهما صراحة كما فى السطور التالية:

```
java.util.Date deadline = new java.util.Date();
java.sql.Date today = new java.sql.Date(...);
```

إنشاء المكتبة Creating Packages

لنفترض الآن أننا سننشىء فصيلة class بها دالة اسمها printData() مهمتها طباعة رسالة على الشاشة. ونريد أن نستخدم هذه الدالة فى فصيلة class أخري.

الخطوات

1-قم بنشاء تطبيق جديد واكتب فيه السطور التالية :

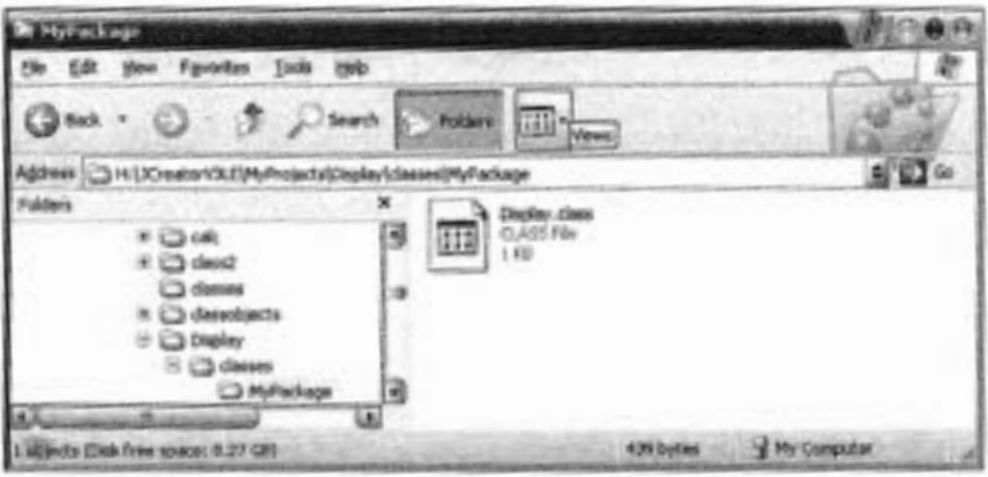
```
1  package MyPackage;
2  public class Display {
3
4      public void printData ()
5      {
6  System.out.println("This is class Display");
7
8      }
9  }
10
```

فى هذه السطور:

فى السطر رقم 1 يتم انشاء مجلداً Directory اسمه MyPackage. هذا المجلد سينشأ فى نفس مسار الملف Display.Java أى D:\Pack .

1. لكى تستطيع استخدام الفصيلة class المسماه Display واستدعاءها من حزمة package أخرى لابد أن يكون تعريفها public لكى تستطيع استخدامها.
2. فى السطر 5 يتم تعريف الدالة PrintData() التى تطبع رسالة على الشاشة.

وبعد كتابة البرنامج بدون أخطاء وترجمته من داخل البرنامج JCreator يتم انشاء المكتبة MyPackage كما فى الشكل (23-15).



شكل (32-15)

فى هذا الشكل تلاحظ انشاء فهرس بالاسم MyPacjage وبداخله الفصيلة Display.class كما أشرنا مع ملاحظة أهمية تحديد المسارات.

ولعمل ترجمة Compile لهذا البرنامج من خلال شاشة الـ Dos افتح شاشة الـ Dos حول المسار إلى المسار الفرعى D:\Pack وهو مسار الملف Display.Java.

قم بترجمة البرنامج لينشئ المكتبة package التى سميناها MyPackage ويضع الفصيلة class الناتجه فى هذا المجلد D:\Pack\MyPackage ولتنفيذ ذلك نكتب الأمر التالى.

```
javac -d . Display.java
```

الاختيار d- ينشىء المجلدات (directories) التى نريد أن ننشئها لاحظ وجود مسافة

بعد d> ثم ضع نقطة ثم اترك مسافة ثم اكتب Display.java. النقطة هنا ضرورية لأنها تشير إلى المسار الحالى وهو D:\Pack لأنك تريد أن تنشىء MyPackage فى هذا المسار.

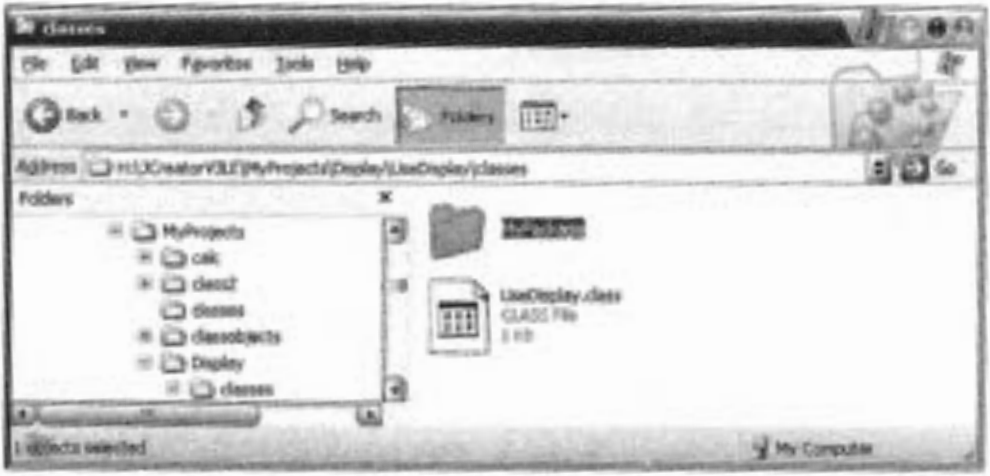
3. فى المسار D:\Pack ستلاحظ أنه تم إنشاء المجلد MyPackage قم بفتح المجلد ستلاحظ وجود Display.class فى هذا المجلد.
4. قم بالرجوع إلى المسار D:\Pack
5. فى المسار D:\Pack قم بإنشاء ملفاً جديداً باسم UseDisplay.java والذي يستخدم المكتبة package التي أنشأناها في البرنامج السابق. لاحظ أن هذا الملف لابد أن يكون فى نفس مسار المجلد MyPackage.

استعمال المكتبات

6. قم بانشاء تطبيق جديد لاستعمال المكتبة MyPackage التى تم إنشائها فى الخطوة السابقة وذلك كما فى السطور التالية:

```
1  import MyPackage.Display;
2  public class UseDisplay {
3
4      public static void main(String[] args) {
5        Display d = new Display ();
6            d.printData ();
7
8      }
9  }
```

مع ملاحظة مسار المكتبة MyPackeg ، ولتنفيذ هذا البرنامج تم وضع المكتبة فى نفس مسار البرنامج كما فى الشكل (24-15).



شكل (24-15)

في سطور البرنامج:

- في الملف UseDisplay.java بدأ البرنامج بتضمين الفصيلة class المسماة Display الموجوده في المجلد MyPackage.
- في السطر 5 يتم إنشاء هدف object من نوع الفصيلة class المسماة Display. لاحظ أن الفصيلة Display تعريفها موجود في المكتبة MyPackage.
- في السطر 6 يتم استدعاء الدالة ()printData التي هي أحد دوال الفصيلة class المسماة Display.
- ترجم Compile هذا الملف بالطريقة العادية.

```
D:\Pack>javac UseDisplay.java
```

- قم بتنفيذ البرنامج عن طريق D:\Pack>java UseDisplay.

لاحظ طباعة الرسالة عن طريق الدالة ()printData.

أو قم بتنفيذ البرنامج من خلال أحد البرامج المخصصة لذلك كما أشرنا وكما تم من قبل .

النوع Static Imports

من المزايا التي اضيفت ابتداء من الإصدار JDK 5.0, إمكانية اضافة الكلمة static بعد الأمر import مما يوفر في استدعاء الدوال وذلك كما في السطور التالية:

```
1. import static java.lang.System.*;
2. class x
3. {
4. public staic void main String args[pl)
5. {
6. out.println("Goodbye, World!"); // i.e., System.out
7. exit(0); // i.e., System.exit
8. }
```

في هذه السطور:

في السطر رقم 1 تم إضافة الكلمة static بعد كلمة import مما سهل استعمال دوال الفصيلة بدون الاشارة الى المكتبة كما في السطر رقم 6.

وتتضح فائدة ذلك عند التعامل مع بعض الدوال التي لا تحتاج أسماء الفصائل والمكتبات كما في السطور التالية:

```
sqrt(pow(x, 2) + pow(y, 2))
```

وذلك بعد بعد استعمال الـ static import للفصيلة Math

وبالتالي يكون ذلك اوضح وأبسط من السطر التالي:

```
Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2))
```

وكذلك بالنسبة للمتغيرات المعرفة (الثوابت) كما في السطر.

```
if (d.get(DAY_OF_WEEK) == MONDAY)
```

بدلا من

```
If (d.get(Calendar.DAY_OF_WEEK) == Calendar.MONDAY
```

في Oracle

ما هى نظرية REUSE إعادة الاستخدام

هي أحد أهم خصائص البرمجة والتطوير SOFTWARE ENGINEERING وهى تعنى انشاء العناصر objects مرة واحدة واستخدامها أكثر من مرة مما يوفر

1- وقت التطوير.

2- توحيد قواعد التطوير.

3- توحيد شكل التطبيق.

4- سرعة التطوير.

ويتم تحقيق هذه النظرية فى جميع عناصر البرمجة مثل:

1- العناصر OBJECTS

وذلك بإنشاء:

1-1- مكتبة عناصر (أدوات) OBJECT LIBRARIES.

2-1-مجموعة عناصر OBJECT GROUP.

2-الخصائص PROPERTIES

وذلك بإنشاء :

1-2-مجموعة الخصائص المرئية VISUAL ATTRIBUATES.

2-2-مجموعة الخصائص PROPERTY CLASS.

3- أكواد PL/SQL

وذلك بإنشاء :

1-3- البرامج الفرعية والدوال PROCEDURES & FUNCTIONS.

2-3- مكتبات حزم الدوال والبرامج الفرعية PACKAGES.

3-3- مكتبات الدوال والبرامج الفرعية PL/SQL LIBRARIES.

وفيما يلى خطوات تحقيق هذه الطرق.

1- بالنسبة لأوامر البرمجة PL/SQL

ما هي حزم الدوال والبرامج الفرعية **PACKAGE**

هي أحد الطرق المتاحة لتجميع أوامر PL/SQL في لغة ORACLE تحت اسم يعبر عن الغرض من هذه الأوامر وقد تناولناها في فصول الـ PL/SQL.

دائما يسعى التطوير في طرق البرمجة إلى النظام واختصار أوامر البرامج لتسهيل إعادة استخدامها.

وكانت أول هذه الطرق البرامج الفرعية PROCEDURES والدوال FUNCTIONS التي تناولناها في أول هذا الكتاب الأول فبدلا من كتابة أوامر PL/SQL كل مرة في أكثر من موضع وعند الاحتياج إليها يعاد كتابتها ظهرت فكرة الدوال والبرامج الفرعية التي تقوم بتجميع مجموعة الأوامر وإنشاء دالة FUNCTION أو PROCEDURE تؤدي غرض معين وعند الاحتياج لهذا الغرض يتم استدعاء الدالة FUNCTION وليس هناك الحاجة لإعادة كتابة سطورها مرة أخرى وهذه هي أول خطوة حيث يتم الإنشاء مرة واحدة والاستدعاء أكثر من مرة.

وجاءت فكرة ال PACKAGE لتزيد الأمر تنظيما فبدلا من إنشاء كثير من الدوال FUNCTION والبرامج الفرعية PROCEDURES المتناثرة وغير المنتظمة يتم إنشاء PACKAGE (حزمة) لغرض معين وليكن الرسائل MESSAGES مثلا ثم تجميع (إنشاء) مجموعة من دوال (FUNCTIONS) و PROCEDURES الخاصة بإرسال الرسائل MESSAGES داخل هذه الحزمة PACKAGE.

وبهذا الأسلوب يصبح لديك مكتبة من الحزم PACKAGES وليس الدوال فمثلا :

- حزمة الرسائل MESSAGE PACKAGE .
- حزمة مراجعة البيانات VALIDATION PACKAGE .
- حزمة التعامل مع قوائم الاختيارات MENU PACKAGE .

وبهذا أصبح الأمر أكثر نظاما .

مزايا استعمال PACKAGES :

مما تناولناه في الفقرة السابقة تجد التعامل مع PACKAGE يتميز بما يلي :

1- تجميع الدوال FUNCTIONS و البرامج الفرعية PROCEDURES التي تخص موضوع واحد في حزمة واحدة PACKAGE.

2- يتم استعمال الدوال والبرامج الفرعية بفتح PACKAGE الخاصة بهم فقط ولا تحتاج لفتح كل دالة أو برنامج فرعي علي حدة .

3- يتم الإشارة إلى اسم الحزمة PACKAGE عند استدعاء دالة FUNCTION أو PROCEDURES داخلها بالشكل التالي :

```
PACKAGE . FUNCTION ( )
```

وهنا يعطى شيء من التنظيم .

4- إنشاء واستعمال PACKAGE يمكنك من إخفاء نصوص أوامر PL/SQL الخاصة بالدوال و البرامج الفرعية حيث يمكنك ترجمتها واستعمالها دون الرجوع لنص البرامج .

هذه بعض المزايا وكما ذكرنا استعمال PACKAGE يعتبر أول الطرق بعد إعادة كتابة الأوامر أكثر من مرة . ولكن مازال هناك الكثير من هذه الطرق مثل :

أجزاء حزمة الدوال PACKAGE PARTS :

قبل الاستمرار وشرح كيفية إنشاء PACKAGE نود أن نوضح أولا أجزائها ،وهي تتكون من الأجزاء التالية :

1. قسم التعريفات PACKAGE SPECIFICATION:

في هذا الجزء يتم الإعلان عن المتغيرات العامة . وكذلك الإعلان عن الدوال FUNCTIONS والبرامج الفرعية PROCEDURES ، والمقصود بالإعلان عنها أي كتابة تعريف أي اسم الدالة ومعاملاتها وأنواعها وكذلك القيمة المرتجعة RETURN VALVE بالنسبة للدوال ولا يتم كتابة سطور أوامر PL/SQL .

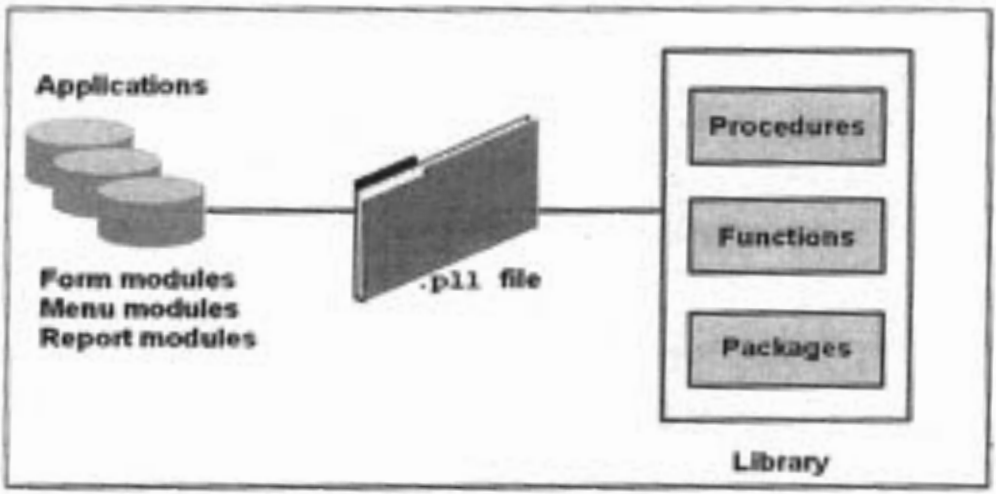
2. قسم الأوامر PACKAGE BODY:

يسمي هذا القسم بجسم الحزمة PACKAGE BODY لأنة يحتوى علي سطور أوامر PL/SQL التي تتكون منها الدوال والبرامج الفرعية فالقسم الأول PACKAGE SPECIFICATION ما هو إلا إعلان وتعريف لأسماء الدوال ولكن هذا القسم هو الذي يحتوي على سطور الأوامر التي تحقق أعمال الدوال وبدونها تصبح الدوال بالتالي حزمة البرامج PACKAGE

ليس لها قيمة فتخيل لو أنك كتبت عنوان دالة تقوم بحساب الضريبة المستحقة ولكنك لم تقم بكتابة الأوامر التي تقوم بحساب هذه الضريبة فهذا ليس له معنى ،وهذا ما يحدث بالضبط بالنسبة لعنوان الحزمة PACKAGE SPECIFICATION وجسم الحزمة PACKAGE BODY وإذا أردت المزيد يمكنك الرجوع الى فصول كتاب الـ PL/SQL.

تركيب المكتبة

ويظهر ذلك كما في الشكل (25-15).



شكل (25-15)

خطوات انشاء PL/SQL LIB

1- قم بعرض النافذة Object Navigator وحدد الاختيار PL/SQL Lib.

2- اضغط الزر + لانشاء مكتبة جديدة ثم قم بتغيير الاسم.

3- قم بكتابة دوال Functions وإجراءات Procedures.

إنشاء عناصر لإعادة الاستخدام DESIGN FOR REUSE

ذكرنا أنه من النقاط المتطورة في عالم التطوير والبرمجة هو إنشاء (إعداد) المكتبات بأشكالها المختلفة من مكتبات دوال PLL ومكتبات عناصر OBJECT LIBRARIES والكثير من الأشكال وهي الخطوة المتقدمة التي يسعى إليها أي شخص مطور أو مكتب أو شركة تطوير وتوجد أكثر.

من صورة لذلك منها ما يلي :-

1- القوالب . TEMPLATES

- قوالب النماذج FORMS TEMPLATES.
- قوالب التقارير REPORTS TEMPLATES.
- قوالب الرسومات GRAPHICS TEMPLATES.

2- الاستفسارات الخارجية EXTERNAL QUERIES.

3- عناصر من النوع VISUAL ATTRIBUTES.

4- عناصر PROPERTY CLASS.

5- عناصر SUB CLASSING &COPYING &READER.

6- مكتبة العناصر OBJECT LIBRARIES &SMART CLASS.

7- مجموعة العناصر OBJECT GROUPS.

8- المعاملات PARAMETERS.

9- قوائم المعاملات PARAMETERS LISTS.

مكتبة العناصر OBJECT LIBIRAY

بالإضافة لفكرة حزم الدوال والبرامج الفرعية PACKAGERS التي شرحناها توجد فكرة أخري تسهل علي المطور DEVELOPER إعادة البناء كل مرة في البداية وهي مكتبة العناصر أو OBJECT LIBRARIES وفكرتها إنشاء مكتبة من العناصر (OBJECTS) الخاصة بالتطوير مع إعطائها الخصائص المطلوبة ثم إعادة استعمالها وقت الحاجة .

ويتم ذلك بتجهيز العناصر وإعطائها الخصائص المطلوبة ثم إضافتها للمكتبة .

ولتوضيح هذا الموضوع نشرح النقاط التالية :

- الخطوة الأولى : إنشاء مكتبة عناصر CREATE OBJECT LIBRARIES.
- الخطوة الثانية : إنشاء عناصر وإعطائها خصائص .
- الخطوة الثالثة : إضافة عناصر لمكتبة العناصر وجعلها من النوع SMART CLASS.
- الخطوة الرابعة : إنشاء نموذج جديد واستعمال مكتبة العناصر .

صديقي :

هذه الفقرة مهمة جدا وهي التعامل مع مكتبة العناصر OBJECT LIBRARIES .

فكما تعلم أن مرحلة اعداد وتنسيق شكل النموذج وتحديد خصائص العناصر ITEMS مهمة وتستغرق الكثير من الوقت بالاضافة لمشكلة تحديد كل مطور الخصائص حسب ما يرى ولكن تخيل ما تستطيع فعلة باستعمال مكتبة العناصر .

تستطيع :

1- إعداد صور مختلفة من جميع العناصر المستعملة في النماذج للاستعمالات المختلفة ، فمثلا إعداد أكثر من عنصر TEXT ITEM:

الأول : هو مربع النص التقليدي المستعمل في الادخال وتعطية اسما وليكن GEN-TEXT مع تحديد الخصائص التي تناسب هذا الاستعمال مثل العرض والطول (HEIGHT,WIDTH) والألوان و الفونتات وغيرة .

الثاني : مربع نص يميز كلمة السر وتحدد له الخصائص التي تميزه وتجعل له اسم أخر وليكن PASSW-TEXT.وثالث ورابع حسب الاستخدامات التي تراها .

2- صور مختلفة من زر الأمر BUTTON أحدهم تقليدي والأخر ICONIC والأخر خاص باستدعاء FORM والأخر خاص باستدعاءLOVوأخر خاص باستدعاء تقرير REPORT .

3- صور مختلفة لجميع العناصر بما فيهم LOV وتحديد خصائصها من طول وعرض وألوان وفونتات.تخيل انشاء مثل هذه المكتبة يعتبر عمل قوي وممتاز.

والسؤال : ماذا تستفيد من انشاء مثل هذه المكتبة أو شئت قل مكتبات ؟

1- تستطيع اعداد النموذج FORM بسرعة تقارب 1/ 10 من السرعة التي تقوم بها باعداد نموذج من البداية وتحديد مواصفاتة كل مرة .

2- تضمن تطابق جميع النماذج في الخصائص أي تصبح الخصائص STANDARD .

3- تضمن عدم قيام كل مطور بتحديد الخصائص كما يشاء .

حقيقة تستفيد الكثير .

عموما : جرب إنشاء مكتبة أو مكتبات وجرب استعمالها وسوف تشعر بالفرق .

كما أنه سوف يفيدك كثيرا بإضافة مكتبة عناصر قيمة إلى أسطوانة الكتاب لتوفر عليك الكثير .

كما يمكن نسخ العناصر من مكتبة إلى مكتبة وكذلك حذف عناصر من مكتبة ويمكن نسخ العنصر إلى بيئة التطوير والتغيير فيه ثم نسخه بعد التعديل للمكتبة .

بالطبع كنا نود شرح خطوات تحقيق ذلك ولكنه يطول لذلك يمكنك الرجوع الى كتاب الطريق الى احتراف اوراكل .

قوالب النماذج Template Form

نتابع موضوع من الموضوعات المتقدمة ولا استغناء عنه في عالم التطوير والبرمجة وهو إعادة استخدام العناصر REUSE الذى بدأناه من قبل ونتابعه بكيفية إنشاء واستعمال قوالب النماذج TEMPLATE FORM ويعتبر هذا الموضوع هو أساس بناء التطبيقات لأي مطور أو شركة اعداد تطبيقات وفي هذا الفصل نتناول النقاط التالية :

- ما هي فكرة قالب نموذج TEMPLATE FORM ؟
- ما هي فوائد استخدام القالب ؟
- ما هي طريقة استخدم القالب ؟
- ما هي مكونات القالب ؟
- كيفية إنشاء نموذج قالب ؟
- كيفية بناء نموذج جديد علي قالب ؟

ما هي فكرة إنشاء قالب نموذج TEMPLATE FORM

هي فكرة إنشاء نموذج قياسى يوفر هذا النموذج كل مواصفات النموذج الذي تحتاجة في جميع نماذج التطبيق من دوال وأوامر ثم استعمال هذا النموذج كقالب TEMPLATE لبناء النماذج الاخري وبالتالي تأخذ النماذج الجديدة خصائص ودوال وإمكانيات النموذج الاصلي TEMPLATE مما يوفر الكثير من وقت التطوير وسوف يتضح ذلك من الفقرات

التالية وسوف يتضح أكثر عند إعداد تطبيق حقيقي باستخدام قالب النموذج TEMPLATE كما في هذا الكتاب.

فوائد استعمال قالب النموذج TEMPLATE FORM

يوفر ذلك الكثير من الفوائد منها:

1. مشاركة النماذج لنفس العناصر

عند قيامك باعداد تطبيق تلاحظ انك تستعمل بعض العناصر في جميع النماذج مثل الرسائل ALERTS وكذلك قائمة القيمة LOV والكثير من العناصر وبالتالي تضطر بدون استعمال القالب TEMPLATE لانشائها اكثر من مرة ولكن باستعمال القالب تجد أنك تنشئها فقط في القالب TEMPLATE وتستعملها في باقي النماذج المبنية علي القالب.

2. مشاركة الاوامر

من الفوائد التي يوفرها استعمال القالب هو إنشاء وحدات البرمجة أي الدوال والبرامج الفرعية ووضعها في مكتبة أوامر PLL LIB وعمل ATTACH لها في النموذج القالب وبالتالي عند بناء نموذج جديد علي القالب يمكنك الاستفادة من هذة الاوامر وذلك مثل دوال التحقق من البيانات VALIDATION ودوال التعامل مع الأخطاء والرسائل وكذلك الأوامر المسؤلة عن الخصائص العامة للنموذج مثل MAXMIZE وغيره

3. الالتزام بقواعد عامة (مقاييس)

يؤدي اعداد قالب TEMPLATE واستعمالة لبناء نموذج آخر إلى توحيد قواعد التطوير مثل شكل النماذج وكذلك اشكال العناصر فبالطبع انشاء مكتبة عناصر OBJECT LIB وتحول عناصرها إلى SMART CLASSES كما سبق وعند إنشاء نموذج جديد نبني علي القالب تاخذ عناصر النموذج الجديد نفس خصائص القالب TEMPLATE عن طريق SMART CLASSES.

4. قصر وقت التطوير

بالطبع بناء نموج قالب TEMPLATE FORM يحتوي علي الخصائص العامة ومكتبة الدوال PLL ومكتبة عناصر OLB بها SMART CLASSES يؤدي إلى زيادة سرعة إنشاء نماذج

جديد مبنية علي هذا القالب عكس قيامك بإنشاء نموذج جديد كل مرة وتحديد خصائص العناصر وكذلك كتابة الأوامر والدوال.

5. تنظيم طريقة التطوير

تؤدي عملية إنشاء القالب إلى تنظيم فريق المطورين فعند البدء في مشروع جديد (NEW PROJECT) يتم تقسيم العمل بحيث يقوم مجموعة من المبرمجين المحترفين باعداد قوالب النظام TEMPLATE ويقوم مجموعة من المبرمجين المبتدءين باستعمال القوالب في اعداد النماذج لتطبيق وبالتالي لا يتم تكرار الوقت والجهد بالاضافة لذلك لاتحتاج إلى عدد كبير من المبرمجين المحترفين.

عيوب استعمال القوالب TEMPLATE

1. عدم تطور مهارات بعض المطورين

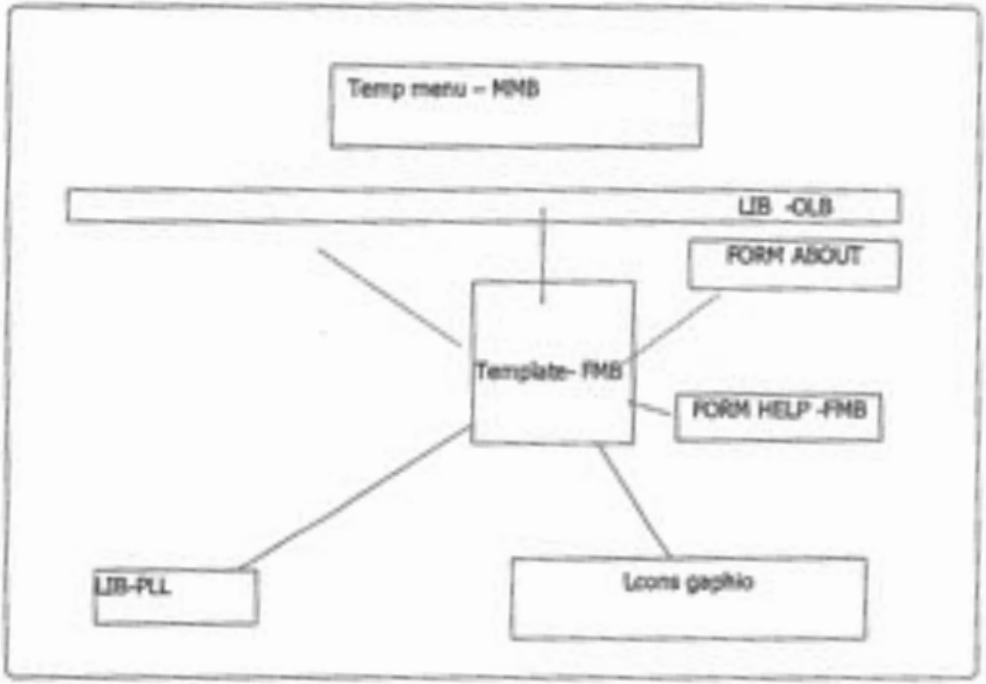
يؤدي استعمال نموذج قالب TEMPLATE FORM من قبل البرمجين المبتدئين لإنشاء نموذج وعدم كتابة الأوامر المطلوبة لهذة القوالب إلى ضعف المستوي لأن القالب يوفر معظم المطلوب وهم يقومون بعمل نماذج تشابه القالب بدون بذل مجهود كبير وبالتالي يضعف المستوي وأكثر من ذلك عند وجود مشكلة تحول إلى المبرمجين المحترفين وبالتالي يظل المبرمجين المبتدئين بعيد عن عملية تطوير المهارات والخل في ذلك محاولة مشاركتهم في عملية التطوير ومراجعة أعمالهم.

مكونات نموذج القالب TEMPLATE FORM

لتوضيح المكونات والعناصر التي يمكن أن يحتوي عليها TEMPLATE انظر إلى الشكل (15-26).

في هذا الفصل عرضنا مقدمة عن قالب النموذج TEMLATE FORM و تناولنا بعض هذة الاجزاء وكيفية إنشائها مثل مكتبة الكائنات EBJECT LIB وتحويل عناصرها إلى SMART CLASSES وكذلك حزم الدوال والبرمجة الفرعية PACKAGES وكذلك مجموعة الخصائص المرئية VISUAL ATTRIBUTE ماذال هناك عناصر وجوانب أخري لقالب

النموذج TEMPLATE FORM نريد تناولها ولكن نتركها لموضوع اخر نتناقش فية عملية التطوير بصفة عامة ولكن نريد أن نشرح هذا من وجهة النظر العملية كيفية استعمال FORM BUILDER لإنشاء نموذج ثم استعمالة كقالب TEMPLATE FORM وإنشاء نموذج جديد مبني عليه.



شكل (26-15)

ويمكنك مشاهدة كيفية تطبيق هذه القواعد فى كتاب الطريق إلى احتراف **Oracle**.

الفصل السادس عشر

الرسم Graphics

فى هذا الفصل Programming Concepts 16

فى هذا الفصل نشرح موضوع من الموضوعات الجذابة وهو موضوع الرسم Graphics وذلك من خلال النقاط التالية :

الرسم Graphics فى VB.NET,C#

- التعامل مع الأداة Picture Box
- ما هى الفصيلة Graphics
- ما هى دوال الرسم Graphics Methods
- خطوات استعمال دوال الرسم
- أمثلة متنوعة على دوال الرسم
- الرسم Graphics فى Java
- فصيلة الرسم Graphics Class
- دوال الفصيلة Graphics Methods

موضوع الرسم Graphics من الموضوعات الجذابة والتى تضيف للبرنامج الكثير من المؤثرات التى قد تحتاجها فى بعض التطبيقات.

فى لغة VB.NET

التعامل مع الأداة Picture Box

شرحنا فيما سبق كيفية عرض صورة فى الأداة Picture Box ولكن فى وقت التصميم وذلك بتوقيع أداة Picture Box على الـ Form ثم عرض الخصائص F4 واختيار الخاصية Image ثم اختيار الصورة المطلوبة.

ويمكن عرض صورة فى الأداة Picture Box باستخدام أوامر البرمجة وذلك بكتابة الصيغة التالية:

```
PictureBox1.image= New BitMap(filename)
PictureBox1.image= New Metafile(filename)
```

ونستخدم الدالة ()BitMap إذا كنا نتعامل مع صور بالامتداد bmp ولتوضيح ذلك تابع الخطوات التالية:

1- قم بإنشاء تطبيق جديد من النوع Windows Application

2- وقع الأداة Picture Box من مربع الأدوات Tool Box على الـ Form

3- وقع زر أمر Button1 وغير عنوانه إلى Load

4- اكتب بحدث Click لزر الأمر Buttton1 السطر التالى:

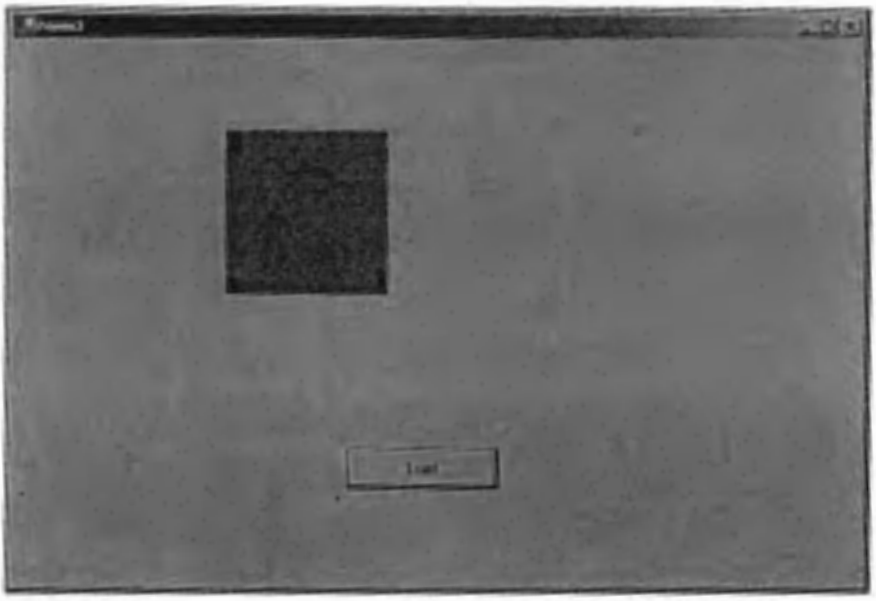
```
PictureBox1.Image = New Bitmap("c:\Gone Fishing.bmp")
```

فى لغة C#

```
pictureBox1.Image = new Bitmap("c:\\myimage.jpg");
```

فى هذا السطر تم تسجيل قيمة الخاصية Image لأداة عرض الصورة PictureBox1 باستعمال الفصيلة BitMap مع إرسال اسم ملف الصورة والمسار الكامل له كمعامل لدالة بناء الفصيلة وبالتالى يتم تحميل هذه الصورة فى مربع الصورة Picture Box1 عند الضغط على الزر Button1 يمكنك تغيير ملف الصورة بما يتناسب معك.

5- نفذ البرنامج واضغط الزر Load يتم تحميل الصورة كما فى الشكل (1-16) .



الشكل(1- 16)

ما هى الفصيلة Graphics

توفر هذه الفصيلة إمكانيات الرسم Graphics من إنشاء عنصر رسم Graphics Object مع توفير دوال الرسم المطلوبة.

ويتم استعمال الفصيلة Graphics بتعريف هدف Object منها كما فى السطور التالية:

```
1.    Dim MyGraphics as Graphics
2.    MyGraphics=Me.CreateGraphics
```

فى لغة C#

```
Graphics  MyGraphics
MyGraphics=this.CreateGraphics();
```

- فى السطر رقم 1 تم الإعلان عن المتغير MyGraphics كهدف Object من نوع الفصيلة Graphics.
- فى السطر رقم 2 تم إنشاء هدف الرسم Graphics Object مع المتغير Me الذى يشير إلى الـ Form والإشارة إلى هذا الهدف بالمتغير المعرف فى السطر رقم 1 MyGraphics

وبهذا عند استدعاء دوال الرسم مع الهدف MyGraphics يتم الرسم على الـ Form ولكن للرسم على أداة اخرى غير الـ Form يتم إنشاء عنصر الرسم مع الأداة الأخرى وسوف نوضح ذلك.

استعمال دوال الرسم Graphics Methods

بعد إنشاء هدف Object رسم يتم استعماله فى استدعاء دوال الرسم ولتوضيح ذلك تابع الخطوات التالية:

1- قم بإنشاء تطبيق جديد من النوع Windows Application.

2- وقع زر أمر Button1 واكتب بحدث Click له السطور التالية شكل (2-16):

```
Dim g As Graphics
g = Me.CreateGraphics
Dim p As New Pen(Color.Red)
g.DrawEllipse(p, 170, 100, 170, 190)

Dim b As New SolidBrush(Color.Blue)
g.FillRectangle(b, 300, 200, 200, 150)

Dim fo As New Font("Times", 40)
g.DrawString("Visual Basic.Net", fo, Brushes.Brown, 25, 75)
```

الشكل(2- 16)

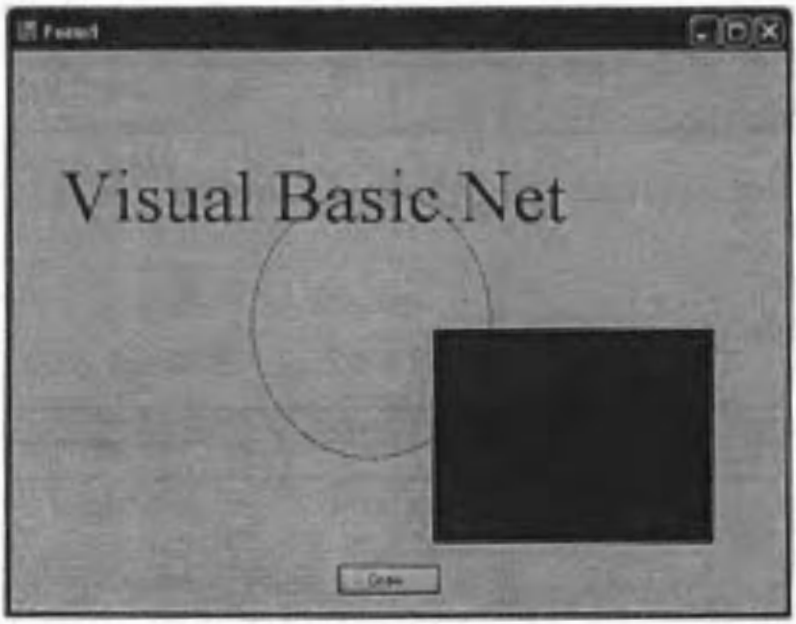
فى لغة C#

```
101         private void Button1_Click(object sender, System.EventArgs e)
102         {
103
104     Graphics    MyGraphics ;
105     MyGraphics=this.CreateGraphics() ;
106     Pen RedPen = new Pen(Color.Red , 3) ;
107     MyGraphics.DrawEllipse (RedPen,170,170,150,190) ;
108
109         SolidBrush b=new SolidBrush (Color.Blue) ;
110         MyGraphics.FillRectangle (b,300,200,200,150) ;
111         Font Fo=new Font ("Times",20) ;
112     MyGraphics.DrawString ("C#.NET",Fo,b,25,75) ;
113         }
```

في هذه السطور:

- في السطر رقم 61 تم الإعلان عن متغير هدف Object بالاسم g من نوع الفصيلة Graphics.
- في السطر رقم 62 تم استخدام هذا المتغير g للإشارة إلى هدف من الفصيلة Graphics تم إنشاءه باستخدم الدالة CreateGrapgics مع المتغير Me الذى يشير إلى الـ Form وبالتالى اصبح المتغير g جاهز لاستعماله للرسم على الـ Form.
- في السطر رقم 63 تم تعريف متغير هدف Object بالاسم p للإشارة لمواصفات قلم الرسم الذى يتم تعريفه باستعمال الفصلة Pen وتم تحديد لون قلم الرسم باللون الأحمر Red.
- في السطر رقم 64 تم استعمال متغير هدف الرسم g في استدعاء دالة رسم شكل بيضاوى DrawEllips() مع تحديد مواصفات قلم الرسم الذى حدد بالمتغير p وتحديد أبعاد الشكل البيضاوى المحددة بالقيم المعطاة.

3- نفذ البرنامج F5 واضغط الزر Draw تحصل على الرسومات التى حددناها كما في الشكل (3-16).



الشكل(3- 16)

الحدث Form_Paint

يحدث هذا الحدث فى بداية تشغيل البرنامج بعد الحدث Form_Load وكذلك عند إعادة اظهار الـ Form وهو يفيد فى إظهار الرسومات تلقائيا وعند إخفاء وإظهار ال Form.

ولتجربة ذلك قم بقص Cut سطور أوامر الرسم السابقة من الحدث Button1_Click وضعها فى الحدث Form1_Click وإعادة تنفيذ البرامج وشاهد النتيجة.

دوال الفصيلة Graphics

تحتوى الفصيلة Graphics على معظم دوال الرسم Drawing Methods والجدول التالى يعرض بعض هذه الدوال.

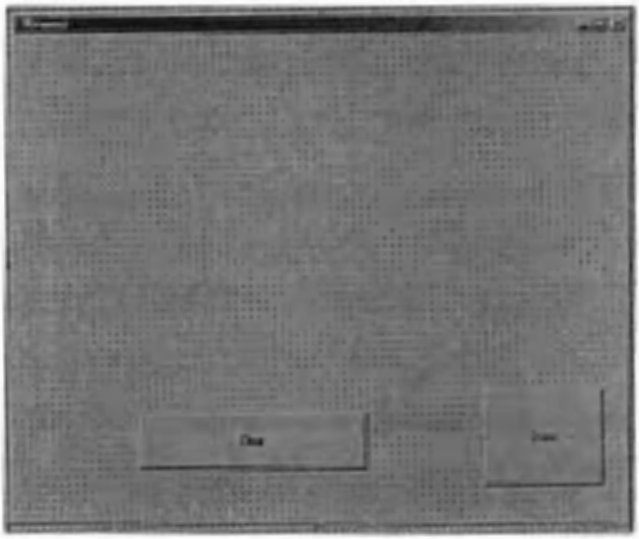
الدالـــــة	الوظيفـــــة
DrawArc	تقوم برسم قوس Arc.
DrawClosedCurve	تقوم برسم منحنى مغلق.
DrawCurve	تقوم برسم منحنى.
DrawEllipse	تقوم برسم شكل بيضاوى.
DrawIcon	تقوم برسم شكل أيقونة.
DrawImage	تقوم برسم صورة محددة الأبعاد.
DrawLine	تقوم برسم خط Line.
DrawLines	تقوم برسم مجموعة خطوط متتالية.
DrawPie	تقوم برسم Pie.
DrawPolygon	تقوم برسم شكل متعدد الخطوات.
DrawRectangle	تقوم برسم مستطيل.
DrawRectangles	تقوم برسم مجموعة مستطيلات.
DrawString	تقوم برسم عبارة نصية محددة الفونت واللون.

تقوم برسم منحني مغلق مع ملئ Fill مساحته المغلقة باللون والتهشير المحدد.	FillClosedCurve
تقوم برسم شكل بيضاوي مغلق مع ملئ Fill مساحته المغلقة باللون والتهشير المحدد.	FillEllipse
تقوم برسم Pie مع ملئ Fill مساحته المغلقة باللون والتهشير.	FillPie
تقوم برسم شكل متعدد الخطوات مع ملئ Fill مساحته المغلقة باللون والتهشير.	FillPolygon
تقوم برسم مستطيل مع ملئ Fill مساحته المغلقة باللون والتهشير.	FillRectangle
تقوم برسم مجموعة مستطيلات مع ملئ Fill مساحته المغلقة باللون والتهشير	FillRectangles

مثال : لتوضيح بعض دوال الرسم تابع الخطوات التالية:

1- قم بإنشاء تطبيق جديد من النوع Windows Application.

2- صمم النموذج Form1 كما بالشكل (4-16).



الشكل(4- 16)

3- الزر Draw يؤدى إلى رسم بعض الأشكال الهندسية ولتحقيق ذلك اكتب بالحدث Click السطور التالية:

```
Dim g As Graphics
g = Me.CreateGraphics
Dim PolyPoints(2) As Point
PolyPoints(0) = New Point(60, 10)
PolyPoints(1) = New Point(10, 170)
PolyPoints(2) = New Point(90, 170)
g.FillPolygon(New SolidBrush(System.Drawing.Color.Blue), _
PolyPoints)
Dim f As New Font("Ariel", 24)
g.DrawString("*", f, Brushes.White, 50, 70)
Dim Curvpoints(5) As Point
Curvpoints(0) = New Point(50, 10)
Curvpoints(1) = New Point(100, 40)
Curvpoints(2) = New Point(160, 210)
Curvpoints(3) = New Point(50, 150)
Curvpoints(4) = New Point(75, 90)
Curvpoints(5) = New Point(150, 200)
g.DrawCurve(New Pen(System.Drawing.Color.Red, 2), Curvpoints)
g.DrawArc(New Pen(System.Drawing.Color.Black, 3), _
    New Rectangle(10, 150, 150, 10), 0, 180)
```

فى هذه السطور:

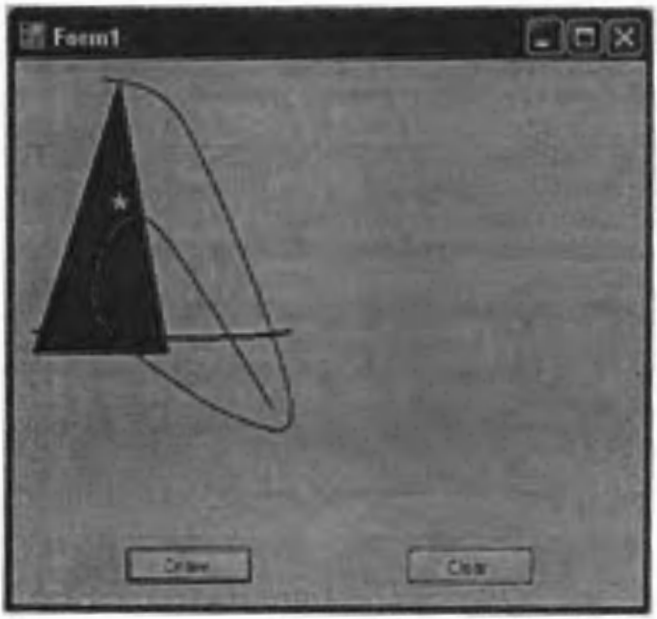
- فى السطر رقم 1 يتم الإعلان عن متغير هدف بالاسم g من نوع الفصيلة Graphics.
- فى السطر رقم 2 يتم إنشاء هدف الرسم g مع المتغير Me الذى يشير إلى الـ Form وبالتالى عند استعمال دوال الرسم مع المتغير g يتم الرسم على الـ Form.
- فى السطر رقم 3 يتم الإعلان عن مصفوفة نقط بالاسم PolyPoints من النوع Point.
- فى السطور 4و5و6 يتم تسجيل قيم نقط عناصر المصفوفة.
- فى السطر رقم 7 يتم استدعاء الدالة () FillPolygon التى تؤدى إلى رسم متعدد الخطوط حسب المواصفات المحددة وكذلك قيم النقط المحددة فى المصفوفة PolyPoints.
- بالمثل تقوم باقى السطور باستدعاء الدالة ()DrawCurve التى ترسم منحنى Curve بعد تجهيز النقط اللازمة للرسم بالمثل يتم استدعاء الدالة DrawArc لرسم قوس بعد تحديد مواصفات الرسم.

4- الزر Clear يؤدى إلى مسح خلفية النموذج Form1 باللون المحدد الهندسية ولتحقيق ذلك اكتب بالحدث Click السطور التالية:

```
1. Dim g As Graphics
2. g = Me.CreateGraphics
3. g.Clear(System.Drawing.Color.Beige)
```

فى هذه السطور يتم استدعاء الدالة ()Clear التى تؤدى إلى مسح الشاشة (النموذج Form) باللون المحدد وهو Beige.

5- نفذ البرنامج ثم اضغط الزر Draw تحصل على بعض الرسومات كما فى الشكل (5-16).



الشكل(5-16)

6- اضغط الزر Clear تلاحظ مسح النموذج Form1 باللون المحدد.

الفصيلة GraphicsPath

هذه الفصيلة من فصائل الرسم الممتازة والتى توفر الكثير من الإمكانيات وهى مع الفصيلة Region تفعل الكثير، فيمكنها مثلا عمل قص لمنطقة محددة مما يضيف شكل جمالى للرسومات ولتوضيح ذلك تابع الخطوات التالية:

1- عد إلى البرنامج السابق وفى الحدث Form_paint اكتب السطور التالية:

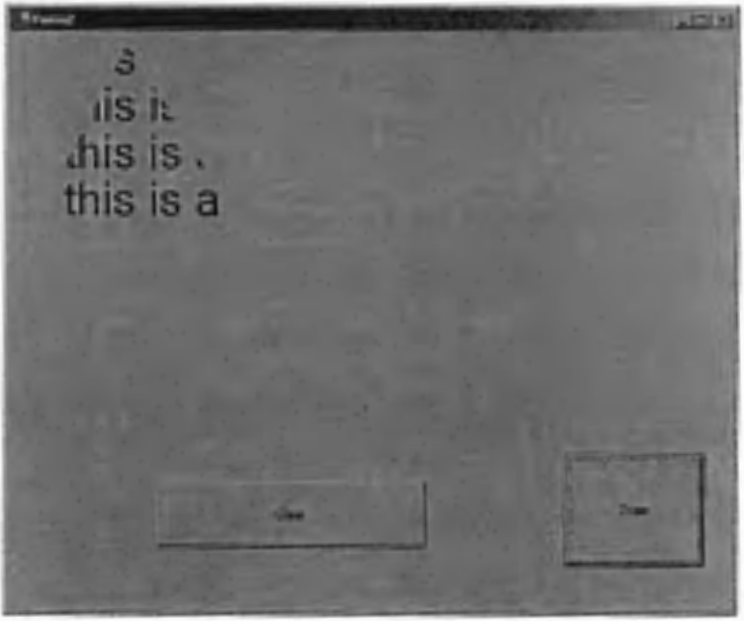
```
1. Dim g As Graphics = e.Graphics
2. Dim gp As New GraphicsPath()
3. Dim p(2) As Point
4. p(0).X = 100
5. p(0).Y = 10
6. p(1).X = 30
7. p(1).Y = 150
8. p(2).X = 220
9. p(2).Y = 200
10. gp.AddPolygon(p)
11. Dim r As New Region(gp)
12. g.Clip = r
13. Dim f As New Font("Ariel", 30)
14. Dim y As Integer
15. For y = 0 To 120 Step 40
16. g.DrawString("this is a Test", f, Brushes.Black, 40, y)
17. Next y
```

فى هذه السطور:

- يتم كتابة هذه السطور فى الحدث Form_paint الذى ينفذ تلقائيا عند رسم النموذج Form كما أشرنا وفيها يتم تعريف متغير هدف Object من الفصيلة GraphicsPath التى تحتوى على دوال تحقق كثير من عمليات الرسم مثل تكوين منطقة Region للتعامل معها.
- فى السطور 3و4و5و6و7و8و9 يتم تعريف مصفوفة نقط Points ثم تسجيل قيم هذه النقط.
- فى السطر رقم 10 يتم AddPolygon() التى تضيف متعدد الخطوط المحدد بالنقط السابقة ثم استعماله لتحديد منطقة Region كما فى السطر رقم 11.
- فى السطر رقم 12 يتم تحديد خاصية القطع Clip لتغير هدف الرسم g بالقيمة r التى تشير إلى المنطقة المحددة مسبقا.

بعد ذلك يتم رسم العبارة this is a test أكثر من مرة باستعمال التكرار for مع مراعاة المنطقة المحددة.

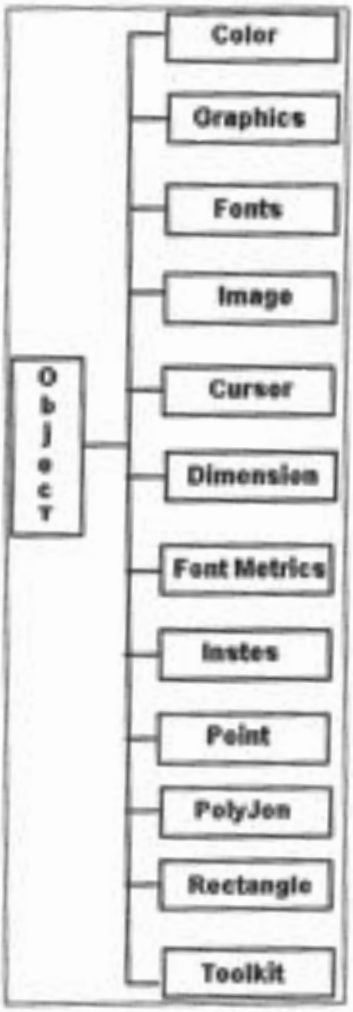
2- نفذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (6-16) .



الشكل(6- 16)

فى لغة Java

توفر لغة Java مجموعة من الفصائل الرسومات Graphics نعرضها كما فى الشكل (7-16).



الشكل (7- 16)

في الشكل مجموعة من الفصائل الرسم سوف نتناول معظمها بالشرح.

فصيلة الرسم Graphics Class

تعتبر الفصيلة Graphics من أشهر فصائل الرسم وهي الأساس للرسومات وقبل أن نعرض كيفية استعمال الفصيلة وكتابة البرامج لها نعرض أولاً دوال الفصيلة.

دوال الفصيلة Graphics Methods

تحتوى الفصيلة Graphics على مجموعة من دوال الرسم التي تأخذ الصيغة.

drawxxx()

ويتم تغيير xxx حسب الشكل المطلوب رسمه والجدول التالى يعرض الدوال الأعضاء فى الفصيلة Graphics ووظيفة كل دالة.

الدالة	الوظيفة
Draw3DRect()	ترسم مستطيل ثلاثى الأبعاد.
DrawArc()	لرسم قوس.
Draw Line()	لرسم خط line.
DrawOval()	لرسم شكل بيضاوى أو دائرى.
DrawPolygon()	لرسم شكل متعدد الاضلاع.
Drawpoly Line()	لرسم شكل متعدد الخطوط.
DrawRect()	لرسم مستطيل.
DrawReoundRect()	لرسم مستطيل مستدير الحواف.
Fill3DRect()	لتلوين مساحة مستطيل ثلاثى الأبعاد.
FillArc()	لتلوين مساحة قوس.
FillOval	لتلوين مساحة شكل بيضاوى أو دائرى.
FillPolygon()	لتلوين مساحة شكل متعدد الأضلاع.
FillRect()	لتلوين مساحة مستطيل.
FillRound Rect()	لتلوين مساحة مستطيل مستدير الحواف.

مثال :

لتوضيح كيفية استعمال دوال الرسم تابع المثال التالى:

```
import Java. Applet. *;
import Java. Awt. * ;
public class graph1 extends Applet {
    Public void init() {
        SetBackground (color. White);
        Setforeground (Color. Black);
                    }
pPublic void paint (Graphics g) {
    g. drawstring ("First graphics prog .. ");
                }
                    }
```

شرح السطور:

هذا البرنامج يتم استغلال (استدعاء) الدالة drawstring () لرسم النصوص وهى عضوة فى الفصيلة Graphics والتى تقوم برسم عبارة حرفيه وهى العبارة "... First graphics prog".

دوال الرسم Drawing

فى هذه النقرة نقوم بتجربة استعمال دوال الرسم التى عرضناها فى الجدول السابق.

رسم خطوط LINES

تستعمل الدالة drawLine() فى رسم الخطوط وتأخذ الصيغة العامة التالية:

```
void drawline (int x1, int y1,int x2, int y2)
```

وفى هذه الصيغة يتم رسم خط يصل بين النقطتين (x1,y1), (x2,y2) ولتوضيح ذلك تابع المثال التالى:

مثال:

```
import Java. Applet. *;
import Java. Awt. *;
    public class lineDemo extend Applet {
            public void paint (Graphics g) {
            for (int i =0; i <200, i ++
            g. drawline (0,0,200,i );
```

فى هذا البرنامج

- تم تعريف هدف بالاسم g من الفصيلة graphics ثم تم استدعاء الدالة drawLine().
- داخل جملة التكرار for مما يؤدى الى رسم مجموعة خطوط.
- رسم المستطيلات Drawing Rectangle.

توجد اكثر من دالة للتعامل مع المستطيلات بأشكالها المختلفة نعرضها كما يلى:

1. الدالة drawRect()

وهى أبسط أنواع الدوال لرسم المستطيل ونأخذ أربعة معاملات كما فى الصيغة العامة التالية:

```
void drawRect (int x, int y, int width, int height)
```

فى هذه الصورة يتم إرسال الركن الشمالى العلوى من المستطيل كمعامل للدالة وهو

احداثى النقطة (x,y)، وكذلك عرض المستطيل وهو المعامل width وطول المستطيل المعامل height.

2. الدالة drawRoundRect()

وتقوم برسم مستطيل مستدير الحواف يسمى Rounded وتأخذ الدالة الصورة العامة التالية:

```
void drawroundrect (int x, int y, int width, int height, int arcwidth, int airtight)
```

3. الدالة FillRoundRect()

وتقوم برسم مستطيل مستدير الحواف كما سبق ولكن مع ملء مساحته باللون وتأخذ نفس الصيغة السابقة كما يلى:

```
void fillRoundRect (int x, int y, int width, int height, int arcwidth, int airtight)
```

4. الدالة draw3Drect()

وتقوم برسم مستطيل مجسم فى ثلاثة أبعاد (3d) وتأخذ الصورة العامة التالية:

```
void draw3Drect (int x, int y, int width, int height, int arcwidth, int airtight)
```

5. الدالة Fill3Drect()

تقوم برسم مستطيل ثلاثى الأبعاد بالإضافة لملئ مساحته باللون ويأخذ الصورة العامة التالية:

```
void Fill3DRect (int x, int y, int width, int height, int arcwidth, int airtight)
```

6. الدالة ClearRect()

تقوم بمسح مساحة مستطيل ولكن بلون الخلفية المحدد وتأخذ معاملات دالة رسم مستطيل كما يلى:

```
void ClearRect (int x, int y, int width, int height, )
```

ولتوضيح هذه الدوال تابع المثال التالى:

مثال:

```
import Java. Applet. * ;
import Java. Awt. *;
```

```
public class RectangleDemo extends Applet {
    public void paint (Graphics g) {
    g.Setcolor (color. Blue);
    g.DrawRect (25,15,55,80);
    g.FillRect (25,15,55,80);
    // Round Rect
g. setcolor (color. Orange)
g. drawRoundRect (100,15,55,70,30,50);
g. FillRoundRect (100,15,55,70,30,50);
}
}
```

رسم أقواس

-الدالة drawArc() وتأخذ الصيغة العامة التالية:

void draw3dRect (int x, int y, int width, int height, int Start Angle, int airtight)

الدالة Fill Arc()

وتأخذ الصيغة العامة التالية:-

void draw3dRect (int x, int y, int width, int height, int Start Angle, int arc Angle)

لتوضيح ذلك تابع المثال التالي:

```
import Java. Applet. * ;
import Java. Awt. *;
public class ArcDemo extends Applet {
    Public void paint (Graphics g) {
    g. drawArc (10,10,60,60,30,300);
    g. FillArc (10,10,60,60,30,300);
        }
            }
```

رسم متعدد الخطوط Polygon

دالة متعدد الخطوط هي دالة تأخذ مجموعه من النقاط وتقوم برسم خطوط بينها وذلك بدلاً من كتابة أكثر من دالة رسم خط وتوجد أكثر من دالة نفرضها فيما يلي:

الدالة drawpolygon()

وتقوم برسم متعدد الخطوط بناء على النقاط وذلك كما في الصيغة العامة التالية:-

```
void polygon (int [] x points, int y points, int points)
```

في هذه الصيغة يتم إرسال مجموعة من النقاط

الدالة ()Fillpolygon

وتأخذ الصيغة التالية:-

```
void polygon (int [] x points, int[]  y points, int points)
```

فصيلة الألوان ودوالها Color Classes

الفصيلة color مسؤلة عن التعامل مع الألوان وتسجيلها ويتضح ذلك من المثال التالى:

```
g.setBackground(Color.red)
```

فى هذا السطر يتم استدعاء الدالة ()setBackground لتغيير لون الخلفية مع تحديد اللون الأحمر red.

فصيلة الفونتات ودوالها Fonts class

يتم استعمال الفصيلة Font فى تحديد مواصفات الفونت المستعمل وذلك كما فى المثال التالى:

```
Font f=new Font("Arial",Font.Bold,36);
g.setFont(f);
```

الفصل السابع عشر

التعامل مع النصوص والأعداد والملفات
String,Number,Date,files

Programming Concepts فى هذا الفصل

17

فى هذا الفصل نشرح كيفية التعامل مع حافظه برنامج النوافذ Clipboard ذلك بالإضافه إليها والنسخ فيها من خلال النقاط التالية :

- التعامل مع النصوص String
- التعامل مع الأعداد Numbers
- التعامل مع التواريخ Dates
- التعامل مع الملفات النصية

فى VB.NET,C#

التعامل مع النصوص String

الـ String هو أى عبارة حرفية توضع بين علامتى تنصيص " " ويتم الإعلان عنها والتعامل معها كما فى السطور التالية:

```
Dim s As  String ="Hi  From VB.Net"
```

أو الطريقة:

```
Dim s As String
S="Hi From VB.Net"
```

وتحتوى الفصيلة string على مجموعة من الأعضاء التى تسهل التعامل مع العبارات الحرفية كما فى الجدول التالى:

Chars	تعيد حرف من داخل العبارة الحرفية المحدد بالموضع داخل العبارة الحرفية.
Length	تعيد عدد حروف العبارة الحرفية.
Compare	تقارن متغيرين من نوع String.
CompareTo	تقارن المتغير الذى تستدعى معه مع متغير أخر.
Concat	لاضافة عبارة حرفية او محتوى متغير Stringالى عبارة حرفية أو متغير String.
Copy	تنسخ محتوى متغير String الى أخر.
EndsWith	تختبر نهاية متغير String هل ينتهى بكلمة محددة أم لا.
Equals	تحدد نتيجة مقارنة متغيرين String.
Format	تقوم بتحويل العبارة المكتوبة الى الصيغة التى ترغبها.
IndexOf	تحدد أول ظهور لحرف معين داخل عبارة حرفية.
Insert	تضيف حرف فى مكان محدد داخل عبارة حرفية.
LastIndexOf	تحدد أخر ظهور لحرف معين داخل عبارة حرفية

Remove	تحذف عدد محدد من الحروف حسب التحديد.
Replace	تستبدل حروف بحروف أخرى.
StartsWith	تختبر بداية عبارة حرفية.
Substring	تعيد عدد محدد من الحروف من عبارة حرفية حسب العدد وحسب البداية.
ToLower	تحول الحروف الكبيرة الى الحروف الصغيرة.
ToString	تحول قيمة معينة الى عبارة حرفية String حتى يمكن التعامل معها بالدوال الحرفية.
ToUpper	تحول الحروف الصغيرة الى الحروف الكبيرة.
Trim	تقوم بقص (حذف)حرف محدد من بداية ونهاية عبارة حرفية ويستخدم بكثرة لحذف المسافة Space.
TrimEnd	تقوم بقص (حذف)حرف محدد من نهاية عبارة حرفية ويستخدم بكثرة لحذف المسافة Space.
TrimStart	تقوم بقص (حذف)حرف محدد من بداية عبارة حرفية ويستخدم بكثرة لحذف المسافة Space.

التعامل مع الأعداد Numbers

الإعداد Numbers سواء الصحيحة Integers أو الحقيقية Float تختلف عن العبارة الحرفية String فى إمكانية إجراء العمليات الحسابية عليها مثل الجمع والطرح.

التعامل مع التواريخ Dates

هى القيم التى تأخذ شكل وقيمة تاريخ Date أو وقت Time وتوجد بعض الدوال التى توفر معظم العمليات المطلوبة مع قيم التاريخ والوقت مثل الدوال التالية:

Date, Day, DayOfWeek, DayOfYear, Hour, Millisecond, Minute, Month, Now, Second, Ticks, TimeOfDay, Today, UtcNow, Year, Add,AddDays, AddHours, AddMilliseconds, AddMinutes, AddMonths, AddSeconds, AddTicks, AddYears, Compare, CompareTo, DaysInMonth, Equals, FromFileTime, FromOADate, GetDateTimeFormats

فى لغة Java

الملفات النصية

توجد أكثر من طريقة للتعامل مع الملفات ، وتستخدم كل طريقة مع الحالة التى تناسبها وهذه الطرق هى :

1- التعامل مع ملفات قواعد البيانات : وتم شرحها فى الفصول السابقة.

2- فتح الملفات بطرق التسلسلية : ويتم استعمال عند الرغبة فى التعامل مع الملفات النصية سواء بالكتابة أو بالقراءة مثل الملف AUTOEXEC.BAT.

3- التعامل مع الملفات العشوائية : وهو حفظ البيانات فى ملف بالصورة الثنائية (Binary) وذلك حسب التطبيقات التى تتطلب ذلك.

وتوجد مجموعة من الفصائل classes التى توفر عمليات التعامل مع الملفات منها.

BinaryReader, BinaryWriter, BufferedStream, Directory, DirectoryInfo DirectoryNotFoundException, EndOfStreamException, ErrorEventArgs, File FileInfo, FileLoadException, FileNotFoundException, FileStream FileSystemEventArgs ,FileSystemInfo, FileSystemWatcher

وتأخذ عملية القراءة من الملفات مجموعة الخطوات التالية:

1- التحقق من وجود الملف المطلوب التعامل معه.

2- فتح الملف.

3- تعريف متغير هدف Object من الفصيلة BinaryReader أو الفصيلة StreamReader حسب الطريقة التى ترغب فى التعامل مع الملف.

4- استخدام دوال Methods لقراءة البيانات.

5- إغلاق الملف.

وتأخذ عملية الكتابة فى الملفات مجموعة الخطوات التالية:

افتح الملف المطلوب التعامل معه وإذا كان موجود تتأكد من المستخدم هل تريد استبداله والكتابة عليه أم لا تعريف متغير هدف Object من الفصيلة BinaryWriter أو الفصيلة StreamWriter حسب الطريقة التى ترغب فى التعامل مع الملف.

4-استخدام دوال Methods لكتابة البيانات فى الملف.

5- إغلاق الملف.

مع ملاحظة استعمال بلوك التعامل مع الأخطاء try..catch للتعامل مع أى أخطاء تظهر مثل عدم وجود الملف المطلوب القراءة منه أو عدم إمكانية إنشاء ملف جديد.

الفصيلة File

توفر هذه الفصيلة إمكانية تحقيق العمليات الأساسية مع الملفات وذلك من خلال مجموعة من الدوال Methods وأشهر هذه الدوال الدالة ()Open المسئولة عن فتح ملف.

AppendText,Copy,Create,CreateText,Delete,Exists,GetAttributes,GetCreationTime,GetLastAccessTime, GetLastWriteTime, Move, Open, OpenRead, OpenText, OpenWrite, SetAttributes, SetCreationTime, SetLastAccessTime, SetLastWriteTime

وتوفر الفصيلة File عمليات التعامل مع الملفات والفهارس مثل اختبار وجود الملف أو الفهرس وإيجاد معلومات عنه مثل الحجم وتاريخ الإنشاء.

التدفق Stream

فى لغة Java

جميع البرامج تتعامل مع المستخدم وتطلب منه بعض البيانات والمعلومات ثم تتعامل مع هذه البيانات ثم يتم عرض النتائج، إذن لابد أن تتعامل لغات البرمجة مع عمليتي الإدخال والإخراج.

سوف نتناول فى هذا الفصل كيفية قراءة البيانات كإدخال من المستخدم وكيفية عرض النتائج كإخراج.

هو فيض من المعلومات والبيانات من مصدر معين Source إلى جهة وصول Destination.

فمثلاً:

- إذا أدخلت بيانات إلى البرنامج فإن النتائج تظهر على الشاشة إذن هنا مصدر المعلومات هو لوحة المفاتيح وجهة الوصول هى الشاشة.
- إذا طبعت ملف فإن النتائج تظهر على آلة الطباعة إذن هنا مصدر المعلومات هو الملف الذى طبعته وجهة الوصول هى آلة الطباعة.
- إذا قمت بكتابة ملف قمت بحفظة فإنك تحفظه على القرص الصلب (Hard Disk) إذن هنا مصدر المعلومات هو الملف وجهة الوصول هى القرص الصلب.

- إذن هناك أكثر من مثال للتدفق Streams فهو ببساطة مجموعة بيانات تنتقل من مصدر إلى جهة وصول بصرف النظر عن نوع البيانات المنتقلة.
- قد تظن أنه لكي تستطيع التعامل مع أنواع التدفق Streams المختلفة، فإنك لابد أن تعرف عدة طرق في البرمجة بحيث تستطيع التعامل مع كل نوع من أنواع التدفق Streams المختلفة. ولكن هذا ليس صحيحاً، فكما سنرى توفر لغة الجافا Java عدة فصائل classes للتعامل مع أنواع التدفق Streams المختلفة بمنتهى السهولة.
- لاحظ أنك إذا كنت تكتب بيانات فإنك في هذه الحالة تكون المصدر Source اما إذا كنت تقرأ البيانات أو قبض المعلومات فإنك تمثل جهة الوصول Destination لهذه البيانات.
- إذا نظرنا نظرة عميقة للتدفق Streams فإننا سنجد أنها في الواقع أربعة أنواع رئيسية: InputStream, OutputStream, Reader, and Writer.

1. InputStream

توفر هذه الفصيلة class وظائف قراءة بايت byte من المعلومات من تدفق البيانات Stream. تحتوى هذه الفصيلة class على دالة method تسمى ()read هي المسئولة عن وظيفة القراءة من التدفق Stream. كل أنواع التدفق التي تتعامل مع عملية الإدخال Input Streams ترث من هذه الفصيلة class.

2. OutputStream

توفر هذه الفصيلة class وظائف كتابة بايت byte من المعلومات إلى التدفق Stream. تحتوى هذه الفصيلة class على دالة method تسمى ()write هي المسئولة عن وظيفة الكتابة إلى التدفق Stream. كل أنواع التدفق التي تتعامل مع عملية الإخراج Output Streams ترث من هذه الفصيلة class.

3. Reader

وهو يماثل الـ InputStream

4. Writer

وهو يماثل الـ OutputStream

إذن هنا نتساءل : ما الفرق بين هذه الفصائل classes؟

الفرق أن الفصائل classes المسماة Reader و Writer تدعم عملية تسمى Internationalization.

Internationalization

هى عملية تصميم البرنامج بحيث يمكنه أن يعمل مع أكثر من لغة بدون أى تغيير في البرمجة بمعنى:

انظر مثلاً لنظام تشغيل النوافذ Windows فإنه يدعم بالطبع اللغة الإنجليزية ولكن إذا أردت أن تحول لغة الواجهة لتصبح العربية فإنك لا تستطيع والحل أنك ستشترى نسخة أخرى لها واجهة باللغة العربية.

إذن لكى تستطيع تغيير اللغة فإنك ستبتاع نسخة أخرى ولكن عملية الـ Internationalizatio تعنى أنه بتصميم البرنامج بطريقة معينة فإنك ستستطيع تغيير اللغة بلا مشاكل وبدون شراء نسخة أخرى.

هذه العملية في غاية الأهمية خصوصاً في شبكة الإنترنت فمثلاً إذا أنشأت موقعك على شبكة الإنترنت باللغة العربية مثلاً وحاول أحد مستخدمي الشبكة الدخول على موقعك ونظام التشغيل على جهازه لا يدعم اللغة العربية ، فلن يستطيع قراءة اللغة العربية وبذلك فلن يمكنه معرفة أي معلومات عن موقعك.

تعرف عملية الـ Internationalization بعملية I18n لأنه يوجد 18 حرف يقعون بين حرفي I و n.

أشهر فصائل classes التدفق Streams

System.in

هو InputStream يكون مصدره لوحة المفاتيح

System.out

هو OutputStream تكون جهة وصوله هى الشاشة

تذكر أننا استخدمنا دالة println() الموجودة فى الفصيلة System.out لطباعة رسالة على الشاشة. إذن فالفصيلة class المسماة System.out هي المسئوله عن الطباعة على الشاشة.

سوف نرى بعض الاستخدامات لهاتين الفصيلتين classes للقراءة وأخذ البيانات من المستخدم فى الأمثلة القادمة.

الملفات Files

بالطبع لابد أن تتعامل مع الملفات لأنك تستطيع القراءة من ملف أو الكتابة إلى ملف.
سوف ترى فى المثال القادم كيفية التعامل مع الملفات.

مثال 1: التعامل مع الملفات

أولاً: هدف المثال

تريد أن تنشئ تطبيقاً يستطيع الحصول على بعض البيانات من ملف نصي اسمه **Data.txt** فلابد أن تتأكد أولاً من وجود هذا الملف وإذا كان موجوداً فإنك تريد أن تعرف بعض المعلومات عن هذا الملف باستخدام الدوال الآتية:

وظيفة الدالة	نوع القيمة المرتجعة	اسم الدالة
تحدد ما إذا كان الملف موجوداً أم لا	boolean	exists ()
تحدد ما إذا كان باستطاعتك القراءة من الملف أم لا	boolean	canRead ()
تحدد ما إذا كان باستطاعتك الكتابة إلى الملف أم لا	boolean	canWrite ()
تحدد المسار الكامل للملف	String	getAbsolutePath ()
تقوم بإنشاء مجلد (directory)	boolean	mkdir ()
تقوم بتحديد اسم الملف	String	getName ()
تقوم بتحديد ما إذا كان الملف مختفياً (Hidden) أم لا.	Boolean	isHidden ()
تقوم بتحديد ما إذا الـ object المنشأ من نوع File، مجلداً أم لا.	boolean	isDirectory ()
تقوم بتحديد ما إذا كان الـ object المنشأ من نوع File ملفاً أم لا	boolean	isFile()
تقوم بإرجاع حجم الملف بالبايت	long	length ()

اسم الدالة	نوع القيمة المرتجعة	وظيفة الدالة
setReadOnly ()	boolean	تقوم بجعل الملف للقراءة فقط (Read –Only)

يقوم هذا البرنامج ببيان استخدام هذه الدوال.

ثانياً: كود البرمجة:

```
1:  import java.io.*;
2:
3:  class MyFile
4:  {
5:   public static void main (String[]args)
6:   {
7:        File myFile = new File ("data.txt");
8:
9:        //Now the file does not exist on your hard
10:       System.out.println("The file does not exist on your hard");
11:       System.out.println("-----------------------------------");
12:       System.out.println("Exists: " + myFile.exists());
13:       System.out.println("Can Read: " + myFile.canRead());
14:       System.out.println("Can Write: " + myFile.canWrite());
15:       System.out.println("Absolute Path: "+ myFile.getAbsolutePath());
16:       System.out.println("File Name: " + myFile.getName());
17:
18:       //create the file
19:       try
20:       {
21:            System.out.println("---------------------------");
22:            System.out.println("Creating File: " + myFile.createNewFile());
23:       }
24:
25:       catch (IOException e)
26:       {}
27:
28:       //Now the file exists on your hard
29:       System.out.println("---------------------------");
30:       System.out.println("The file exists on your hard");
31:       System.out.println("---------------------------");
32:       System.out.println("Exists: " + myFile.exists());
33:       System.out.println("Can Read: " + myFile.canRead());
34:       System.out.println("Can Write: " + myFile.canWrite());
35:       System.out.println("Absolute Path: " + myFile.getAbsolutePath());
36:       System.out.println("File Name: " + myFile.getName());
```

```
37:     System.out.println("Hidden: " + myFile.isHidden());
38:     System.out.println("Directory: " + myFile.isDirectory());
39:     System.out.println("File: " + myFile.isFile());
40:     System.out.println("Length: " + myFile.length());
41:     System.out.println("Set Read Only: " + myFile.setReadOnly());
42: }
43:}
```

ثالثاً: شرح الكود:

- في السطر 1 يتم تضمين الحزمة package المسماة (java.io.*) لانها تتضمن فصائل التدفق STREAMS اللازمة لعمل البرنامج.
- في السطر 7 يتم إنشاء ملف اسمه data.txt
- لاحظ أنك أنشأت هدف object من نوع الفصيلة class المسماة File ومررت لدالة البناء constrcutor القيمة data.txt التي تمثل اسم الملف الذي نريد التعامل معه.
- لاحظ أيضاً أنه كونك أنشأت هذا الهدف object فإن ذلك لا يعني أن الملف اصبح موجوداً على القرص الصلب (Hard Disk) وستعرف حالاً كيف تستطيع إنشاءه على القرص الصلب.
- في السطور من 12 إلى 16 يتم استخدام الدوال الموجودة في الجدول السابق.
- لاحظ أن الملف data.txt غير موجود في نفس مسار البرنامج لأننا لم ننشئه بعد إذن فالملف غير موجود وبالطبع لا يمكنك القراءة منه أو الكتابة إليه لأنه غير موجود.
- في السطر 22 يتم إنشاء الملف فعلياً على القرص الصلب عن طريق الدالة createNewFile() .
- لاحظ أنك في هذه الحالة أنشأت الملف في نفس مسار البرنامج.
- فمثلاً نفترض أن الملف Files.java موجود في المسار E:\ على القرص الصلب (Hard Disk) إذن بعد تنفيذ السطر 22 فإنك ستجد الملف Data.txt موجوداً في نفس المسار وهو E:\.
- تم تنفيذ السطر 22 داخل try بسبب إمكانية حدوث استثناء Exception فقد لا تستطيع عمل ملف بسبب مثلاً عدم وجود مساحة وهنا نوع الاستثناء Exception هو IOException.
- في السطور من 29 إلى 41 يتم بيان استخدام باقي الدوال الموجودة في الجدول السابق.

- لاحظ أنك أنشأت الآن الملف data.txt فعلياً على القرص الصلب إذن فهو موجود وتستطيع الكتابة إليه والقراءة منه وتستطيع معرفة إذا كان مخفياً (Hidden) أم لا.
- في السطر 38 يتم اختبار الهدف object الذى انشأناة باسم myFile. هل هو مجلد (Directory) أم لا. بالطبع فنحن أنشأنا ملفاً عادياً وسنرى فيما بعد كيف ننشئ مجلداً.
- في السطر 39 يتم اختبار myfile هل هو ملف عادى أم لا وبالطبع الإجابة هى نعم أو true.
- في السطر 40 يتم معرفة حجم الملف عن طريق الدالة () length. انظر شكل (15-1).
- في السطر 41 يتم جعل الملف Read – only اى للقراءة فقط أى أنك إذا ذهبت للمسار E:\ الموجود فيه الملف النصي data.txt وقمت بالضغط بالزر الأيمن للفأرة على الملف Data.txt ثم اخترت Properties فإنك ستجد أن الملف Read only
- لاحظ أن Read – only تعنى أنه يمكنك القراءة من الملف فقط ولا يمكنك الكتابة إليه أو تعديله.
- لاحظ أيضاً أنه إذا كان الملف موجوداً فإن محاولة تخليق الملف فى سطر 22 ستعيد القيمة false لأنك لن تستطيع تخليق ملف موجود أصلاً.

رابعاً: نتيجة التنفيذ:

بعد تنفيذ البرنامج فستظهر النتيجة على الشاشة كما في الشكل (1-17)

الشكل (1-17)

الشكل (17-1)

Directories : (المجلدات)

كما تعاملنا مع الملفات فإننا نريد أن نتعامل مع المجلدات Directories فنريد أن نعرف كيف ننشئ مجلداً وكيف نعرض أسماء المجلدات والملفات الموجودة فى مسار معين أى مثل أمر dir فى الـ Dos.

قراءة بايت Byte من الفصيلة System.in

ما نريد أن نفعله الآن هو أن نطلب من المستخدم إدخال حرف فإذا أدخل حرف X فذلك يعنى أنه يريد إنهاء البرنامج والخروج منه (Exit).

مثال 3: قراءة بايت Byte من الفصيلة System.in

أولاً: هدف المثال:

نريد توضيح كيفية قراءة بايت Byte من الفصيلة System.in أي أننا سنقرأ البايت المدخلة من لوحة المفاتيح.

ثانياً: كود البرمجة:

```
1:  import java.io.*;
2:
3:  class Read
4:  {
5:   public static void main ( String[]args )
6:   {
7:        int i = 0;
8:        InputStream is = System.in;
9:
10:       System.out.println("Enter a character");
11:
12:       while ( (char)i != 'x' )
13:       {
14:            try
15:            {
16:                 i = is.read();
17:            }
18:
19:            catch ( IOException e )
20:            {
21:                 System.out.println(e);
22:            }
23:       }
24: }
25: }
```

ثالثاً: شرح الكود:

- في السطر 8 يتم إنشاء هدف object من نوع الفصيلة InputStream واسمه is ثم جعلنا is هو System.in.
- لاحظ أننا نريد أن نقرأ بايت من لوحة المفاتيح وكما ذكرنا من قبل فإن System.in هو عبارة عن InputStream ومصدرة هو لوحة المفاتيح .
- في السطر 10 يتم طباعة رسالة للمستخدم تطالب منه إدخال حرف .
- في السطر 16 يتم استخدام الدالة ()read لقراءة البايت Byte من لوحة المفاتيح.
- لاحظ أن الدالة ()read لها قيمة مرتجعه من نوع int ولذلك عرفنا متغير من نوع int اسمه i في سطر 7 وجعلنا له قيمة ابتدائية تساوى صفر وكما تعلم فإنك لابد أن تعطى قيمة ابتدائية لأى متغير في لغة الجافا Java.

- لاحظ أيضاً أن الدالة ()read من الممكن أن تسبب استثناء Exception من نوع IOException ولذلك استخدمنا try و catch لمعالجة هذا الاستثناء.
- في السطر 12 يتم استخدام الدوارة while لأننا نريد أن نقرأ بايت طالما أن الحرف المدخل ليس x.
- لاحظ أن البايت الذي سيدخلة المستخدم تم حفظة في المتغير i ولكن i من نوع int كما ذكرنا. إذن فإننا نريد أن نقارن i بالحرف x وذلك مستحيل لاختلاف نوعي المتغيرين ولذلك استخدمنا كلمة char لعمل تحويل Casting للمتغير i ليعامل كحرف وبذلك تصبح المقارنة ممكنة. انظر الشكل (2-17).

رابعاً: نتيجة التنفيذ:

بعد تنفيذ البرنامج فستظهر النتيجة على الشاشة كما في الشكل (2-17).

الشكل (2-17)

Wrapper Streams

- لاحظنا في المثال السابق أننا كنا نقرأ بايت Byte. وهذا نادراً ما يحدث فالطبيعي أنك تطلب إدخال عدة سطور من الأحرف.
- إذن الـ Wrapper class هي الفصيلة class التي تضيف بعض الوظائف إلى الفصائل classes الخاصة بالتدفق Streams.
- إذن فإننا سنستخدم فصيلة class تسمى BufferedReader والتي تحتوى على الدالة

readLine() التي تسمح لنا بقراءة عدة سطور. كما أنها تتميز بالقدرة على تخزين الحروف المدخلة بحيث يمكن القراءة منها مباشرة بدلاً من القراءة من نظام التشغيل وهذا يعطي سرعة أكبر.

- سوف نرى استخدام الفصيلة BufferedReader لاحقاً.

عمليات الملفات File Operations

عمليات القراءة والكتابة في الملفات تعتبر من أهم العمليات في البرمجة ولذلك توفر لغة الجافا Java اثنين من الفصائل classes لعملية القراءة والكتابة للملفات وهما:

1. FileReader: للقراءة من ملف.
2. FileWriter: للكتابة إلى ملف.

- ولكن المشكلة أنك تريد أن تقرأ سطراً من ملف ولذلك لا تستطيع استخدام الفصيلة FileReader مباشرة، إذن نريد فصيلة class لقراءة سطر من الملف وهذه الفصيلة class هي BufferedReader.
- ايضاً توجد نفس المشكلة في الكتابة إلى ملف فلا يمكنك استخدام الفصيلة FileWriter مباشرة لانك ستكتب سطر في الملف ولذلك نستخدم فصيلة class لكتابة سطر إلى ملف وهذه الفصيلة class هي PrintWriter.
- قد تبدو هذه السطور غامضة ولكننا سنعطي مثال صغير لبيان كيفية القراءة والكتابة بالنسبة للملفات.

أولاً: القراءة من ملف

```
File inputFile = new File ("data.txt");
FileReader fr = new FileReader (inputFile);
BufferedReader br = new BufferedReader (fr);
```

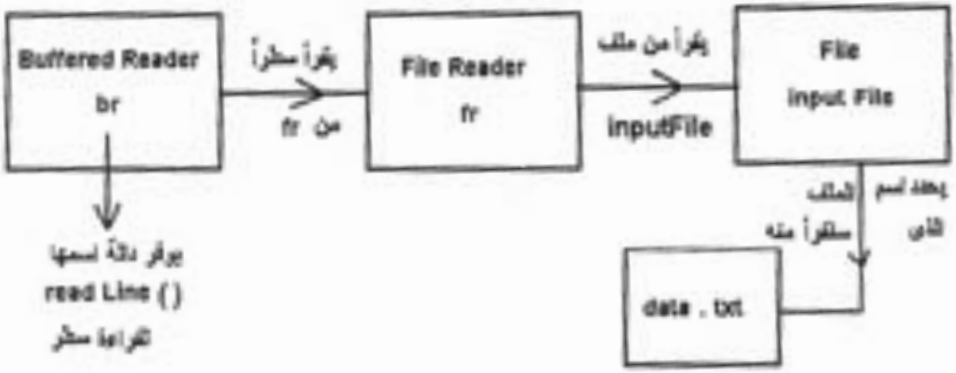
- في السطر الأول أنشأنا هدف object من نوع الفصيلة class المسماة File. إذن هنا أنشأنا ملفاً جديداً اسمه data.txt.
- حيث أنك تريد أن تقرأ من هذا الملف، فإنك تستخدم هدف object من نوع الفصيلة class المسماة FileReader. دالة البناء constructor لهذه الفصيلة class تستقبل هدف

object من نوع الفصيلة File. إذن هنا حددنا أننا سنقرأ من inputFile الذى بدورة هو عبارة عن الملف data.txt.

- كما ذكرنا لا يمكن استخدام FileReader مباشرة للقراءة من الملف وذلك لأنك تقرأ سطر من الملف وليس الملف كاملاً ولذلك استخدمنا هدف object من نوع الفصيلة class المسماة BufferedReader. ودالة البناء constructor فهذه الفصيلة class تستقبل هدف object من نوع FileReader لتحديد الملف الذى ستقرأ منه.
- كملخص: فإن السطر الثالث ينشئ br الذى يقرأ سطراً وسوف يقرأ هنا السطر من fr.
- أيضاً fr هو الذي يستخدم للقراءة من الملف وهذا الملف هو inputFile.
- أيضاً inputFile هو عبارة عن الملف الفعلى الذى ستقرأ منه وهو الذى اسمه data.txt.
- بعد هذه الثلاثة سطور فيمكننا القراءة مباشرة من br وكما ذكرنا فإن BufferedReader يوفر لنا دالة اسمها readLine() للقراءة منه. أى انه للقراءة من br فإننا نكتب. String

```
s = br.readLine();
```

- الرسم التالى يوضح هذه العملية.



سوف ترى فى المثال القادم كيف يتم ذلك.

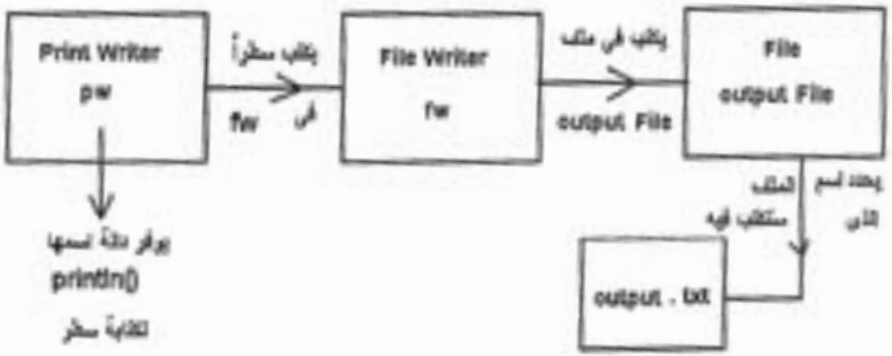
ثانياً: الكتابة إلى ملف:

```
File outputFile = new File ("output.txt");
FileWriter fw = new Filewriter (outputFile);
PrintWriter pw = new PrintWriter (fw);
```

- بالطبع يوجد تشابة شديد بين الكتابة والقراءة ولكننا سنعيد شرحهم لمزيد من الفهم.
- فى السطر الأول أنشأنا هدف object من نوع الفصيلة class المسماة File. إذن هنا أنشأنا ملفاً للكتابة فيه اسمه output.txt.
- حيث أنك تريد أن تكتب فى هذا الملف، فإنك تستخدم هدف object من نوع الفصيلة class المسماة FileWriter. دالة البناء constructor لهذه الفصيلة class تستقبل هدف object من نوع الفصيلة File. إذن هنا حددنا أننا ستكتب فى outputFile الذى بدورة هو عبارة عن الملف output.txt.
- كما ذكرنا لا يمكن استخدام FileWriter مباشرة للكتابة فى الملف وذلك لأنك ستكتب فيه سطر بسطر وليس السطور كاملة ولذلك استخدمنا هدف object من نوع الفصيلة class المسماة PrintWriter. ودالة البناء constructor هذه الفصيلة class تستقبل هدف object من نوع الفصيلة FileWriter لتحديد الملف الذى ستكتب فيه.
- كملخص: فإن السطر الثالث ينشىء pw الذى يكتب سطراً وسوف يكتب هذا السطر فى fw. أيضاً fw هو الذى يستخدم للكتابة فى الملف وهذا الملف هو outputFile أيضاً outputFile هو عبارة عن الملف الفعلى الذى سنكتب فيه وهو الذى اسمة output.txt .
- بعد هذه الثلاثة سطور فيمكننا الكتابة مباشرة فى pw وكما نعرف أنه يوجد الدالة المسماة println() التى تستخدم للكتابة. أى أنه للكتابة فى pw فإننا نكتب

```
String s = br.readLine ();
pw.println (s);
```

- أى أننا سنقرأ من br ونكتب فى pw هذه العبارة (s) .
- الرسم التالى يوضح هذه العملية.



الفصل الثامن عشر

مفهوم برمجة الشبكات
Network Programming

Programming Concepts في هذا الفصل

18

في هذا الفصل نتناول موضوع من الموضوعات الشائكة والمتقدمة فى عالم البرمجة وهو اعداد برامج تعمل من خلال الشبكات NetWork وذلك من خلال النقاط التالية :-

- ما المقصود ببرمجة الشبكات NetWork Programming
- استعمال connections فى اعداد تطبيق خادم Server
- استعمال connections فى اعداد تطبيق مرسل Client
- استعمال DataGrams فى اعداد تطبيق خادم Server
- استعمال DataGrams فى اعداد تطبيق مرسل Client
- اعداد تطبيق تبادل الرسائل Chating

-اعداد تطبيق الخادم Server

-اعداد تطبيق مرسل Client

ما هى الشبكات؟: Networks

غرض الشبكات أساساً هو أن تتمكن جميع أجهزة الكمبيوتر من الاتصال ببعضها البعض على مسافات قريبة أو بعيدة. ما نحتاجه هو كارت شبكات وكابلات تصل بين الأجهزة واستخدام بروتوكول مثل TCP/IP.

ما المقصود ببرمجة الشبكات NetWork Programming

المقصود بذلك اعداد تطبيقات تعمل من خلال الشبكة واشهرها هو تطبيقات تبادل الرسائل Chating المشهورة وامكانية إرسال الملفات من جهاز إلى جهاز آخر.

وفى هذا الفصل نتناول فكرة اعداد تطبيقات للشبكات NetWork Programming بأكثر من طريقة وفيها نقوم باعداد نوعين من التطبيقات

الأول: تطبيق خادم Server وهو التطبيق الذى يستقبل الاتصال من الجهاز المرسل كما يستقبل منه الرسائل.

الثانى: تطبيق مرسل Client وهو التطبيق الذى يقوم بالاتصال بالجهاز الخادم Server ويرسل و يستقبل منه الرسائل.

أصبح العالم أشبة بقرية صغيرة يمكن من أي مكان في العالم أن تعرف أي أخبار تريدها أو أن تتبادل المعلومات مع أي شخص في العالم. يرجع الفضل في ذلك إلى علم الشبكات Networking والذي أمكن من خلاله وصل العالم كله. على الرغم من أن علم الشبكات Networking علم منفصل وقائم بذاته، إلا أن هناك حلقة وصل بين علم البرمجة Programming وعلم الشبكات Networking لأنه يمكنك استخدام الشبكة في بعض تطبيقات البرمجة مثل برامج الدردشة Chatting والتي تضم علمي البرمجة والشبكات في نفس الوقت.

في هذا الفصل سنبدأ إلقاء الضوء على بعض المفاهيم الخاصة بالشبكات حتى تكون معتاداً عليها وسنصل في آخر الفصل إلى بناء برنامج دردشة Chatting.

البروتوكول: Protocols

تخيل شخص أجنبي يتحدث إليك فإنك بالطبع لن تفهم منه شيئاً إلا إذا كنت تتحدث بلغته. تخيل أيضاً وجود مجموعة من الأشخاص يتحدثون بلغات مختلفة فهل يمكن التحدث معهم؟ بالطبع لا. نفس هذا الكلام بالنسبة لأجهزة الكمبيوتر فإذا لم يكن هناك مترجم ونظام يتيح للأجهزة أن تتصل ببعضها البعض وتتبادل المعلومات فإن تبادل المعلومات ببعضهم البعض يكون مستحيلاً.

ما نحتاجه هو بعض القواعد التي تتبعها الأنظمة للاتصال على شبكة. إذن البروتوكول هو القواعد المتعارف عليها بين الأنظمة المختلفة لكي تتمكن من الاتصال وتبادل المعلومات بينها.

أحد أشهر البروتوكولات هو Transmission Control Protocol Internet Protocol (TCP/IP)

وهو يتعامل مع أي نظام من أنظمة التشغيل. إذن يمكن اتصال الأجهزة بصرف النظر عن أن نظام التشغيل كان Windows, Dos, Macintosh, Unix, Linux.

ولكن تخيل وجود أكثر من جهاز كمبيوتر في الشبكة. الطبيعي أنك تريد أن تتصل بجهاز معين لتتبادل معه المعلومات. إذن كيف يمكن تحديد جهاز معين للاتصال معه في الشبكة؟ لابد أن يكون لكل جهاز عنوان خاص به لكي تتمكن من الوصول إليه. هذا العنوان يسمى IP.

عناوين الإنترنت IP addresses

كما ذكرنا يستخدم عنوان الإنترنت IP address كعنوان لجهاز الكمبيوتر في الشبكة حتى تتمكن من الوصول لجهاز معين.

يتكون عنوان الإنترنت IP address من 4 أرقام يفصل بينهم نقطة، كل رقم من الأربعة تتراوح قيمته بين صفر و255 ولذلك يسمى عنوان الإنترنت IP Address باسم Dotted quads.

مثال لعنوان الإنترنت IP address هو 130.10.32.100

لكى تعرف عنوان الإنترنت IP address لجهازك

- ☐ فى أنظمة Windows 98 و Windows NT فستجد إلى شاشة سطح المكتب أيقونة للشبكات اسمها My Network Places. انقر بالزر الأيمن بالفأرة على هذه الأيقونة واختر Properties. انقر على Protocols ثم انقر مرتين على (Internet Protocol TCP/IP) فإنك ستجد عنوان الإنترنت IP address لجهازك. بالطبع لابد أن يكون عندك كارت شبكات.
- ☐ فى أنظمة Windows 2000 فانقر باليمين على My Network Places ثم اختر Properties. ثم انقر باليمين على Local Area Connection ثم اختر Properties ثم انقر مرتين على TCP/IP Internet Protocol . ستجد عنوان الإنترنت IP address الخاص بجهازك.
- ☐ طريقة أخرى سهلة لجميع الأنظمة: افتح شاشة Dos ثم اكتب الأمر التالى ipconfig فإنك ستعرف عنوان الإنترنت IP address لجهازك.

عنوان الإنترنت IP address يحدد شيئين هما: عنوان الشبكة وعنوان الجهاز على هذه الشبكة إذن كيف نعرف كم رقم من الأربع أرقام الخاصين بعنوان الإنترنت، يحدد عنوان الشبكة وكم رقم يحدد عنوان الجهاز على الشبكة؟ نعرف ذلك عن طريق حجاب الشبكة الفرعية Subnet Mask.

حجاب الشبكة الفرعية Subnet Mask

يستخدم حجاب الشبكة الفرعية Subnet Mask لتحديد عدد الأرقام التى تحدد عنوان الشبكة من الأربع أرقام الخاصين بعنوان الإنترنت IP address وبالطبع باقى الأرقام فى عنوان الإنترنت IP address تحدد عنوان الجهاز على الشبكة. وضعت فئات لحجاب الشبكة الفرعية Subnet Mask على حسب عدد الأجهزة التى يمكن وضعها في الشبكة.

Class الفئة	IP address from عنوان الإنترنت يبدأ من	IP address to عنوان الإنترنت إلى	Subnet Mask حجاب الشبكة الفرعية	No. of Computers
A	0.0.0.0	127.255.255.255	255.0.0.0	256*256*256 ≅ 17 million
B	128.0.0.0	191.255.255.255	255.255.0.0	256*256 ≅ 65.536
C	192.0.0.0	223.255.255.255	255.255.255.0	254
D	224.0.0.0	239.255.255.255	Not Available	Not Available
E	240.0.0.0	247.255.255.255	Not Available	Not Available

مثال :

إذا كان جهازك له عنوان الإنترنت IP address (64.32.1.3) إذن فهو واقع ضمن Class A لأن الرقم الأول (64) يقع بين 0 و 128 إذن إذا نظرنا لحجاب الشبكة الفرعية Subnet Mask نجدة 255.0.0.0 إذن الرقم الأول (64) هو الذى يحدد عنوان الشبكة وبالطبع باقى الأرقام (32.1.3) تحدد عنوان الجهاز فى هذه الشبكة.

إذا كان جهازك له عنوان إنترنت IP address (130.10.32.100) إذن فهو واقع ضمن Class B لأن الرقم الأول (130) يقع بين 192,128 إذن إذا نظرنا إلى حجاب الشبكة الفرعية Subnet Mask نجده 255.255.0.0 أى أن أول رقمين (130.10) يحددان عنوان الشبكة وبالطبع (32.100) يحددان عنوان الجهاز فى هذه الشبكة.

فى Class A مثلاً نجد أن هناك 3 أرقام تحدد عنوان الجهاز وحيث أن كل رقم يقبل رقماً من صفر إلى 255 أى 256 رقم، إذن عدد الأجهزة الكلى الذى نستطيع توصيله فى شبكة من نوع Class A هو 256*256*256 أى تقريباً 17 million جهاز فى الشبكة الواحدة وبالطبع هذا الرقم كبير جداً ولذلك تستخدمه الشركات الكبيرة جداً فقط.

ونجد أن Class D,E محجوزان للتجارب المستقبلية وغير مستخدمين حالياً.

Class الفئة	Used for تستخدم فى
A	الشركات الكبيرة
B	الشركات المتوسطة
C	الشركات الصغيرة
D	غير مستخدم
E	غير مستخدم

- لا يمكن أن يتكرر عنوان الإنترنت IP address أبداً على مستوى العالم كله لأنه لو تكرر فذلك يعنى أن هناك أكثر من جهاز له نفس العنوان وذلك مستحيل.
- لا يمكن أن يكون لجهازك عنوان إنترنت IP address كلة أصفار أو كله 256 ولذلك

نجد أن عدد الأجهزة الكلي ينخفض جهازين ولذلك نجد أن Class C نستطيع توصيل 254 جهاز فقط بدلاً من 256.

- ولكن معظم الناس لا يفضلون الأرقام في التعامل مع عناوين الإنترنت فإنهم يفضلون الأسماء لأن حفظها أسهل من الأرقام ولذلك لابد من وجود بديل لعنوان الإنترنت IP address نستطيع من خلاله التعامل مع عناوين الإنترنت كنص من الحروف وليس أرقام و يتم ذلك عن طريق ما يعرف باسم (Domain Name System DNS).

DNS:

هو نظام يقوم بتحويل الاسم إلى عنوان الإنترنت IP address المناظر. إذن فإننا إذا فتحنا مستعرض الإنترنت Internet Browser مثل Internet Explorer وكتبت العنوان التالي 192.18.97.137 فإنه يكون تماماً مثل www.javasoft.com.

إذن الأسهل هو حفظ الاسم بدلاً من الأرقام إذن نظام DNS يحول الاسم إلى الـ IP address المناظر.

لاحظ أن استخدام الاثنين سواء عنوان الإنترنت بالأرقام IP Address أو بالاسم متماثل تماماً ولا فارق بينهما إلا أن استخدام الأسماء وحفظها اسهل من الأرقام كثيراً.

الآن عرفنا كيف نحدد جهاز بعينه في الشبكة ولكن المهم كيف يتم نقل البيانات؟ لابد أن نذكر اولاً ما هو الـ Port.

Port

- عادة يكون هناك كارت شبكة واحد في الكمبيوتر ولذلك يوجد توصيل واحد لكل كمبيوتر بالشبكة. جميع المعلومات والبيانات تصل إلى الكمبيوتر عن طريق هذا التوصيل الوحيد ولكن بعض المعلومات قد تكون مخصصة لتطبيقات معينة في الكمبيوتر.
- إذن كيف يستطيع الكمبيوتر معرفة التطبيق لتوصيل البيانات له؟ ذلك يتم عن طريق الـ Port.
- إذن الـ Port هو عبارة عن خدمة معينة لها رقم والبروتوكول المسمى TCP/IP يستخدم الـ Port لتوصيل البيانات الواردة إلى تطبيق معين في الكمبيوتر.

- الـ Port إذن هو رقم int يتراوح بين 0 و 65535 . ولكن من 0 إلى 1023 هى أرقام محجوزة لخدمات معينة معروفة أما الباقى فهو يستخدم للتجارب فمثلاً الـ Port رقم 21 يستخدم لنقل الملفات (FTP) والPort رقم 80 يستخدم لنقل صفحات الوب (HTTP).
- إذا أردت معرفة الخدمات Services ورقم الـ Port المناظر لها فابحث عن ملف اسمه Services بلا أى امتداد فى جهازك. افتح هذا الملف باستخدام Notepad.

أى أنك ستذهب إلى Start, Find.

Files or Folders

ثم اكتب اسم Services ستجد أكثر من نتيجة. ما يهمنا هو الملف المسمى Services بلا أى امتداد. افتح هذا الملف بالضغط مرتين عليه. سيطلب منك التطبيق الذى تريد فتح الملف به.

أختر Notepad واستعرض محتويات الملف لتعرف رقم الـ Port الخاص بكل خدمة، لاحظ أن هذا الملف لا يحتوى على جميع الخدمات ولكنه يعطيك فكرة عن الخدمات Services الموجودة.

المقابس Sockets

يتكون المقبس Socket من جزئين:

- عنوان الإنترنت IP address
- رقم الـ Port

مثلاً:إذا أردت أن تتنقل لموقع Java soft على الوب فإنك تطلب نقل صفحة الوب الخاصة بشركة Java soft اى أنك تستخدم Port رقم 80 وبالطبع لابد أن تستخدم عنوان الإنترنت IP address أو اسم الموقع Domain Name أى أنك تكتب.

http://www.javasoft.com:80

إذا لم تحدد رقم الـ Port فإن مستعرض الإنترنت يستخدم Port رقم 80 ولكنك إذا أردت استخدم Port أخر فإنك تحدد رقمة بشرط أن يكون الموقع يقدم هذه الخدمة التى تريدها.

أهم الفصائل Classes المستخدمة فى الشبكات

Class الفصيلة	وظيفته	Methods الدوال الموجودة في الفصيلة
InetAddress	نستخدمه لمعرفة عنوان الإنترنت IP address لشبكة معينة.	getLocalHost() تقوم بإرجاع هدف object من نوع الفصيلة InetAddress يمثل الجهاز الذى نستخدمه. getAddress(): تقوم بإرجاع مصفوفة Array تتكون من 4 أرقام من نوع byte تمثل عنوان الإنترنت IP address. getHostName(): تقوم بإرجاع String تمثل اسم الجهاز.
Socket	هو المستخدم من جهاز المستخدم Client لتحديد اتصال بينه والجهاز الخادم Server عن طريق تحديد عنوان الشبكة IP address والـ Port للجهاز الخادم Server.	getInputStream(): تقوم بمعرفة البيانات الداخلة إلى جهاز المستخدم Client. getOutputStream(): تقوم بمعرفة البيانات الخارجة من جهاز المستخدم Client. close(): تغلق الاتصال مع اى جهاز
ServerSocket	يستخدمه الجهاز الخادم Server ويستمع لكل مستخدم Client يحاول إجراء اتصال معه.	accept(): تنتظر إجراء اتصال بين الجهاز الخادم Server وجهاز المستخدم Client

فى لغة VB.NET

استعمال connections فى اعداد تطبيق خادم Server

الخطوة الاولى كما ذكرنا هى اعداد تطبيق الخادم Server الذى ينتظر اتصال تطبيقات العملاء Clients وينتظر منها الرسائل وفى هذه الفقرة نستعمل الطريقة الأولى وهى استعمال الاتصال المباشر Connection بينهما ويمر ذلك بخمسة خطوات هى:

1- تعريف متغير Object من الفصيلة TcpListener

كما فى السطور التالية

```
1. Dim listener As TcpListener
2. listener = New TcpListener(5000)
```

فى لغة C#

```
1. Dim listener As TcpListener
2. listener = New TcpListener(5000)
```

فى هذه السطور

- فى السطر رقم 1 تم الإعلان عن المتغير listener (منصت) من الفصيلة TcpListener
- فى السطر رقم 2 تم تخليق هدف الفصيلة TcpListener باستعمال الكلمة New
- والقيمة 500 هى رقم المنفذ Port الذى يتم الاتصال من خلاله

2- تشغيل الـ listener وذلك باستدعاء الدالة Start() كما فى السطر التالى

```
listener.Start()
```

فى لغة C#

```
listener.Start()
```

3- تعريف متغير هدف من الفصيلة Socket واستعماله فى الموافقة على عملية اتصال تطبيق العميل ويتضح ذلك من السطور التالية Clent.

```
1. Private connection As Socket
2. connection = listener.AcceptSocket()
```

فى لغة C#

```
3. Private Socket connection;
4. connection = listener.AcceptSocket()
```

- فى السطر رقم 1 تم الإعلان عن متغير هدف بالاسم connection من الفصيلة Socket.
- فى السطر رقم 2 تم استدعاء الدالة AcceptSocket() مع الهدف listener والتى تقبل اتصال تطبيق العميل والإشارة إلى الاتصال بينهما بمتغير الهدف Client Connection

4- قراءة النصوص المرسلة من الـ Client كما فى السطور التالية:

```
1. Private reader As BinaryReader
2. reader = New BinaryReader(socketStream)
3. theReply = reader.ReadString()
```

فى لغة C#

```
1. Private BinaryReader reader ;
2. reader = New BinaryReader(socketStream);
3. theReply = reader.ReadString()
```

فى هذه السطور:

- فى السطر رقم 1 تم الإعلان عن متغير هدف بالاسم reader من نوع الفصيلة BinaryReader التى توفر إمكانيات قراءة بيانات ثنائية وسوف يستعمل هذا المتغير فى قراء البيانات المرسلة.
- فى السطر رقم 2 يتم إنشاء الهدف reader مع إرسال الهدف socketStream كمعامل له لتتم القراءة منه.
- فى السطر رقم 3 يتم استدعاء الدالة ReadString() مع المتغير Reader الذى تم تعريفه فتتم القراءة.

5- الخطوة الأخيرة هى إغلاق الاتصال بين الجهاز المرسل Client والجهاز المستقبل Server كما فى السطور التالية:

```
1. reader.Close()
2. socketStream.Close()
3. connection.Close()
```

فى لغة C#

```
1. reader.Close()
2. socketStream.Close()
3. connection.Close()
```

فى هذه السطور:

- تم اخلاق متغير هدف القراءة reader ومتغير socketStream ومتغير الوصلة connection.
- وبعد عرض الخطوات المتبعة لإعداد تطبيق مستقبل Server تعال بنا نطبق معا هذه الخطوات لبناء التطبيق.

مثال

هذا المثال يوفر تطبيق مستقبل Server عند تنفيذه يعمل التطبيق وينتظر الاتصال به ولإعداده تابع الخطوات التالية:

1- قم بإعداد تطبيق Windows Application جديد.

2- قم بتصميم شكل النماذج الرئيسى Form لتصبح كما فى الشكل (1-18).

الشكل (1-18)

3- قم بتغيير اسم مربع النص الكبير الذى يظهر نتائج الاتصال وكذلك الرسائل المرسلة إلى الاسم Display وتغيير اسم مربع النص الصغير المستعمل فى كتابة الرسائل من تطبيق الـ Server إلى تطبيق Client إلى الاسم Input.

4- ادخل إلى صفحة الأوامر الخاصة بالـ Form1 وقبل تعريف فصيلة (Public Form1 اكتب السطور التالية: Class Form)

```
Imports System.Windows.Forms
Imports System.Threading
Imports System.Net.Sockets
Imports System.IO
```

فى لغة C#

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Threading;
using System.Net.Sockets;
using System.IO;
```

فى هذه السطور:

- تم استعمال الأمر imports لفتح (للإشارة إلى) المكتبات NameSpaces التى سوف نحتاجها فى البرنامج والمعرف بها الفصائل مثل System.Threading المعرف بها الفصيلة Thread وكذلك System.Net.Sockets المعرف بها فصائل التعامل مع الشبكات.

5- قم بتعريف مجموعة المتغيرات التالية داخل فصيلة Form1 كما فى السطور التالية:

```
Public Class Form1
Inherits Form
1. Private connection As Socket
2. Private readThread As Thread ' server thread
3. Private socketStream As NetworkStream
4. Private writer As BinaryWriter
5. Private reader As BinaryReader
```

فى لغة C#

```
public class Server : System.Windows.Forms.Form
 private Socket connection;
private Thread readThread;
private NetworkStream socketStream;
private BinaryWriter writer;
private BinaryReader reader
```

فى هذه السطور:

يتم تعريف مجموعة المتغيرات التى أشرنا اليها فى الخطوات السابقة.

6- قم بتعديل الدالة Sub New() لتصبح كما فى السطور التالية:

```
Public Sub New()
    MyBase.New()
    InitializeComponent()
1.  readThread = New Thread(AddressOf RunServer)
2.  readThread.Start()
End Sub ' New
```

فى لغة C#

```
public Server()
      {
         InitialiseComponent();
        readThread = new Thread( new ThreadStart( RunServer ) );
         readThread.Start();
      }
```

فى هذه السطور:

- فى السطر رقم 1 تم إنشاء الهدف readThread مع إرسال اسم الدالة التى تنفذ معه إلى دالة البناء الخاصة به لتنفذ وهى الدالة RunServer()
- فى السطر رقم 2 يتم استدعاء الدالة Start() مع الهدف readThread التى تؤدى لى بدء تنفيذ الـ Thread

7- قم بكتابة الدالة RunServer() التى تتولى عمل تطبيق الـ Server وهو الاتصال واستقبال الرسائل من تطبيق الـ Client كما فى السطور التالية:

```
Public Sub RunServer()
1.  Dim listener As TcpListener
2.  Dim counter As Integer = 1
3.  Try
4.  listener = New TcpListener(5000)
5.  listener.Start()
6.  While True
7.  Display.Text = "Waiting for connection" & vbCrLf
8.  connection = listener.AcceptSocket()
9.  socketStream = New NetworkStream(connection)
10. writer = New BinaryWriter(socketStream)
11. reader = New BinaryReader(socketStream)
```

```
12. Display.Text &= "Connection " & counter & _                " received." & vbCrLf
13. writer.Write("SERVER>>> Connection successful")
14. Input.ReadOnly = False
15. Dim theReply As String = ""
16. Try
17. Do
18. theReply = reader.ReadString()
19. Display.Text &= vbCrLf & theReply
20. Loop While (theReply <> CLIENT>>>TERMINATE" _              AndAlso
    connection.Connected)
21. Catch inputOutputException As IOException
22. MessageBox.Show("Client application closing")
23. Finally
24. Display.Text &= vbCrLf & "User terminated connection"
25. Input.ReadOnly = True
26. writer.Close()
27. reader.Close()
28. socketStream.Close()
29. connection.Close()
30. counter += 1
31. End Try
32. End While
33. Catch inputOutputException As IOException
34. MessageBox.Show("Server application closing")
35. End Try
36. End Sub ' RunServer
```

فى لغة C#

```
37. public void RunServer()
38. {
39. TcpListener listener;
40. int counter = 1;
41.
42. try
43. {
44. listener = new TcpListener( 5000 );
45. listener.Start();
46. while ( true )
47. {
48. displayTextBox.Text = "Waiting for connection\r\n";
49. connection = listener.AcceptSocket();
50. socketStream = new NetworkStream( connection );
51. writer = new BinaryWriter( socketStream );
52. reader = new BinaryReader( socketStream );
53. displayTextBox.Text += "Connection " + counter +
54. " received.\r\n";
55. writer.Write( "SERVER>>> Connection successful" );
```

```
56.
57. inputTextBox.ReadOnly = false;
58. string theReply = "";
59. do
60. {
61. try
62. {
63. theReply = reader.ReadString();
64. displayTextBox.Text += "\r\n" + theReply;
65. }
66. catch ( Exception )
67. {
68. break;
69. }
70.
71. } while ( theReply != "CLIENT>>> TERMINATE"  &&
72. connection.Connected );
73.
74. displayTextBox.Text +=
75. "\r\nUser terminated connection";
76. inputTextBox.ReadOnly = true;
77. writer.Close();
78. reader.Close();
79. socketStream.Close();
80. connection.Close();
81.
82. ++counter;
83. }
84. } // end try
85.
86. catch ( Exception error )
87. {
88. MessageBox.Show( error.ToString() );
89. }
90.
91.}
```

- فى هذه السطور يتم تعريف الدالة RunServer() التى تتولى استقبال محاولات الاتصال وتحقيق الاتصال بالـ Clients وكذلك إرسال البيانات مرة أخرى إلى الـ Client كما يلى:
- فى السطر رقم 1 تم الإعلان عن متغير من الفصيلة TcpListener كما أشرنا فى الفقرات السابقة
- فى السطر رقم 4 يتم إنشاء الهدف من الفصيلة TcpListener مع تحديد المنفذ Port رقم 5000 للاتصال.

- فى السطر رقم 5 يتم استدعاء الدالة Start() مع الهدف Listener لبداية انتظار الاتصال.
- فى السطر رقم 8 يتم استدعاء الدالة AcceptSocket() مع الهدف listener الذى تم انشاءه والإشارة إلى ذلك بالمتغير connection.
- فى السطور رقم 9و10و11 يتم إنشاء الأهداف socketStream وwriter وreader المستعملة الاتصال والقراءة والكتابة ثم يتم طباعة رسائل على الشاشة تفيد تحقق الاتصال.
- فى السطور 17،18،19و20 يوجد تكرار Loop While... DO يقوم بتكرار القراءة طالما يتحقق الاتصال ولم يكتب المستخدم الكلمة TERMINATE
- وهذا هو قلب هذه الدالة وهو الاستمرار فى القراءة حتى يتم إغلاق الاتصال

وفى باقى السطور يتم التعامل مع الأخطاء وإغلاق ما تم فتحه.

8- قم بإضافة حدث Key_Down لمربع النص Input واكتب به السطور التالية:

```
Private Sub Input_KeyDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles Input.KeyDown
1.  Try
2.  If (e.KeyCode = Keys.Enter AndAlso Not connection Is Nothing) Then
3.  writer.Write("SERVER>>> " & Input.Text) '
4.  Display.Text &= vbCrLf & "SERVER>>> " & Input.Text
5.  If Input.Text = "TERMINATE" Then
6.  connection.Close()
7.  End If
8.  Input.Clear()
9.  End If
10. Catch exception As SocketException
11. Display.Text &= vbCrLf & "Error writing object"
12. End Try
13.  End Sub '
```

فى لغة C#

```
92. protected void inputTextBox_KeyDown(
93. object sender, KeyEventArgs e )
94. {
95. try
96. {
97. if ( e.KeyCode == Keys.Enter && connection != null )
98. {
99. writer.Write( "SERVER>>> " + inputTextBox.Text );
100.
101.    displayTextBox.Text +=
```

```
102.    "\r\nSERVER>>> " + inputTextBox.Text;
103.    if ( inputTextBox.Text == "TERMINATE" )
104.    connection.Close();
105.
106.    inputTextBox.Clear();
107.    }
108.    }
109.    catch ( SocketException )
110.    {
111.    displayTextBox.Text += "\nError writing object";
112.    }
113.    }
```

- في هذه السطور يتم أخذ ما يكتبه المستخدم في الـ TextBox ثم التأكد من الزر المضغوط إذا كان Enter يتم إرسال ما كتب إلى تطبيق الـ Client كما يلي.
- في السطر رقم 2 يتم اختبار ما كتبه المستخدم اذا كان الزر Enter وأيضا مازال هناك اتصال.
- في السطر رقم 3 يتم إرسال ما كتبه المستخدم في مربع النص بالإضافة إلى الكلمة Server للإشارة إلى ان مصدر البيانات هي الـ Server.
- في السطر رقم 5 يتم اختبار ما كتب إذا كان الكلمة TERMINATE يتم إغلاق الاتصال وباقي السطور كما سبق.

9- نفذ البرنامج تحصل على نتيجة التنفيذ كما في الشكل (2-18).

الشكل (2-18)

- في هذا الشكل يبدأ البرنامج عارضا رسالة تفيد الانتظار حتى يتم الاتصال.

استعمال connections فى اعداد تطبيق مرسل Client

فى هذه الفقرة نقوم باعداد تطبيق العميل Client الذى يقوم بالاتصال وإرسال الرسائل وذلك من خلال خطوات تشابه الخطوات السابقة كما يلى:

1- قم باعداد تطبيق Windows Application جديد.

2- قم بتصميم شكل النماذج الرئيسى Form لتصبح كما فى الشكل (3-18).

الشكل (3-18)

- قم بتغيير اسم مربع النص الكبير الذى يظهر نتائج الاتصال وكذلك الرسائل المرسلة إلى الاسم Display وتغيير اسم مربع النص الصغير المستعمل فى كتابة الرسائل من تطبيق الـ Server إلى تطبيق Client إلى الاسم Input.

4- أدخل إلى صفحة الأوامر الخاصة بالـ Form1 وقبل تعريف فصيلة (Public Class Form1 اكتب السطور التالية:

```
Imports System.Windows.Forms
Imports System.Threading
Imports System.Net.Sockets
Imports System.IO
```

فى لغة C#

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Threading;
using System.Net.Sockets;
using System.IO;
```

5- قم بتعريف مجموعة المتغيرات التالية داخل فصيلة Form1 كما فى السطور التالية:

```
Private output As NetworkStream
Private writer As BinaryWriter
Private reader As BinaryReader
Private message As String = "" ' message sent to server
Private readThread As Thread
```

فى لغة C#

```
        private NetworkStream output;
        private BinaryWriter writer;
        private BinaryReader reader;
        private string message = "";
```

6- قم بتعديل الدالة Sub New() لتصبح كما فى السطور التالية:

```
Public Sub New()
    MyBase.New()
    InitializeComponent()
    readThread = New Thread(AddressOf  RunClient)
    readThread.Start()
End Sub ' New
```

فى لغة C#

```
public Client()
        {
            InitializeComponent();
       readThread = new Thread( new ThreadStart( RunClient ) );
            readThread.Start();
}
```

7- قم بكتابة الدالة RunClient () التى تتولى عمل تطبيق الـ Client وهو الاتصال كما فى السطور التالية:

```
Public Sub RunClient()
1.  Dim client As TcpClient
2.  Try
3.  Display.Text &= "Attempting connection" & vbCrLf
4.  client = New TcpClient()
5.  client.Connect("localhost", 5000)
6.  output = client.GetStream()
7.  writer = New BinaryWriter(output)
8.  reader = New BinaryReader(output)
9.  Display.Text &= vbCrLf & "Got I/O streams" & vbCrLf
10. Input.ReadOnly = False
11. Try
12. Do
13. message = reader.ReadString
14. Display.Text &= vbCrLf & message
15. Loop While message <> "SERVER>>> TERMINATE"
16. Catch inputOutputException As IOException
17. MessageBox.Show("Client application closing")
18. Finally
19.  Display.Text &= vbCrLf & "Closing connection." & _vbCrLf
20. writer.Close()
21. reader.Close()
22. output.Close()
23. client.Close()
24. End Try
25. Application.Exit()
26. Catch inputOutputException As Exception
27. MessageBox.Show("Client application closing")
    End Try
End Sub ' RunClient
```

فى لغة C#

```
public void RunClient()
{
TcpClient client;
try
{
displayTextBox.Text += "Attempting connection\r\n";
client = new TcpClient();
client.Connect( "localhost", 5000 );
output = client.GetStream();
writer = new BinaryWriter( output );
reader = new BinaryReader( output );
displayTextBox.Text += "\r\nGot I/O streams\r\n";
inputTextBox.ReadOnly = false;
```

```
do
{

try
{
message = reader.ReadString();
displayTextBox.Text += "\r\n" + message;
}
catch ( Exception )
{
System.Environment.Exit(
System.Environment.ExitCode );
}
} while( message != "SERVER>>> TERMINATE" );

displayTextBox.Text += "\r\nClosing connection.\r\n";
writer.Close();
reader.Close();
output.Close();
client.Close();
Application.Exit();
}
catch ( Exception error )
{
MessageBox.Show( error.ToString() );
}

}
```

فى هذه السطور يتم تصميم الدالة الرئيسية للبرنامج والتى تنفذ داخل الـ Thread والتى تتولى الاتصال واستمرار القراءة طالما مازال يوجد اتصال ولم يكتب تطبيق الـ Server الكلمة TERMINATE كما سبق فى البرنامج السابق.

8- قم باضافة حدث Key_Down لمربع النص Input واكتب به السطور التالية:

```
Private Sub Input_KeyDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles Input.KeyDown
Try
If e.KeyCode = Keys.Enter Then
writer.Write("CLIENT>>> " & Input.Text)
Display.Text &= vbCrLf & "CLIENT>>> " & _
Input.Text
Input.Clear()
End If
      Catch exception As SocketException
```

```
        Display.Text &= vbCrLf & "Error writing object"
    End Try
End Sub ' Input_KeyDown
```

فى لغة C#

```
protected void inputTextBox_KeyDown (
object sender, KeyEventArgs e )
{
try
{
if ( e.KeyCode == Keys.Enter )
{
writer.Write( "CLIENT>>> " + inputTextBox.Text );
displayTextBox.Text +=
"\r\nCLIENT>>> " + inputTextBox.Text;
inputTextBox.Clear();
}
}
catch ( SocketException )
{
displayTextBox.Text += "\nError writing object";
}

} // end method inputTextBox_KeyDown
```

ق هذه السطور:

يتم اختبار ما يكتبه المستخدم ق مربع النص Input فاذا كان الزر المضغوط هو Enter يتم استعمال الهدف Writer واستدعاء الدالة Write لإرسال ما كتب يسبقه الكلمة Client.

9- نفذ برنامج ال Server و نفذ برنامج ال Client تحصل عل نتيجة التنفيذ كما ق الشكل (4-18).

الشكل(4-18)

10- جرب كتابة رسائل بمربع نص الكتابة Input ببرنامج الـ Client ثم الضغط على الزر Enter تلاحظ إرسال الرسالة إلى تطبيق الـ Server كما فى الشكل (5-18).

الشكل (5-18)

فى لغة Java

الحزم Packages المستخدمة في الشبكات

الحزم Packages اللازمة فى الشبكات تستدعيهم عن طريق import ونستخدم هنا عدد 2 Packages للشبكات وهما:

```
import java.net.*;
import java.io.*;
```

إنشاء مقبس جهاز المستخدم Building Client Socket

لابد من إنشاء هدف object من نوع الفصيلة Socket وكما ذكرنا فإننا لابد أن نحدد عنوان الإنترنت IP address الخاص بالجهاز الخادم Server وايضاً رقم الـ Port.

مثلاً:

```
Socket clientSocket = new Socket ("130.10.32.100",8000);
```

حيث القيمة الأولى لدالة البناء constructor تحدد عنوان الإنترنت IP address الخاص بالجهاز الخادم Server وتحدد القيمة الثانية رقم الـ Port.

لاحظ استخدام رقم 8000 يمكنك بالطبع استخدام أى رقم أكبر من 1023 لأنه كما

ذكرنا فإن الأرقام من 0 إلى 1023 محجوزة ولذلك فإننا سنستخدم رقم 8000 والذي بالطبع أكبر من 1023.

القراءة والكتابة من وإلى المقبس Socket

- للكتابة إلى المقبس Socket نستخدم هدف object من نوع الفصيلة class المسماة PrintStream كالآتي:

```
PrintStream out = new PrintStream (clientSocket.getOutputStream ());
```

- لاحظ أن clientSocket هو هدف object من نوع الفصيلة class المسماة Socket كما أنشأناه في النقطة السابقة.
- أما للقراءة من المقبس Socket فإننا نستخدم هدف object من نوع الفصيلة class المسماة BufferedReader كالآتي:

```
DataInputStream in = new DataInputStream (clientSocket.getInputStream ());
```

- الدوال ()getOutputStream,()getInputStream الموجودتان في الفصيلة class المسماة Socket تستطيع استخدامهما للاتصال بين جهاز المستخدم Client والجهاز الخادم Server حيث أن الدالة ()getOutputStream تستخدم للكتابة إلى المقبس Socket والدالة ()getInputStream تقوم بالقراءة من المقبس Socket.

إغلاق أو إنهاء الاتصال

نستخدم الدالة ()close كما ذكرنا كالآتي

```
out.close ();
in.close ();
```

بناء مقبس الجهاز الخادم Building Server Socket

لابد من إنشاء هدف object من نوع الفصيلة ServerSocket وكما ذكرنا فإن بالنسبة للجهاز الخادم Server فإننا نحدد رقم الـ Port فقط الذي يجب أن يكون متفقاً تماماً مع رقم الـ Port لجهاز المستخدم Client.

السؤال الآن: لماذا نحدد عنوان الإنترنت IP address للجهاز الخادم Server ولا نحدد عنوان الإنترنت IP address لجهاز المستخدم Client.

الإجابة: تخيل أنك مدرس ولديك طلبة. الطلبة يسألونك لأنك مدرس فالكل يعرفك ويعرف اسمك ويستطيع أن يجدك في المدرسة. ولكنك لا تعرف أسماء كل الطلبة عندك ولكن من يسألك سوف تجيبه أيضاً تستطيع القول بأن نظام المستخدم/ الخادم Client/Server يماثل هذا الموقف فالجهاز الخادم Server هو المدرس والطلبة هم المستخدم Client والمدرسة هي الـ Port فلابد أن يكون الـ Port واحد بين المستخدم Client والخادم Server كما لابد أن يلتقي المدرس والطلبة في نفس المدرسة فلا يمكن أن ينتظر المدرس الطلبة في مدرسة ويذهب الطلبة إلى مدرسة أخرى، وأيضاً الجهاز الخادم Server يستقبل طلبات المستخدم Client ويجيب عنها إذن أجهزة المستخدم Client لابد أن تعرف عنوان الجهاز الخادم Server ولكن الجهاز الخادم Server لا يهمه عنوان المستخدم Client فكل مستخدم Client يطلب منه خدمة فإنه يعطيه كالمدرس بالضبط.

ننشئ هدف object من نوع ServerSocket كالآتي:

```
ServerSocket server = new Socket Server (8000);
```

لاحظ استخدام نفس رقم الـ Port

الاستثناءات Exceptions الممكن حدوثها في الشبكات

من الممكن كتابة عنوان إنترنت IP address خاطئ للجهاز الخادم Server وهذا يسبب استثناء Exception.

الاستثناءات Exceptions لجهاز المستخدم Client هما UnknownHostException إذا كان عنوان الإنترنت IP address للجهاز الخادم Server غير صحيح والاستثناء IOException إذا حدث خطأ في القراءة أو الكتابة من أو إلى المقبس Socket.

الاستثناء Exception للجهاز الخادم Server هو IOException إذا حدث خطأ في القراءة أو الكتابة من أو إلى المقبس Socket.

الاستماع إلى أجهزة المستخدمين Clients

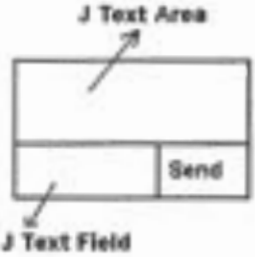
كما ذكرنا من قبل فإننا تستخدم الدالة المسماة ()accept الموجودة في الفصيلة class المسماة ServerSocket كالآتي.

```
Socket client = serverSocket.accept ();
Connection con = new Connection (client);
```

الآن نريد أن ننشئ تطبيقاً لبيان قوة لغة الجافا Java في الشبكات Networks. التطبيق هو برنامج دردشة Chatting بسيط فهيا إلى شرح المثال.

مثال 1: بناء برنامج دردشة Chatting

- نريد أن ننشئ تطبيقاً كما بالشكل.



- نفس هذا التطبيق سيتكرر مرة أخرى بنفس الشكل في الجزء القادم.
- التطبيق الأول هو الخاص بالجهاز الخادم (Server) هذا التطبيق لابد أن يكون قادراً على استقبال رسائل من أي شخص في الشبكة وكتابتها في أداة النص متعددة الأسطر JTextArea وفي نفس الوقت يستطيع إرسال الرسالة المكتوبة في أداة النص JTextField إلى الشخص الذي يتحاور معه.

إذن ما نريد ان ننشئه هو برنامج دردشة بسيط بحيث أنك تكتب الرسالة التي تريد إرسالها في أداة النص JTextField وعند الضغط على زر Send فإنك ترسل الرسالة. أي رسالة تستقبلها أو ترسلها يتم طباعتها في أداة النص متعددة الأسطر JTextArea.

نفس هذا الكلام يتم تطبيقه على الشخص الذي تتحاور معه ولكننا سننشيء برنامجاً على طريقة Client/Server بمعنى أن الجهاز الخادم سيكون قادراً على استقبال الرسالة من أكثر من شخص وجهاز المستخدم سيرسل فقط الرسالة ويستقبلها من الجهاز الخادم.

الفصل التاسع عشر

تنفيذ أكثر من عملية فى وقت واحد
Threading

Programming Concepts 19

فى هذا الفصل

فى هذا الفصل نتناول موضوع اعداد برنامج ينفذ أكثر من عملية فى نفس الوقت وهو ما يسمى Threading وذلك من خلال النقاط التالية :-

- ما هو الـ Thread
- كيفية إنشاء Thread
- ما هى الفصيلة Thread
- أعضاء الفصيلة Thread
- تغيير درجة أولوية التنفيذ Priority
- التحكم فى تنفيذ الـ Thread

تخيل أنك كنت تجلس على مكتبك تقرأ كتاباً. بإمكانك شرب كوب من الشاى وأنت تقرأ كما يمكنك أيضاً الاستماع إلى موسيقى.

- إذن فى نفس الوقت أمكنك القيام بعدة أعمال وإلا كان لابد أن تكمل قراءة الكتاب ثم تشرب كوب الشاى ثم تستمتع إلى الموسيقى.
- طبعاً كونك أديت جميع هذه الأعمال فى نفس الوقت وفرت عليك وقتاً لتنفيذ جميع هذه العمليات.
- من هنا نستطيع أن نعرف اكثر من مصطلح.

العمليات المتعددة Multiprocessing

- هو القدرة على القيام بأكثر من عملية process فى نفس الوقت.
- فمثلاً يمكن فتح برنامج الكتابة Word وبرنامج الجداول Excel وبرنامج العروض PowerPoint فى نفس الوقت ولكن لاحظ أنه لا توجد علاقة بين برنامج الكتابة Word وبرنامج الجداول Excel وبرنامج العروض PowerPoint بمعنى أن اى برنامج منهم لا يعتمد على الآخر أو ينتظر إتمام عملية من الآخر.

المهام المتعددة Multithreading

- هو القدرة على أداء اكثر من وظيفة داخل نفس العملية process.
- فمثلاً إذا كنت تلعب لعبة تتعلق بالفضاء مثلاً فإنك تحرك طائرتك وتظهر لك أهدافاً لتضربها كما يوجد أيضاً صوت فى اللعبة.
- إذن داخل العملية الواحدة (والتى هي اللعبة) فيمكنك القيام بأكثر من وظيفة فى نفس الوقت وإلا تخيل أنك تضرب ثم تظهر الأهداف ثم تتحرك ثم يظهر الصوت وهي طبعاً عملية غير منطقية لأنك تريد أن تنفذ جميع هذه العمليات فى نفس الوقت.

إذن نستطيع القول أنه عندنا نوعان من التطبيقات:

1. التطبيقات ذات المهمة الواحدة Single-Threaded Applications
2. التطبيقات ذات المهام المتعددة Multi-Threaded Applications

التطبيقات ذات المهمة الواحدة Single-Threaded Applications

هو التطبيق الذى يستطيع أداء وظيفة واحدة فى وقت ما. فإذا كان عندك اكثر من وظيفة تريد تنفيذها فلابد أن ينتهى تنفيذ الوظيفة الأولى حتى يمكنك تنفيذ الثانية ولابد أن ينتهى تنفيذ الوظيفة الثانية حتى تبدأ الثالثة وهكذا حتى ينتهى تنفيذ جميع الوظائف الموجودة بالبرنامج.

التطبيقات ذات المهام المتعددة Multi-Threaded Applications

- هو التطبيق الذى يستطيع أداء أكثر من وظيفة فى نفس الوقت. فإذا كان عندك أكثر من وظيفة فيمكنك تنفيذ هذه الوظيفة مع وظيفة أخرى فى نفس الوقت بدون أن تنتظر انتهاء الوظيفة الأولى.
- فمثلاً يمكنك باستخدام برنامج الكتابة Word كتابة التقارير وطباعتها.
- أيضاً يمكنك باستخدام برنامج مستكشف الإنترنت Internet Explorer أن تنزل برنامج (Download) وأن تستمع إلى موسيقى وأن تتصفح صفحات الوب.
- إمكانية أداء أكثر من وظيفة فى نفس الوقت عملية هامة جداً لأنها أقرب إلى الحياة الواقعية.
- فى هذا الفصل سوف نتحدث عن كيفية أداء اكثر من وظيفة فى نفس الوقت وكيفية الاستفادة من ذلك.

فى لغة VB.NET

ما هو Threading؟

هى طريقة تسمح بتنفيذ اكثر من عملية Task فى وقت واحد كل عملية تنفذ فى Thread مع امكانية التحكم فى هذه العمليات مثل ايقاف الـ Thread مؤقتا أو نهائيا.

ومثال لذلك تطبيق الانترنت الذى يأتيه أكثر من طلب إذا أمكن التعامل مع أكثر من طلب فى نفس الوقت زادت كفاءة التطبيق ويتم ذلك بإنشاء أكثر من Thread باستعمال الفصيلة thread. ولتوضيح كيفية استعمال فكرة إنشاء Thread وتنفيذ عملية داخله تابع المثال التالى

مثال

1- قم باعداد تطبيق جديد من النوع Console Application

2- اكتب به السطور التالية.

```
Imports System
Imports System.Threading
Module Module1
    Sub Main()
        Dim t1 As System.Threading.Thread
        t1 = New System.Threading.Thread(AddressOf ThreadEntryMethod)
        t1.Name = "Ready"
        t1.Start()

        Dim t2 As System.Threading.Thread
        t2 = New System.Threading.Thread(AddressOf ThreadEntryMethod)
        t2.Name = "Tells"
        t2.Start()
    End Sub

    Public Sub ThreadEntryMethod()
        Dim r As Random = New Random(Thread.CurrentThread.GetHashCode())
        Do While (True)

            Console.WriteLine(Thread.CurrentThread.Name)
            Dim rount As Integer = r.Next(1000)
            Thread.Sleep(rount)
        Loop
    End Sub
End Module
```

في C#

```
class MyThreadClass
{
    //ThreadEntryMethod method
    public void ThreadEntryMethod()
    {
        Random r = new Random(
            Thread.CurrentThread.GetHashCode());
        while (true)
        {
            Console.WriteLine(Thread.CurrentThread.Name);
            int rount = r.Next(1000);
            Thread.Sleep(rount);
        }
    }
}
class CreateThread
{
    public static void Main()
    {
        MyThreadClass mtc = new MyThreadClass();

        //establish thread entry method
        ThreadStart ts = new ThreadStart(mtc.ThreadEntryMethod);

        //create first thread, set name to "Ready", start
        Thread t1 = new Thread(ts);
        t1.Name = "First";
        t1.Start();

        //create second thread, set name to "Tells", start
        Thread t2 = new Thread(ts);
        t2.Name = "Second";
        t2.Start();
    }
}
```

فى هذه السطور

- فى السطر رقم 2 يتم عمل Imports للمكتبة (NameSpace) الموجودة بالاسم System.Threading وذلك لتعريف الفصيلة Thread بها.
- فى السطر رقم 5 تم الاعلان المعرفة فى Threading المعرفة فى System.
- فى السطر رقم 6 تم إنشاء المتغير (الهدف) Object المسمى t1 باستخدام الكلمة New مع ارسال معامل الدالة البناء هو عنوان الدالة ThreadEntryMethod وهى الدالة التى تنفذ فى هذا الـ Thread.
- فى السطر رقم 7 تم تسجيل اسم الـ Thread عن طريق تسجيل قيمة الخاصية Name للهدف t1.
- فى السطر رقم 8 تم استدعاء الدالة ()Start العضوة فى الفصيلة Thread والتى تؤدى إلى بدء عمل الـ Thead وبالتالى تنفيذ الدالة المعرفة فيه ولمشار اليها عند إنشاء هدف من الـ Thead.
- والدالة المعرفة مع الهدف t1 هى الدالة ()ThreadEntryMethod و بالتالى يبدأ تنفيذ هذه الدالة وهذة الدالة معرف فى السطور من 16 إلى 24.
- فى السطر رقم 10 تم إنشاء المتغير (الهدف) Object المسمى t2 باستخدام الكلمة New مع ارسال معامل الدالة البناء هو عنوان الدالة ThreadEntryMethod وهى الدالة التى تنفذ فى هذا الـ Thread.
- فى السطر رقم 11 تم تسجيل اسم الـ Thread عن طريق تسجيل قيمة الخاصية Name للهدف t1.
- فى السطر رقم 12 تم استدعاء الدالة ()Start العضوة فى الفصيلة Thread والتى تؤدى إلى بدء عمل الـ Thead وبالتالى تنفيذ الدالة المعرفة فيه ولمشار إليها عند إنشاء هدف من الـ Thread.
- فى السطور من 16 إلى 24 تم تعريف الدالة ()ThreadEntryMethod و التى تنفذ مع كلا من t1,t2 المعرفين من الفصيلة Thread.

- فى السطر رقم 17 يتم الاعلان عن متغير r من النوع Random وباستعمال الدالة Random لتوليد أرقام عشوائية وكل مرة يسجل رقم فى المتغير r.
- فى السطر رقم 18 يبدأ التكرار Do While بالشرط True الذى يؤدى إلى استمرار التكرار.
- فى السطر رقم 20 يتم استعمال ()Writeline مع الفصيلة Console لطباعة اسم الـ Thread.
- فى السطر رقم 21 يتم أخذ رقم تالى من الأرقام العشوائية ثم ارسال هذا الرقم للدالة ()Sleep فيؤدى إلى توقف الـ Thread لمدة المتغير Count الذى يأخذ قيمته من المتغير r.

نفذ البرنامج F5 تحصل على نتيجة التنفيذ كما فى الشكل (1-19).

الشكل (1-19)

فى هذا الشكل نلاحظ طباعة الكلمة Heads الخاص بالـ Thread الأول t1 ثم الكلمة Tails الخاص بالـ Thread الثانى t2 بشكل غير منتظم ويرجع ذلك إلى قيام الكمبيوتر بتنفيذ ها بشكل غير منظم حسب تفرغ المشغل processor مما أدى إلى ظهور العبارات بشكل غير منظم.

ما هى الفصيلة Thread!

هى الفصيلة المسؤلة عن تعريف Thread جديد مع توفير الأعضاء (الدوال Methods والخصائص Properties) التى تتحكم فى جميع عمليات الأت Thread مثل بدء تشغيل الـ Thread (الدالة Start()) وايقاف التشغيل Stop() والكثير من العمليات.

تغيير درجة أولوية التنفيذ Priority

عند تشغيل أكثر من Thread يقوم الكمبيوتر بتشغيلها حسب الوقت المالح لمشغل Processor ولكن قد تحتاج لتشغيل Thread قبل أخر وذلك لاهميته أو قد تحتاج لإعادة ترتيب الأولويات وهذا ما يسمى Priority وهى عبارة عن خاصية يتم اعطائها قيمة حسب الأولوية التى تريدها ، وتأخذ هذه الخاصية أحد مجموعة قيم.

ولتوضيح ذلك تابع المثال التالى:

1- قم بإعداد تطبيق جديد من النوع Console Application

2- اكتب به السطور التالية.

```
Imports System
Imports System.Threading
Module Module1
   Sub Main()
      Dim t1 As System.Threading.Thread
      Dim t2 As System.Threading.Thread
      Dim t3 As System.Threading.Thread
      Dim t4 As System.Threading.Thread

t1 = New System.Threading.Thread(AddressOf threadEntryMethod)
t2 = New System.Threading.Thread(AddressOf ThreadEntryMethod)
t3 = New System.Threading.Thread(AddressOf ThreadEntryMethod)
t4 = New System.Threading.Thread(AddressOf ThreadEntryMethod)

1.  t1.Priority = ThreadPriority.Normal
2.  t2.Priority = ThreadPriority.Lowest
3.  t3.Priority = ThreadPriority.Highest
4.  t4.Priority = ThreadPriority.AboveNormal
      t1.Name = "Normal"
      t2.Name = "Lowest"
      t3.Name = "Highest"
      t4.Name = "AboveNormal"
```

```
        t1.Start()
        t2.Start()
        t3.Start()
        t4.Start()
    End Sub

    Public Sub ThreadEntryMethod()
        Dim r As Random = New
Random(Thread.CurrentThread.GetHashCode())
        Do While (True)

            Console.WriteLine(Thread.CurrentThread.Name)
            Dim count As Integer = r.Next(1000)
            Thread.Sleep(count)
        Loop
    End Sub
End Module
```

فى لغة C#

```
using System;
using System.Threading;

class ThreadPriority
{
public static void Main()
{
MyThreadClass mtc = new MyThreadClass();
ThreadStart ts = new ThreadStart(mtc.ThreadStart);

//create the threads and set the priorities
Thread threadLowest = new Thread(ts);
threadLowest.Priority =
System.Threading.ThreadPriority.Lowest;
Thread threadBelowNormal = new Thread(ts);
threadBelowNormal.Priority =
System.Threading.ThreadPriority.BelowNormal;
Thread threadNormal = new Thread(ts);
threadNormal.Priority =
System.Threading.ThreadPriority.Normal;
Thread threadAboveNormal = new Thread(ts);
threadAboveNormal.Priority =
System.Threading.ThreadPriority.AboveNormal;
Thread threadHighest = new Thread(ts);
threadHighest.Priority =
System.Threading.ThreadPriority.Highest;
```

```
//start thread race
threadLowest.Start();
threadBelowNormal.Start();
threadNormal.Start();
threadAboveNormal.Start();
threadHighest.Start();
}
}

class MyThreadClass
{
public void ThreadStart()
{
for (long count = 0; count <10000000; count++);
Console.WriteLine(
"Priority " + Thread.CurrentThread.Priority);
}
}
```

الجديد فى هذه السطور:

- فى هذه السطور تم تغيير درجة أولوية تنفيذ الـ Theads وذلك بتغيير قيمة الخاصية Priority كما فى السطور 1، 2، 3، 4.

حيث يتم إعطاء كل Thread درة أولوية مختلفة.

3- نفذ البرنامج F5 لحصل على نتجة التنفيذ كما فى الشكل (2-19).

الشكل (2-19)

التحكم فى تنفيذ الـ Thread

يمكن التحكم فى تنفيذ الـ Thread بحيث يمكن تغيير حالته من Start اى ابتداء تشغيله إلى إيقافه مؤقت إلى استعادة تشغيله إلى إيقافه نهائيا وهكذا ولتوضيح ذلك تابع المثال التالى

1- قم بإعداد تطبيق جديد من النوع Console Application.

2- اكتب به السطور التالية.

```
Imports System
Imports System.Threading
Module Module1
Sub Main()
Dim t1 As System.Threading.Thread
t1 = New System.Threading.Thread(AddressOf ThreadEntryMethod)
t1.Name = "Normal"
1.  t1.Start()
2.  t1.Suspend()
3.  Thread.Sleep(1000)
4.  t1.Resume()
5.  t1.Suspend()
6.  Thread.Sleep(1000)
7.  t1.Resume()
8.  t1.Suspend()
9.  Thread.Sleep(1000)
10. t1.Resume()
11. t1.Abort()
End Sub
Public Sub ThreadEntryMethod()
Dim r As Random = New Random(Thread.CurrentThread.GetHashCode())
Do While (True)
Dim i
Console.WriteLine("**********")
Dim count As Integer = r.Next(1000)
Thread.Sleep(count)
Loop
End Sub
End Module
```

فى لغة C#

```
using System;
using System.Threading;
class ControlThread
{
    public static void Main()
    {
        MyThreadClass mtc = new MyThreadClass();
        ThreadStart ts = new ThreadStart(mtc.ThreadStart);
        Thread t = new Thread(ts);
        t.Start();

        for (int i=0; i<3; i++)
        {
        t.Suspend();
            Thread.Sleep(1000);
            t.Resume();
            Thread.Sleep(1000);
            Console.WriteLine();
        }
        t.Abort();
    }
}

class MyThreadClass
{
    public void ThreadStart()
    {
        while (true)
        {
            Console.Write("*");

            //sleep for 0.1 second
            Thread.Sleep(100);
        }
    }
}
```

فى هذه السطور:

- تم تغيير حالة الـ Thread من حالة إلى حلة وذلك من السطر رقم 1 إلى السطر رقم 11 كما يلى.
- فى السطر رقم 1 تم بدء تشغيل الـ Thread باستدعاء الدالة start().
- فى السطر رقم 2 تم إيقاف تشغيل الـ Thread ايقاف مؤقتا فى وضع الاستعداد وذلك باستدعاء الدالة Suspend().

- فى السطر رقم 3 يتم تعطيل العمليات لمدة 1000 ملى ثانية وذلك باستدعاء الدالة Sleep().
- فى السطر رقم 4 يتم استئناف تشغيل الـ Thread وذلك باستدعاء الدالة Resume().
- وهكذا باقى السطور حتى يتم انهاء عمل الـ Thead ايقاف نهائى فى السطر رقم 11.

فى لغة Java

أنواع المهام Threads فى لغة الجافا Java

يوجد نوعان من المهام Threads فى لغة الجافا Java وهما:

1. مهام المستخدم user Threads.
2. المهام الحارسة daemon threads.

إنشاء المهام Threads

توجد طريقتان لإنشاء المهام threads فى لغة الجافا Java:

1. أن ترث من الفصيلة class المسماة Thread.
2. أن تنفذ الـ interface المسمى Runnable.

كل طريقة لها مميزاتها وعيوبها واستخدام اى طريقة منهما تعتمد على احتياجك. فى الصفحات التالية سنعطى أمثلة على استخدام كلاً منهما ثم نبين مميزات وعيوب كلا الطريقتين.

أولاً: الوارثة من الفصيلة class المسماة Thread

يمكنك الوارثة من الفصيلة Thread مباشرة كما تعلمنا من قبل. فإذا أردت إنشاء فصيلة class اسمها MyThread وتريدها أن تنفذ مهمة Thread معينة فإنك ترث من الفصيلة Thread كالآتي

```
class  MyThread extends Thread
{
}
```

كما ذكرنا فإن المهمة thread هى وظيفة يمكنها أن تعمل فى نفس الوقت مع عدة وظائف أخرى فإذا أردت أن تكتب كود البرمجة الخاص بالوظيفة التى تريدها، فلابد من إضافة دالة تسمى () run وبداخلها تكتب الكود الذى ينفذ الوظيفة التى تريدها أى كالآتى:

```
class  MyThread extends Thread
{
public void run ()
        {
            \\ put your code here
         }
}
```

- الوظيفة التى تريد تنفيذها سوف تضيف الكود الخاص بها فى الدالة ()run.
- اتفقنا ان دالة الـ ()main تعتبر مهمة thread وهى الوحيدة التى تبدأ أوتوماتيكياً بدون تدخل المبرمج. إذن من داخل دالة الـ ()main فإنك تريد ان تبدأ المهمة thread التى عرفتها فى الكود الخاص بالدالة () run.
- أولاً لابد أن تنشئ هدف object من الفصيلة class المسماة MyThread كالآتى

```
MyThread firstThread = new Mythread ();
```

- الآن إذا أردت بدأ المهمة thread المعرفة فى الدالة () run فإنك تستخدم الدالة () start المعرفة فى الفصيلة class المسماة Thread. عندما تستدعى الدالة ()start فإن لغة الجافا Java سوف تستدعى الدالة ()run اى انك تبدأ المهمة thread كالآتى:

```
firstThread.start ();
```

- قد تسأل سؤالاً: لماذا لا نستدعى الدالة ()run مباشرة بدلاً من استدعاء الدالة ()start؟
- الإجابة هى أن الدالة () start هى التى تعنى بدء المهمة thread بالنسبة للغة الجافا Java فإذا استدعيت الدالة () run فستظن لغة الجافا Java أنك تستدعى دالة عادية عرفتها أنت كمبرمج. لاحظ أن الفصيلة MyThread هى فصيلة class عادية ويمكنك وضع أى دالة فيها بجانب الدالة ()run ولذلك تم وضع الدالة ()start ليبان ان الكود الذى سينفذ عند استدعائها هو مهمة thread بمعنى انه يمكن تنفيذ وظيفة أخرى معها. فإذا استدعيت الدالة ()run مباشرة فلن تكون مهمة thread فى هذه الحالة ولابد أن ينتهى الكود الخاص بها قبل استدعاء وتنفيذ اى دالة أخرى.
- لذلك تعود انك تستدعى الدالة ()start وهذه الدالة تنفذ الكود الموجود فى الدالة run () كمهمة thread أى انه سيتم تنفيذ هذه الدالة مع إمكانية تنفيذ وظيفة أخرى معها.
- الآن نريد مثالاً يوضح لنا كيف يمكننا تنفيذ أكثر من وظيفة فى نفس الوقت.

مثال 1 بدون المهام thread:

هذا المثال لا يمثل مهمة thread ولكننا وضعناه لبيان مشكلة ما وسوف نرى كيف يتم حلها باستخدام المهمة thread.

واليك كود البرمجة:

```
1:  class MyThread
2:  {
3:   public void start()
4:   {
5:        run();
6:   }
7:
8:   public void run()
9:   {
10:       while(true)
11:       {
12:            System.out.println("This is my thread");
13:       }
14: }
15:}
16:
17:class Main
18:{
19: public static void main (String[]args)
20: {
21:       MyThread firstThread = new MyThread ();
22:       firstThread.start();
23:
24:       while(true)
25:       {
26:            System.out.println("This is main thread");
27:       }
28: }
29:}
```

شرح الكود:

- في السطر 1 يتم إنشاء فصيلة class تسمى MyThread. هذه الفصيلة class تحتوى على دالة تسمى () start كما فى سطر 3.
- هذه الدالة () start تستدعى الدالة الأخرى المسماة () run كما فى سطر 5.

- في السطر 8 يتم تعريف الدالة () run.
- في السطر 10 يتم إنشاء دوارة لا نهائية infinite loop باستخدام while.
- في السطر 12 يتم طباعه رسالة على الشاشة.
- في السطر 17 يتم تعريف فصيلة class اسمها Main وهي تحتوى على الدالة () main.
- في السطر 21 يتم إنشاء هدف object من نوع الفصيلة class المسماة MyThread.
- في السطر 22 يتم استدعاء الدالة () start المعرفة فى الفصيلة class المسماة MyThread.
- في السطر 24 نبدأ دوارة لا نهائية infinite loop باستخدام while.
- في السطر 26 يتم طباعه رسالة على الشاشة.
- الآن يبدأ تنفيذ البرنامج باستدعاءك الدالة ()start كما فى سطر 22 وكما ذكرنا من قبل فإن هذه الدالة تستدعى الدالة ()run التى تطبع رسالة على الشاشة بصورة لا نهائية.
- إذا نفذت البرنامج فستجد طباعة الرسالة "This is My Thread" بصورة لا نهائية. انظر شكل (3-19).

شكل (3-19) المثال الأول

- إذن ما العمل إذا أردت طباعة الرسالة "This is My Thread" ثم فى نفس الوقت تنفذ باقى الكود الموجود فى دالة الـ ()main الذى يطبع الرسالة "This is Main Thread"؟
- الحل هو استخدام الوارثة من الفصيلة class المسماة Thread كما فى المثال التالى:

مثال 2: استخدام الفصيلة Thread

كود البرمجة:

```
1:  class MyThread extends Thread
2:  {
3:   public void run()
4:   {
5:        while(true)
6:        {
7:        System.out.println("This is my thread");
8:        }
9:   }
10:)
11:
12:class Main
13:{
14: public static void main (String[]args)
15: {
16:       MyThread firstThread = new MyThread ();
17:       firstThread.start();
18:
19:       while(true)
20:       {
21:            System.out.println("This is main thread");
22:       }
23: }
24:}
```

شرح الكود:

- لاحظ عدم وجود تعديل في الفصيلة class المسماة Main.
- التعديل حدث في الفصيلة class المسماة MyThread فأولاً جعلناها ترث من الفصيلة Thread ثم حذفنا الدالة ()start تماماً.
- لاحظ أن سطر 17 يستدعي الدالة ()start وقد تظن أنها غير موجودة بعد أن حذفناها ولكن تذكر حيث أننا ورثنا من الفصيلة Thread والتي يحتوي على الدالة ()start وهذه الدالة ()start هي التي تستدعي الدالة () run لتنفيذها كمهمة thread.
- الآن نفذ البرنامج ولاحظ طباعة الرسالتين "This is My Thread" و "This is main Thread" بطريقة لا نهائية.

النتيجة:

- أنه بالرغم من أن الدالة ()run تمثل دوارة لا نهائية ولكن أمكن تنفيذ دالة أخرى معها وهو باقى الكود فى الدالة ()main. اى أننا لم ننتظر انتهاء الدالة ()run لتنفيذ باقى الكود ولكن تم تنفيذ الاثنين فى نفس الوقت وهذه هي المهمة thread. انظر شكل (4-19).
- طبعاً ستقول أن هذا البرنامج السابق بلا فائدة؟ معك حق ولكننا وضعناه فقط للتوضيح وسترى فى أخر الفصل تطبيق عملى للمهام threads .

شكل (4-19) المثال الثانى

تنفيذ الـ interface المسمى Runnable

ستجد اختلافاً بسيطاً بين الوراثة من الفصيلة Thread وتنفيذ الـ interface المسمي Runnable وهذا سيتم توضيحه فى المثال التالى.

مثال 3: استخدام الـ interface المسمي Runnable

كود البرمجة:

```
1:  class MyThread implements Runnable
2:  {
3:   public void run()
4:   {
5:        while(true)
6:        {
```

```
7:              System.out.println("This is my thread");
8:      }
9:  }
10:}
11:
12:class Main
13:{
14: public static void main (String[]args)
15: {
16:      MyThread firstThread = new MyThread ();
17:      Thread t = new Thread(firstThread);
18:      t.start();
19:
20:      while(true)
21:      {
22:          System.out.println("This is main thread");
23:      }
24: }
25:          }
```

شرح الكود:

- اختلاف بسيط حدث بالنسبة للفصيلة class المسماة MyThread فبدلاً من الوراثة من الفصيلة Thread فهي تنفذ الـ interface المسمي Runnable كما في سطر 1 ولا يوجد تغيير آخر.
- في السطر 16 يتم تعريف هدف object من نوع الفصيلة MyThread ولا جديد هنا.
- الجديد هو سطر 17 وله أهمية قصوى وذلك لأن الـ interface المسمى Runnable لا يحتوى على الدالة start() ولذلك لا يمكن أن نكتب.

```
firstThread.start();
```

- وإلا سيعطى رسالة خطأ. إذن ما العمل؟
- لاحظ أن الدالة start() موجودة في الفصيلة class المسماة Thread ولذلك لابد أن ننشئ هدف object من نوع الفصيلة Thread ونمرر لدالة البناء constructor الخاصه بالفصيلة Thread، هدف object من نوع الـ interface المسمي Runnable اى كالآتي:

```
Thread t = new Thread (firstThread);
```

- إذن هنا أنشأنا هدف object من نوع الفصيلة Thread ومررنا له الفصيلة class الفعليه التي تحتوى على الكود الخاص بالمهمة Thread التي تريد تنفيذها كما في سطر 17.

- حيث أن t هو هدف object من نوع الفصيلة Thread، إذن فهو يحتوى على الدالة start() ولذلك نستطيع استدعاء هذه الدالة كما فى سطر 18.
- نفذ البرنامج وستلاحظ الحصول على نفس نتيجة البرنامج السابق الذى تم تنفيذه باستخدام الوارثة من الفصيلة Thread.

الرسم التالى يوضح عملية بدء الـthread فى البرنامج السابق.

النتيجة:

- لاحظت ظهور نفس النتيجة باستخدام الوراثة من الفصيلة Thread وتنفيذ الـ interface المسمي Runnable إذن ما الفرق بينهما ومتى نستخدمها؟
- تذكر أن لغة الجافا Java لا تدعم التوارث المتعدد Multiple Inheritance ولذلك إذا ورثت من الفصيلة Thread، فلا يمكنك أن ترث من أى فصيلة class أخري وهذا يعتبر عائقاً لك في بعض الحالات.
- أما إذا كنت تريد أن تنفذ مهمة thread وفى نفس الوقت ترث من فصيلة class أخري، ففي هذه الحالة لا بد من استخدام الـ interface المسمي Runnable.
- ولكن الفصيلة class المسماة Thread تحتوى على الدالة start() التى تبدأ المهمة thread كما تحتوى الفصيلة Thread على عدة دوال أخرى مفيدة ستذكر لاحقاً كما أن استخدام الفصيلة Thread يعتبر أسهل من استخدام الـ interface المسمي Runnable.
- كملخص: إذا كنت ستنشئ الفصيلة class الخاصة بإنشاء المهمة thread ولن تحتاج أن

ترث من فصيلة class أخري فاستخدم الفصيلة Thread لأنها أسهل وتوفر لك الدوال التى تريدها مباشرة أما إذا كنت سترث من فصيلة class أخري فاستخدم الـ interface المسمي Runnable.

بعض الدوال المهمة للمهام Threads

1. الدالة currentThread()

عادة يكون عندك أكثر من مهمة thread تعمل فى نفس الوقت فإذا أردت الحصول على هذف object من المهمة thread الحالية فاستخدم الدالة currentThread() كما سنري في المثال القادم.

2. الدالة getName()

كل مهمة thread لها اسم فمثلاً مهمة دالة الـ main() اسمها ايضاً main أما المهمة thread التى تنشئها بنفسك فتأخذ اسم Thread-0,Thread-1 ...وهكذا.

3. الدالة setName()

إذا لم تكن تعجبك أسماء Thread-0,Thread-1 وهكذا، كأسماء للمهمة thread، فبإمكانك تغيير اسمها عن طريق الدالة setName().

4. الدالة sleep ()

وهى دالة لجعل المهمة thread تتوقف مؤقتاً عدداً من الثوانى يتم تحديده بمعرفة المبرمج. هذه الدالة تستقبل معامل من نوع long لتحدد كم ألف جزء من الثانية تريد أن تتوقف بمعنى أنك إذا أردت أن تجعل المهمة thread تتوقف لثانية واحدة فتمرر للدالة sleep() القيمة 1000.

أى أن وحدة القياس هنا هى المللى ثانية وليس ثانية. لاحظ أنك تمرر قيمة long أى لا يوجد بها كسور أو أرقام عشرية.

5. الدالة setPriority ()

حيث أنه عندك عدة مهام تنفذ جميعها فى نفس الوقت، فمن الطبيعى أن يكون لكل

مهمة أولوية معينة فبعض المهام تكون حرجة ولابد أن تتم في أسرع وقت أما البعض الأخر فلا تمثل السرعة فارقاً ولا يهمنا أن تتم سريعاً.

الدالة () setPriority تحدد أولوية تنفيذ المهمة thread بحيث أنك تمرر للدالة معامل من نوع int وهذا المعامل لا بد أن يكون رقماً من 1 إلى 10 بحيث أن القيمة 1 تمثل الأقل أهمية والقيمة 10 تمثل الأهمية القصوى والقيمة 5 تمثل الأهمية العادية.

- إذا لم تحدد أولوية المهمة فإن لغة الجافا Java تعتبر الأولوية تساوى 5 أى تعطى لكل مهمة أولوية عادية ولا تكمل مهمة معينة على حساب مهمة أخرى.

الأمثله التالية توضح استخدام هذه الدوال

مثال 4: توضيح استخدام بعض الدوال المهمة للمهام Threads

كود البرمجة:

```
1:  class MyThread implements Runnable
2:  {
3:   public void run()
4:   {
5:         Thread myThread = Thread.currentThread();
6:         myThread.setName("MyThread");
7:         System.out.println(myThread.getName());
8:   }
9:  }
10:
11: class Main
12: {
13: public static void main (String[]args)
14: {
15:       MyThread firstThread = new MyThread ();
16:       Thread t = new Thread(firstThread);
17:       t.start();
18:
19:       Thread myThread = Thread.currentThread();
20: myThread.setName("Main Thread");
21: System.out.println(myThread.getName());
22: }
23:          }
```

شرح الكود:

- في السطر 1 يتم تعريف الفصيلة class المسماة MyThread والتي سنستخدمها لإنشاء مهمة thread ولذلك فهي تنفذ الـ interface المسمي Runnable وتنشئ فيها الدالة run () كما في سطر 3.
- في السطر 5 يتم إنشاء هدف object من نوع الفصيلة Thread اسمها myThread . ولجعل الهدف myThread يشير إلى المهمة thread الحالية، استخدمنا الدالة currentThread() . لاحظ ان الدالة currentThread() هي دالة static ولذلك استخدمنا اسم الفصيلة class المسماة Thread مباشرة دون الحاجه إلى إنشاء هدف object منها.
- في السطر 6 يتم إعطاء المهمة thread الحالية اسم "My Thread" وبالطبع يمكنك اختيار أي اسم .
- في السطر 7 يتم طباعه اسم المهمة thread الحالية. وقد حصلنا على اسم المهمة thread الحالية عن طريق الدالة getName().
- السطور 5،6،7 كررناها مرة أخرى في سطور 19،20،21 ولكن بالنسبة لدالة الـ main(). يمكنك الآن تنفيذ البرنامج ولاحظ طباعة اسم كل مهمة thread كما حددناه. انظر شكل (5-19).

شكل (5-19) المثال الرابع

المثال التالى يوضح الدالة sleep()

مثال 5: توضيح استخدام الدالة sleep() باستخدام الـ interface المسمى Runnable

نريد أن نطبع الأرقام من صفر إلى 10 مع وجود فارق زمنى يساوي ثانية واحدة بين طباعة كل رقم والذى يليه.

كود البرمجة:

```
1:  class MyThread implements Runnable
2:  {
3:   public void run()
4:   {
5:        Thread myThread = Thread.currentThread();
6:
7:        for(int i = 0 ; i <= 10 ; i++)
8:        {
9:             System.out.println(i);
10:
11:            try
12:            {
13:            myThread.sleep(1000);
14:            }
15:
16:            catch (InterruptedException e)
17:            {
18:            System.out.println(e);
19:            }
20:       }
21: }
22:}
23:
24:class Main
25:{
26: public static void main (String[]args)
27: {
28:       MyThread firstThread = new MyThread ();
29:       Thread t = new Thread(firstThread);
30:       t.start();
31: }
32:     }
```

شرح الكود:

- المشكلة التي تحدث عند تنفيذ الـ interface المسمى Runnable أنه لا يحتوى على الدالة ()sleep ولكنها موجودة في الفصيلة Thread، إذن نريد إنشاء هدف object من نوع الفصيلة Thread ليشير إلى المهمة thread الحالية ولذلك استخدمنا الدالة ()currentThread كما في سطر 5.
- في السطر 7 يتم إنشاء دوارة من صفر إلى 10.
- في السطر 9 يتم طباعه قيمة المتغير i والذي يمثل قيمة العداد.
- في السطر 13 يتم جعل المهمة thread تتوقف مؤقتاً لمدة ثانية واحدة.
- لاحظ أن الهدف object المسمي myThread (سطر 5) يشير إلى المهمة thread الحالية وأنه من نوع الفصيلة Thread إذن فهو يحتوى على الدالة ()sleep وبذلك نستطيع ان نوقف المهمة thread مؤقتاً عن طريق استدعاء الدالة ()sleep.
- محاولة استخدام الدالة ()sleep قد ينتج عنها حدوث استثناء Exception واسمه InterruptedException ولذلك استخدمنا try و catch لمعالجة هذا الاستثناء Exception
- أخيراً بعد تنفيذ البرنامج لاحظ طباعة الرقم صفر ثم التوقف لمدة ثانية ثم طباعة الرقم 1 ثم التوقف لمدة ثانية وهكذا حتى رقم 10. انظر شكل (6-19).

شكل (6-19) المثال الخامس

الفصل العشرون

برمجة الإنترنت
Web Programming

Programming Concepts في هذا الفصل

20

في هذا الفصل نشرح كيفية استعمال VB.NET & ASP.NET وهو الجديد والذي يوفر أدوات حديثة ونشرح مقدمة عن إعداد صفحات الإنترنت (Internet Developing) و الطرق المختلفة لها ونقدم بعض هذه الطرق وذلك من خلال النقاط التالية:

- ما هي برمجة وتطوير الإنترنت Internet Developing &programming
- طرق إعداد مواقع للإنترنت web sites
- طرق كتابة برامج للإنترنت Internet Programming
- مزايا استعمال ASP مع HTML
- مزايا استعمال ASP.NET
- أول تطبيق ASP.NET
- كيف يعمل برنامج ASP.NET
- ما هي أدوات SERVER CONTROLS
- أدوات VALIDATION CONTROL
- أنواع التحقق من صحة البيانات VALIDATORS
- استعمال ملفات CSS

ما هي برمجة وتطوير الإنترنت WEB Developing & programming

المقصود بتطوير وبرمجة الإنترنت هو إعداد المواقع و التطبيقات التي تعمل علي الإنترنت .

المواقع Web Sites

الموقع قد يكون بسيط عبارة عن صفحة رئيسية ومجموعة من الصفحات المرتبطة بها وذلك مثل كثير من المواقع ، والغرض منها هو التعرف بالشركة أو الجهة أو الهيئة وعرض أنشطتها ، وبياناتها و كيفية الاتصال بها .

وهناك الكثير من الأدوات التي تستعمل لذلك سوف نعرض لك بعضها .

تطبيق الإنترنت Web Application

تطبيق الإنترنت (web Application) هو تطبيق يعمل من خلال شبكة الإنترنت وذلك لتحقيق وظيفة معينة مثل التطبيق الموجود بموقع شركة التوظيف ، وكذلك تسجيل بيانات الزائرين الذين يريدون وظائف ،و يوفر كذلك إمكانية البحث يستطيع صاحب عمل البحث عن موظفين ،وكذلك يستطيع الزائر البحث عن وظيفة ،وكذلك توجد تطبيق آخرى للإنترنت مثل تطبيقات التجارة الإلكترونية التى تقوم بتوفير قائمة بالشركات المنتجات وتوفير إمكانية اختيار المنتجات وإعداد فاتورة شراء وكثير من التطبيقات الأخرى.

وكذلك يمكن للموقع أن يحتوي علي نموذج أو أكثر من التطبيقات وليس تطبيق متكامل فمثلاً شاشة (نموذج) تسجيل البيانات Registration from في أي موقع ويتم بإعداد برامج الإنترنت وكذلك نموذج تطبيقات الزائر feed back وغيرها.

طرق وأدوات إعداد مواقع انترنت web site tools

كما أشرنا في الفقرة السابقة فإن متطلبات إعداد موقع web site تحتاج مجهود وخبرة أقل من متطلبات إعداد تطبيق إنترنت web application وهناك الكثير من الأدوات التي تساعد في إعداد موقع منها ما يلي:

1- استعمال أوامر وعلامات HTML.

2- استعمال برامج جاهزة مثل Front Page وبرنامج Dream weaver.

3- استعمال برامج حركة مثل Flash.

4- استعمال برامج مساعدة مثل Anfy, Cool 3D, XARA 3D, SWICH.

5- إضافة برامج JAVA والتي تسمي APPLETS.

6- تحميل المواقع علي الإنترنت UPLOAD.

هذه بعض أدوات ومتطلبات إعداد مواقع بعضها يجب أن تتعرف علي أساسياته والأخر اختياري يمكنك معرفته أو استبداله وفيما يلي شرح أساسيات بعض هذه الأدوات.

استعمال أوامر وعلامات HTML

ليس هناك في عالم تطوير الإنترنت من لا يعرف أو يتعرض لعلامات و أوامر HTML، مبدئياً الحروف HTML اختصار للعبارة Hyper Text Markupable Language ويطلق عيها لغة الترميز.

وهي الطريقة المستعملة لعرض البيانات والصور والصوت والحركة علي صفحات الإنترنت و ربما لا تستعمل أنت الأوامر والعلامات كثيراً ولكن لابد من التعرف عليها و علي أساسياتها وهي ليست لغة برمجة ولكن مجموعة من العلامات التي تتحكم في شكل البيانات علي صفحة الإنترنت مع توفير علامات للعمليات الأساسية مثل ربط الصفحات ببعضها والتعامل مع الصور وغيره.

طرق كتابة مستند HTML

هناك أكثر من طريقة لإعداد مستند HTML منها :

1- استعمال أي محرر سطور وليكن NotePade.

2- استعمال برامج جاهزة لكتابة ملفات HTML.

3- استعمال برامج التصميم مثل Front Page.

استعمال محرر السطور Notepad

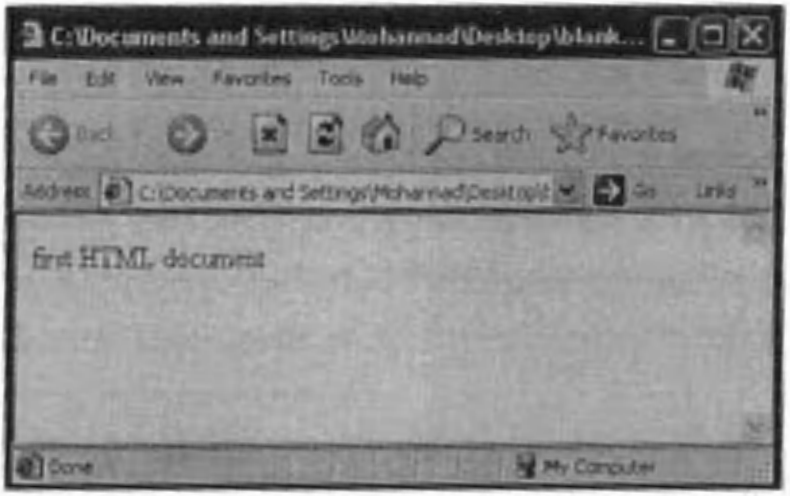
لتوضيح ذلك تابع الخطوات التالية:

1- من برنامج النوافذ من programs اختر Accessories ومنها اختر برنامج Notepad واكتب النص التالي:

```
1: < HTML >
2: <HEAD>
3:
4: </HEAD>
5: <BODY>
6: first HTML document
7: </body>
8: </HTML>
```

2- احفظ هذا الملف بالاسم HTEX1 بشرط أن يكون امتداده HTM أو HTML.

3- افتح هذا الملف من برنامج النوافذ (EXPLORER) يتم فتحة كصفحة إنترنت وعرضه كما بالشكل (1-20).



الشكل(1-20)

شرح السطور

- من النص السابق نلاحظ أنه عبارة عن مجموعة من علامات تفتح وتغلق للإشارة إلى شيء ما.
- في السطر رقم 1 يبدأ المستند بما يشير إلى أنه مستند HTML وذلك بالعلامة <HTML>.
- في السطر رقم 8 يتم إغلاق هذا المستند بنفس العلامة مع إضافة / وذلك بالشكل </HTML> وهكذا كل علامة تفتح لابد أن تغلق.

- في السطر رقم 2 قسم العنوان و المسمي HEAD ويبدأ بالعلامة <HEAD> ثم <TITLE> الذي يكتب به عنوان المستند والذي يظهر في شريط العنوان Title bar ، ويتم أنهاء هذا القسم في السطر 4 بالعلامة </Head>
- في السطر 5 يبدأ أشهر أجزاء المستند وهو جسم المستند والذي يبدأ بالعلامة <body> وبعد هذه العلامة وبداخله هذا القسم يتم كتابة كل المطلوب إظهاره بالمستند من عبارات وصور وحركة وصوت وجداول وغيره.
- وهذا ما تم حيث كتبت العبارة first HTML document في السطر السادس
- وتستمر السطور بعد العلامة <Body> وفي نهاية هذا القسم يتم أنهائه في السطر 7 بالعلامة </Body>
- وفي السطر 8 يتم أنهاء المستند كما أشرنا بالعلامة </HTML>

ملحوظة:

يتم فتح أي قسم بالعلامة وأنهائه بنفس العلامة مع إضافة / وذلك مثل <Body> وأنهائه بـ</Body>

وجميع أوامر HTML عبارة عن علامات بهذا الشكل <body> كل علامة لها معني وتؤدي إلي تأثير معين،وستكتفي بهذه المقدمة عن مستند HTML ولكن يمكنك تعلم جميع قواعدها بالرجوع إلي أحد كتب HTML أو الرجوع إلي كتابنا "تعلم HTML "

استعمال برنامج Front Page

يتميز برنامج front page بأن به صفحة تصميم تستعملها كما تستعمل برنامج الكتابة word وكذلك صفحة HTML تعرض علامات مستند HTML القابل لصفحة التصميم .

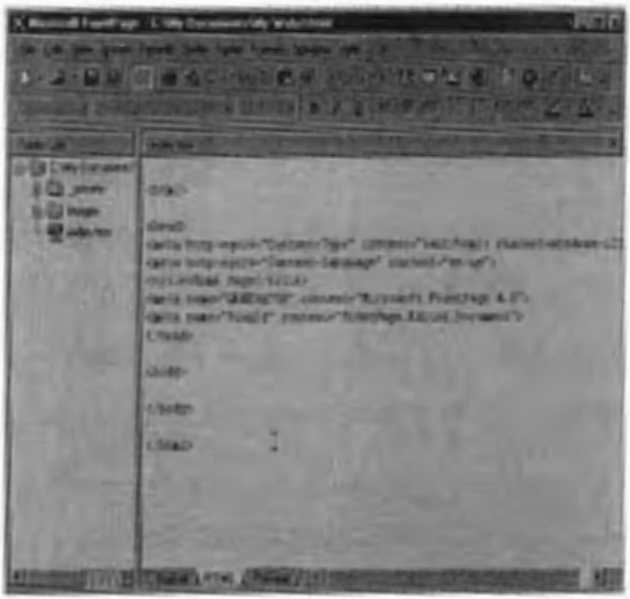
ويوجد من برنامج Front page نسخه مصغرة تأتي مع برنامج النوافذ win 98 تسمي front page express وتوجد نسخة مفضلة هي front page 2000 وسنري ما نشرحه هنا علي النسختين ولتوضيح ذلك تابع ما يلي:

1- قم بتشغيل برنامج front page (أي نسخة) لحصل علي الشاشة الرئيسية كما بالشكل (2-20).

الشكل(2-20)

في هذا الشكل نلاحظ وجود صفحة التصميم (Normal) صفحة HTML ، وصفحة العرض preview .

2- بدون إجراء أي تعديلات اضغط الصفحة HTML تحصل على الشكل الأساسي للمستند (HTML) كما بالشكل (3-20) .



الشكل (3-20)

3- اضغط الصفحة Normal تعود إلي صفحة التصميم مرة أخرى اكتب بها عبارة كما تكتب في برنامج الكتابة word ولتكن العبارة :

"HI from front page"

4- اضغط الصفحة HTML تحصل علي المقابل للمتغير، في هذا الشكل تلاحظ كتابة العبارة في قسم جسم المستند <BODY>

5- احفظ الصفحة ثم قم بفتح الملف كما سبق تلاحظ فتح ملف HTML كصفحة إنترنت.

ملحوظة

يمكنك تعلم قواعد وعلامات HTML عن طريق برامج FRONT PAGE وذلك بكتابة عبارات وتغيير الفونتات وغيره في صفحة التصميم ثم مشاهدة المقابل لها في صفحة HTML .

ويساعدك هذا البرنامج في إعداد مواقع متكاملة مع إمكانية التعاون مع البرامج الأخرى وفي هذه الفقرة تقدم لك شرحاً مختصر عن كيفية استعمال برامج front page في إعداد موقع.

1- قم بتشغيل برنامج front page (بعد التأكد من إعداده بجهازك).

2- تحصل علي صفحة بيضاء كصفحة الكتابة في برنامج word.

3- قم بكتابة وتنسيق وتغير نوع وحجم الخطوط لتعطي الشاشة الرئيسية لموقع باسم friends

4- احفظ هذه الصفحة تحت الاسم main page (أو اسم مؤقت).

5- ابدأ صفحة جديدة وذلك باختيار file ثم New ثم page ، تحصل علي صفحة بيضاء اكتب بها محتويات صفحة حول الموقع احفظ هذه الصفحات تحت اسم About.

6- كرر الخطوتين 6،7 مع باقي الاختيارات لتحصل علي الصفحات بباقي عناوين الصفحة الرئيسية.

7-افتح الصفحة الرئيسية .

8- قم بربط كل اختيار من الصفحة الرئيسية بالصفحة الخاصة به كما يلي:

- قم بتظليل الاختيار الأول "حول الموقع" .

- اضغط مفتاحي ctrl + K (و اضغط الطرف الأيمن للماوس على الاختيار واختر hyperlink) يظهر مربع حول تحديد ملف الصقر المطلوب ربطه اختر ملف الصفحة about.

9- كرر الخطوة السابقة مع باقي اختيارات الصفحة الرئيسية حتى يتم ربط جميع الاختيارات بالصفحات المقابلة لها .

10- احفظ الصفحة الرئيسية بعد التعديلات.

11- اضغط تبويب (صفحة) preview الموجودة أسفل الشاشة تلاحظ عرض الصفحة الرئيسية جرب اضغط الاختيار الأول "حول الموقع " تلاحظ ظهور الصفحة المرتبطة.

12- جرب الاختيارات الأخرى ولاحظ النتيجة .

بهذا تم إعداد الشكل الرئيسي لموقع بصفحاته ويمكنك تكرار نفس الفكرة للإعداد موقع مشابه مثل :

1- موقع شركة أو مؤسسة.

2- موقع بنك.

من فضلك حاول استعمال الخطوات السابقة في إعداد موقع آخر .

ويوفر برنامج front page الكثير من المهارات و الإمكانيات الأخرى مثل :

1- إضافة صور. 2- إضافة ملفات فيديو.

3- إضافة أوامر JAVA script . 4- إضافة برمجيات JAVA applet.

5- إضافة أوامر ASP.

6- التعامل مع قواعد البيانات والكثير من الإمكانيات ، ولكننا حاولنا فقط تقديم فكرة عن :

إعداد الموقع باستعمال برنامج FRONT PAGE .

و إذا أردت التفصيل ارجع إلى كتب تعليم FRONT PAGE أو كتابنا تعلم "إعداد مواقع الإنترنت".

استعمال برامج تحرير مستندات HTML

يوجد كثير من البرامج المعدة لغرض كتابة مستندات HTML وتوجد نسخ تجريبية منه على download.com قم بالبحث عن برامج وجرب استعمالها.

استعمال برنامج Flash

برنامج flash في البرامج المشهورة في مجال إعداد مواقع الإنترنت وهو يختلف عن برنامج front page وكذلك برنامج dream weaver فهو يوفر خدمات مختلفة شيئاً ما منها:

1- إعداد قطع حركة على الصور Move .

2- تكوين مناظر .

والشكل التالي شكل (4-20) يعرض أحد الأمثلة التي تم إعدادها باستعمال Flash وكذلك الشاشة الرئيسية للبرنامج

الشكل(4-20)

استعمال برامج مساعدة

هنا الكثير من برامج المساعدة وقد شرحنا بعض هذه الأدوات في فصل أدوات تطوير صفحات الإنترنت ومنها.

برنامج swish

الذي يساعد في إعداد النصوص text وحركتها بشكل جيد والشكل (5-20) يعرض

نص تم إعداده باستعمال switch كذلك الشاشة الرئيسية للبرنامج.

الشكل(5-20)

برنامج Xara3D

ويساعد في إعداد الأزرار (Buttons) والنصوص وغيره ، و الشكل (6-20) يعرض أزرار ونصوص تم إعدادها باستعمال برنامج exra3d.

الشكل(6-20)

إضافة برنامج Java Applets

هناك برامج صغيره تسمى Applet تم إعدادها باستعمال لغة Java ويوجد منها الكثير على شبكة الإنترنت ، وتوفر هذه البرامج (Applets) الكثير من الوظائف والعمليات مثل عمليات عرض الصور باستعمال أشكال مختلفة وكذلك الصوت والحركة و كثير من الأعمال الجذابة ، ولتصوير كمية البرامج Applets الكثيرة الموجودة على شبكة الإنترنت ادخل إلى شبكة الإنترنت وقم بعمل بحث عن كلمة Applet سوف تحصل على كم كبير منها .

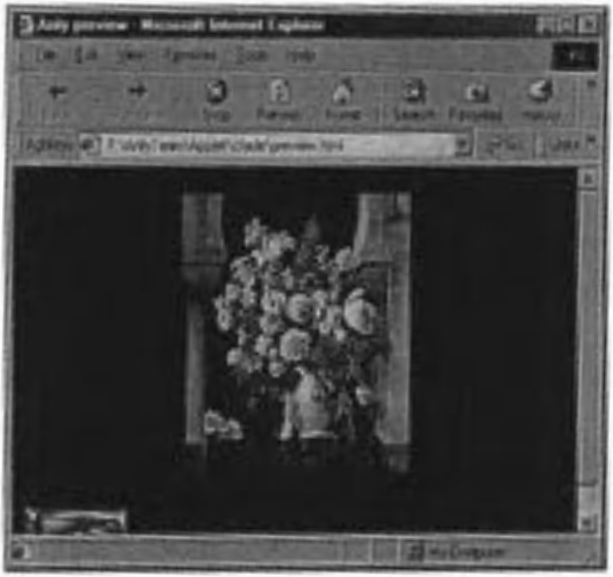
- ويوجد أكثر من طريقة لإضافة برنامج Applet

استعمال برنامج ANFY

برنامج ANFY يوفر كمية كبيرة من التأثيرات سواء على الصور أو النصوص باستعمال برنامج Java ANFY والشكل (7-20)يعرض مثال تم إعداده باستعمال برنامج Anfy.

الشكل(7-20)

والشكل (8-20) يعرض الشاشة الرئيسية للبرنامج.



الشكل(8-20)

ملحوظة هامة :

كل هذه الأدوات يمكنك الحصول عليها من الموقع download.com.

تحميل الموقع على شبكة الإنترنت Uploading site

بعد إعداد الموقع محلياً أي على الجهاز الخاص بك لابد من تحميله على أحد أجهزة الخادم (servers) المتصلة بالشبكة حتى يصبح متاح لجميع مستخدمي الإنترنت وهذه العملية تسمى uploading ولإجرائها لابد من حجز مساحة على أحد الأجهزة الخادمة (servers) بأحد طريقتين :

1. مساحة مدفوعة الأجر

وهذه هي الطريقة التقليدية حيث يتم الاتصال بأحد شركات خدمات الإنترنت ISP (Internet service provider) والاتفاق على حجز مساحة لتحميل الموقع عليها مقابل أجر شهري أو سنوي أو حسب الاتفاق.

2. مساحة مجانية free space

توجد كثير من الشركات التي توفر استضافة موقعك علي الخادم الخاص بها وذلك بإعطائك مساحة مجانية لتحميل الموقع عليه ولكنها محددة المساحة و محددة الإمكانيات .

ولكن علي الأقل تصلح للمواقع البسيطة وكذلك للتعلم والاختبار .

ومن العناوين التي توفر المساحات المجانية ما يلي:

```
www .freeservers.com
www .geocities .com
www .50megs .com
```

ويمكنك الحصول على الكثير منها بمجرد البحث عن كلمة Free space

حجز لمساحة

في هذه الفقرة نشرح لك كيفية استعمل العناوين التي توفر مساحة مجانية لحجز مساحة وكذلك كما يلي:

1- قم بالاتصال بشبكة الإنترنت.

2- اكتب العنوان www .geocities .com

تحصل علي الصفحة الرئيسية للموقع حاول اتباع الخطوات الموجودة لحجز مساحة .

وجرب استعمال الموقع 50meges.com فهو أسهل فى الاستعمال.

تحميل الموقع uploading

بعد إعداد الموقع وحجز المساحة لا يتبقى سوي تحميل هذا الموقع علي المساحة ويتم ذلك بطرق كثيرة منها:

1- استعمال Microsoft Web Publishing

2- استعمال برامج نقل الملفات FIP

3- استعمال Microsoft Web Publishing

وهو برنامج يأتي مع برنامج النوافذ يساعدك من خلال مجموعة من البيانات من تحميل موقعك ، ولتوضيح ذلك تابع ما يلي:

من برنامج النوافذ اختر start ثم programs ثم Microsoft Web Publishing

ثم Web Publishing Wizard تحصل علي أول شاشات البرنامج كما في الشكل (9-20).

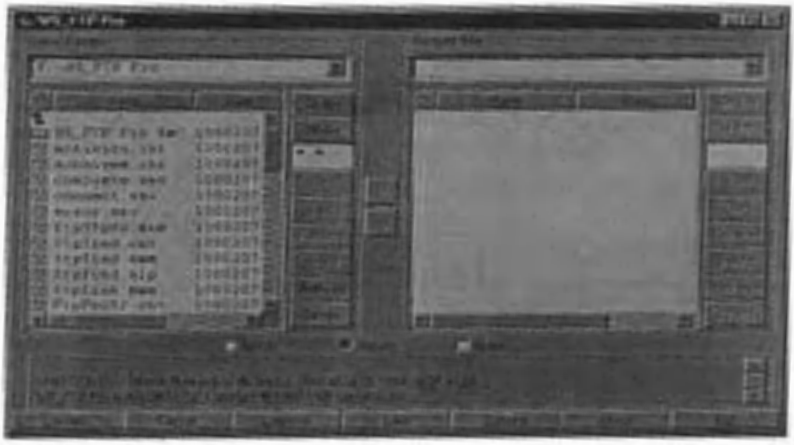
الشكل(9-20)

تابع الخطوات لتكون فكرة عن متطلبات التحميل.

استعمال برنامج نقل الملفات FTP

الحروف FTP اختصار للعبارة file transfer protocol وهو برتوكول نقل الملفات ،وهي برامج كثيرة تقوم بنقل ملفات موقعك من جهازك إلى جهاز الخادم (المساحة المحجوزة).

- ومن هذه البرامج برنامج WS FTP Pro الذي توجد منه نسخة تجريبية على الموقع Download.com والشكل (10-20) يعرض الشاشة الرئيسية.



الشكل(10-20)

هذا الشكل عبارة عن نصفين الأول يعرض محتويات الجهاز والثاني عليك أن تعرض فيه محتويات المساحة المحجوزة وتستخدم البرنامج لنقل الملفات من الجهاز إلى المساحة المحجوزة .

طرق كتابة برامج للإنترنت web programming

في الفقرات السابقة شرحنا كيفية إعداد مواقع للإنترنت ولكن لم نتناول بها أوامر برمجة وهناك الكثير من العمليات التي تحتاج لأوامر البرمجة مثل التأكد من مدخلات نموذج أو لتحقيق عمليات قواعد البيانات مثل حفظ البيانات بقاعدة البيانات وكثير من العمليات .

وتوجد لغات مختلفة لكتابة برامج للإنترنت وتقسمها كما يلي:

1- أوامر تسمى scripts.

2- لغات برمجة متكاملة.

1. أوامر تسمى Scripts

هناك مجموعة من اللغات التي تسمى scripts وكلمة scripts تطلق على مجموعة من الأوامر والقواعد التي لاتصل إلى مستوي اللغة فهي ليست لها شكل لبرنامج متكامل مثل اللغات (C, Pascal ,....) وليست لها بيئة عمل خاصة بها ولا يتم ترجمتها لتعطي برنامج تنفيذي ، وأوامر script تكتب داخل صفحة الإنترنت (HTML) ، وتنقسم أوامر scripts إلى ما يلي:

أ. أوامر scripts تنفذ عند المستخدم وتسمي client

وهي أوامر تكتب داخل ملفات HTML الخاصة بالصفحات وتنفذ من خلال عارض صفحات الإنترنت (Internet explorer أو Netscape) .

ومن أمثلة هذه اللغات Java script و VB script ووظيفتها التعامل مع المستخدم مثل التأكد من مدخلات.

ب. أوامر script تنفذ علي جهاز الخادم (server)

وتسمي server side لأنها تنفذ علي جهاز الخادم المحمل عليه ملفات الموقع .

ومن أمثلة هذه اللغات PHP، JSP، ASP وتستعمل هذه الأوامر في إجراء عمليات

علي جهاز الخادم مثل تسجيل بيانات في قاعدة بيانات موجودة علي جهاز الخادم أو للاستعلام عن بيانات أو غيره من العمليات التي تخص جهاز الخادم SERVER.

لغات البرمجة programming language

بالإضافة للأوامر script يوجد لغات برمجة لإعداد تطبيقات للإنترنت من أشهر هذه اللغات لغة Java ولغات visual studio .net مثل VB.net و vc.net و c#.

وتختلف هذه اللغات عن أوامر script بأن لها بيئة عمل لإعداد تطبيق متكامل يتم ترجمته و اختباره.

كتابة أوامر ASP

الحروف ASP هي اختصار للعبارة Active server pages (ومعناها صفحات الخادم الفعالة)، وهي كما أشرنا أوامر تكتب لتنفذ علي جهاز الخادم لتحقيق عمليات علي الخادم ولكتابة الأوامر مطلوب ما يلي:

1ـ توفير أحد برنامجي IIS, PWS

IIS يعمل مع برنامج النوافذ WIN NT أو WIN 2000

- PWS يعمل مع برنامج النوافذ WIN 98.
- والحروف IIS اختصار للعبارة Internet Information server.
- والحروف PWS اختصار للعبارة personal web server.
- وأهمية وجودها هو إنها المسئولة عن ترجمة أو تفسير أوامر ASP لجهاز الخادم (SERVER) وبدون وجودها تصبح أوامر ASP عبارة عن نصوص (TEXT) مكتوبة ليس لها معني .
- و برنامج IIS يأتي تلقائياً مع WIN 2000 ،ولكن مع WIN NT يجب إعداده وكذلك PWS مع WIN 98 يجب إعداده.

2ـ كتابة أوامر ASP

- كتابة أوامر ASP داخل ملف HTML.
- وضع الملفات في الفهرس المناسب وهو WWWROOT.

- لاختيار ملفات ASP يتم إنشاء فهرس بالاسم INPUB.
- تلقائياً و داخل الفهرس WWW وهو الفهرس التقليدي الذي ينظر إليه جهاز الخادم (server) عند استدعائه لذلك يتم وضع الملفات داخل الفهرس WWW داخل الفهرس INTPUB .

● ولتوضيح خطوات كتابة أوامر ASP تابع ما يلي:

1- قم بإعداد IIS أو PWS حسب برنامج النوافذ.

2- قم بكتابة أوامر ASP داخل ملف HTML كما في السطور التالية :

```
<HTML>
 <BODY>
<% responance .write
=Date
NOW
%>
</Body>
</HTML>
```

في هذه السطور :

1- قم بحفظ هذه الملفات في الفهرس WWW بالاسم A1.ATM

2- جرب استدعاء هذا الملف كما يلي:

افتح عارض صفحات الإنترنت internet explorer

في خانة العنوان اكتب السطور التالية :

```
http://localhost /A1.HTML
```

يتم استدعاء الملف A1 وعرض نتيجة التنفيذ كما بالشكل

الشكل

والسؤال هو ما معني LOCAL HOST ؟

هذه الكلمة لها معني هو استدعاء الملف A1 من الفهرس WWWROOT الموجود بهذا الجهاز محلياً (LOCAL) وهذا معناه معاملة الجهاز مرتين :

- الأولي: كجهاز خادم SERVER تحمل عليه الصفحة .
- الثانية: كجهاز مستخدم CLIENT ينادي الصفحة.

ويمكن استعمال

1- الكلمة localhost.

2- اسم الجهاز المحمل عليه الموقع (الصفحة).

3- الرقم 127.0.0.1.

تنشأ باستعمال HTML ولكن ذلك غير كافي لإعداد تطبيقات والتعامل مع المستخدم والتعامل مع قاعدة البيانات ،لذلك أضافت مايكروسوفت أوامر ASP التي وفرت التعامل مع المستخدم وكذلك حفظ البيانات في قواعد البيانات و التعامل معها .

وأضافت أيضاً asp.net مزايا أخرى منها :

- فصل ملفات html عن أوامر ASP مما سهل شكل الأوامر وكذلك سهولة تعامل مع أحداث الأدوات Events handler .
- إضافة أدوات controls جديدة تساعد في إعداد التطبيقات.
- إضافة ما يسمى web services.

في لغة Java

تعريفات هامة

هناك بعض التعريفات التي يجب الإلمام بها قبل دراسة لغة JAVA هي:

1- بيئة تشغيل الـJava JVM

2- مترجم برامج الـJava JIT

3- ما هي Java Applet Java Applet

4- ما هو تطبيق Java Java Application

5-ما هي مكتبة SDK, JDK

6-ما هي ال Servelt

7-ما هي JSP وتطبيقات الإنترنت

تعريفات مهمة

بيئة التشغيل JVM

الحروف JVM اختصار للعبارة Java virtual Machine أى ماكينة جافا الخيالية وهى فكرة قامت Java بإنشائها لتجعل لغة Java تعمل على جميع أو معظم أنظمة التشغيل، وتقوم الفكرة على إنشاء طبقة وسيطة (Software) وكأنها برامج تشغيل للبرامج Run Time لكل نظام تشغيل يتم إنزاله أولاً على الأجهزة بحيث تفهم هى برامج Java وتفسرها لنظام التشغيل ثم الجهاز وبهذا كان من مزايا لغة Java إنها تعمل على كثير من نظم التشغيل الموجودة بعد إعداد JVM الخاصة بمعظم التشغيل فلا يهم أركان البرنامج مكتوب لنظام النوافذ Windows أو نظام Unix المهم أن البرنامج يكتب ثم يحمل إلى الجهاز وعلى الجهاز يوجد JVM للنظام الموجود وبالتالى يعمل البرنامج.

ما هى Java Applet ؟

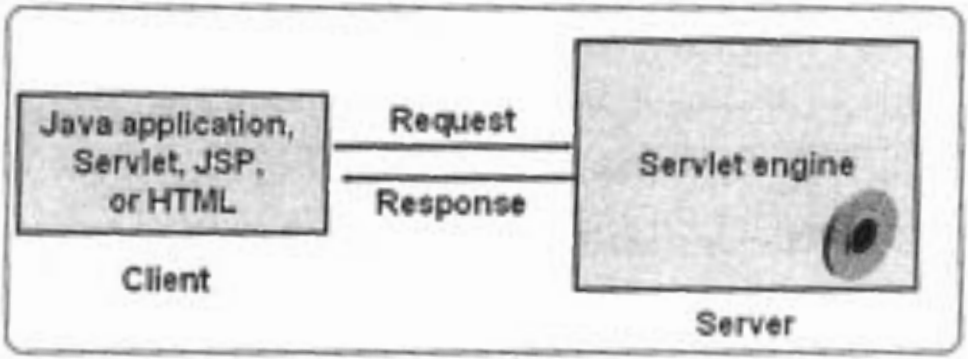
نوع من أنواع التطبيقات الذى صمم خصيصاً للإنترنت حيث يقوم المطور (Developer) بإعداد هذا البرنامج Applet ثم يتم استدعائه من خلال ملف HTMLبشرط تحميل برنامج Applet على Server الموجود عليه ملف HTML فيتم عرض هذا التطبيق من خلال صفحة الإنترنت عندما يستدعى المستخدم هذه الصفحة أما طريقة إنشاء Applet وطريق استدعائها من داخل ملف HTML فهذا ما سوف نشرحه فى الفقرات التالية.

ما هو تطبيق Java Application ؟

هو تطبيق يشابه التطبيقات التى يتم إنشائها بجميع لغات البرمجة الأخرى ليعمل مع نظم التشغيل بعيداً عن شبكة الإنترنت والمشهور عن لغة Java إنها تعد برامج للإنترنت ولكن غير المشهور أيضاً إنها توفر كثير من نقاط القوة فى إعداد تطبيق سواء مكتبى Desktop أو خاص بالشبكات Client / server مما شجع شركة ORACLE على تبنى اللغة واستعمالها فى أعداد التطبيقات من خلال الأداة Jdeveloper وقاعدة بيانات ORACLE.

ما هى ال Servlet ؟

هى صورة من صور فصائل الـ Java ،وهى صممت خصيصا لإعداد تطبيقات على شبكة الإنترنت ، والشكل التالى يوضح كيف تعمل فصائل الـ Servlet.



ما هى JSP وتطبيقات الإنترنت ؟

الحروف JSP اختصارا للعبارة :JavaServer Pages وهى أحد صور الـ Scripts التى يطلق عليها Server Side Script وبالطبع تستخدم لاعداد صفحات الانترنت ذات العمليات المتغيرة مثل عمليات قواعد البيانات وغيرها ، ويتم كتابة أكوادها داخل صفحات HTML مثلها مثل جميع الـ Scripts.

ما هى الـ JSTL ؟

هى مجموعة من العلامات tags جاهزة قامت شركة Sun بإعدادها فى مكتبات لغة Java لتسهيل كتابة الأوامر داخل صفحة الـ JSP.

ما هى الـ Custom Tags ؟

هى طريقة من الطرق المتقدمة لإنشاء علامات تخص المبرمج لتحقيق أغراض معينة يستعملها وقتما يشاء مما يسهل إعادة بناء التطبيقات ويحقق نظرية الـ Reusability.

ما هى الـ EJB ؟

طريقة لإعداد وحدات components تسهل التعامل مع قواعد البيانات وذلك من خلال وحدات تتصل بقاعدة البيانات وتوفر دوال التعامل معها.

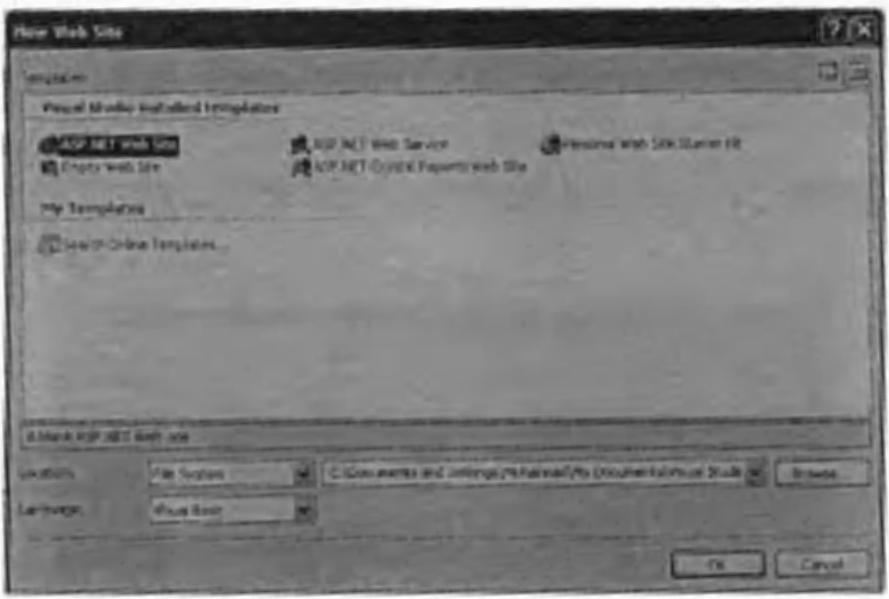
فى VB.NET ,C#

أول تطبيق ASP.NET

ملحوظة :

لإنشاء تطبيق ASP لابد من وجود IIS 4.0 على الجهاز الذي تعمل عليه وإذا كنت تستعمل WIN 2000 SERVER فلا توجد مشاكل لأنه يحتوى على IIS وهذا ما اشرنا اليه فى أول الفصل ولإعداد تطبيق Web باستعمال ASP.Net تابع الخطوات التالية.

1- اختر file ثم new ثم project ومن أنواع اللغات اختر visual basic projects وأيضاً من أنواع برامج VB اختر Web Application كما بالشكل (11-20) .



الشكل(11-20)

لاحظ أن البرنامج يحفظ بالموقع http://localhost لأن VS يعتبر الجهاز SERVER.

2- اضغط زر OK يقوم VS.NET بالاتصال بالخادم Server لينشئ التطبيق ، ثم تحصل على التطبيق وفيه صفحة خواليه من الأدوات ونص يشير إليها كما بالشكل (12-20).

الشكل(12-20)

لاحظ وجود الملف Web form1 ASPX في مربع المشروع والامتداد ASPX هو الامتداد الخاص بـ ASP.NET

1. اضغط التبويب HTML أسفل صفحة التصميم لحصل على أوامر HTML المقابلة للصفحة كما بالشكل (13-20).

الشكل(13-20)

2. اضغط تبويب Design مرة أخرى ثم اضغط مربع الأدوات TOOLBOX واختر web forms ومنها وقع أداة عنوان label وزر أوامر button كما تعودت.
3. اضغط زر الأمر مرتين واكتب السطر التالي :

```
label1. text = "hi from the button "
```

4. نفذ البرنامج عند هذه المرحلة تلاحظ ظهور browser وبه ملف البرنامج جرب الاضغط علي زر الامر ولاحظ النتيجة كما بالشكل (14-20).

الشكل(14-20)

5. اختر قائمة view ثم view source يظهر سطر HTML.

ما هي server controls ؟

أضافت ASP.NET مجموعة من الأدوات أطلقت عليها Server controls و أطلق عليها هذا الاسم لان أوامر هذه الأدوات تنفذ من قبل الخادم server و هي التي تصلح لكتابة أوامر عكس صفحة أدوات HTML التي توفر مجموعة من الأدوات لتكوين صفحة HTML ثابتة فقط مثل الجداول وغيرها .

ولتوضيح الفرق تابع ما يلي:

1. اضغط مربع الأدوات toolbox واختر HTML واختر منها زر أمر BUTTON ووقعه علي الصفحة الجديدة .
2. اضغط مربع الأدوات TOOLBOXواختر Server control واختر منها زر أمر

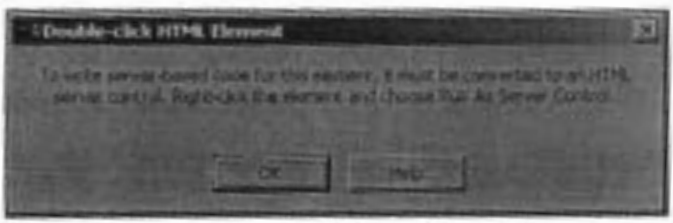
button و وقعه علي الصفحة الجديدة.

3. اضغط تبويب HTML في نهاية الصفحة لإظهار أوامر HTML المقابلة لهذه الصفحة تلاحظ وجود السطور التالية لوصف زري الأمر :

```
1: <INPUT Type = button value = button>
2: <ASP :Button ID = Button2 rural = "server" text = "button">
</ASP :Button>
```

في هذه السطور :

- في السطر رقم 1 تم إضافة سطر HTML عادى لتعريف زر أمر وهو باستعمال input Type و هذا هو السطر المقابل لزر الأمر الخاص لمجموعة HTML.
- في السطر رقم 2 تم إضافة ما يشير إلي أن زر الأمر من مجموعة SERVER حيث بدأ
- ب ASP كما أضيف له العبارة RUNAT أي سوف يتم تنفيذ أوامره عنده ،وكانت تساوي SERVER أي عند SERVER.
- وعند محاولتك كتابة أوامر في زر HTML تحصل علي رسالة اعتراض تخبرك بأن زر الأمر لابد أن يكون SERVER control جرب ذلك تحصل علي الرسالة التالية شكل (15-20):



الشكل(15-20)

والتي تخبرك بضرورة تغيير النوع بهذا الأسلوب يمكنك أضافة أدوات server وكتابة الأوامر المطلوبة و تحقيق أفكار تطبيقات كثيرة .

إنشاء برنامج الآلة الحاسبة

لتوضيح سهولة إعداد التطبيقات من خلال VB.Net نقوم بإعداد برنامج الآلة الحاسبة الذى قمنا بإعداده فى أول الكتاب ولكن فى هذه الحالة نعده داخل صفحة Web

أى يصبح Web Application وذلك كما فى الخطوات التالية:

1- قم بإنشاء تطبيق جديد من النوع Asp Application.

2- قم بتصميم شكل نموذج الآلة الحاسبة ليصبح كما فى الشكل (16-20).

الشكل(16-20)

3-قم بكتابة سطور أوامر أزرار أوامر العمليات كما يلى:

- عملية الجمع

اكتب السطر التالى:

```
TextBox4.Text = Val(TextBox1.Text) + Val(TextBox2.Text)
```

- عملية الطرح

اكتب السطر التالى:

```
TextBox4.Text = Val(TextBox1.Text) - Val(TextBox2.Text)
```

- عملية الضرب

اكتب السطر التالى:

```
TextBox4.Text = Val(TextBox1.Text) * Val(TextBox2.Text)
```

- عملية القسمة

اكتب السطر التالى:

```
TextBox4.Text = Val(TextBox1.Text) / Val(TextBox2.Text)
```

4-قم بتنفيذ البرنامج عند هذه المرحلة لحصل على نتيجة التنفيذ كما فى الشكل (17-20).

الشكل(17-20)

من هذا المثال نلاحظ أن عملية إعداد تطبيقات إنترنت من خلال VB.Net أصبحت سهلة وتقارب إعداد التطبيقات العادية Windows Applcations مع وجود بعض الاختلافات ، وبنفس قواعد Windows Application تستطع الاستفادة من الخصائص المتاحة للأدوات وكذلك برمجة الأحداث Event Programing حيث سهل ذلك البرمجة عن الأسلوب التقليدى للغة ASP القديمة وهو أسلوب Request/Response.

أداة الجدول Table

قمنا سابقا يا صديقى بتوضيح الفرق بين الـ Server Controls و الـ HTML Controls وقمنا أيضا باستعراض الأدوات ومقارنتها فى كلا النوعين ووجدنا أن هناك أدوات جديدة فى الـ Server Controls لم تكن موجودة فى الـ HTML Controls ،وأدوات أخرى موجودة فقط فى الـ HTML Controls ،ونوع ثالث موجود فى الاثنين ومن هذا النوع الأداة التى معنا الآن وهى أداة الجدول إذن فمعنى هذا الكلام أن هناك أداة اسمها Table موجودة فى الـ Server Controls والتى تمثلها Web Forms وهناك أيضا أداة اسمها Table وهى موجودة فى فى الـ HTML Controls والتى تمثلها HTML ونحن فى هذا المجال سوف نشرح الأداة Table الموجودة فى الـ Server Controls أما بالنسبة للنوع الآخر فسوف نقوم بعرضها فى الملحق الخاص بالـ HTML والذى سنتناوله فى آخر الكتاب.

ملحوظة :

إذا أردنا إضافة جدول وتحديد عدد أعمدته وعدد صفوفه أثناء التصميم فما علينا إلا إضافة الأداة Table والتى تخص HTML Controls أما بالنسبة للأداة Table والموجودة فى Server Controls فلا يمكن تحديد عدد أعمدته وعدد صفوفه أثناء التصميم بل لابد من تحديدها خلال التنفيذ بواسطة الأوامر وذلك من خلال الكائن TableRow والذى يمكن من خلاله إضافة صف إلى الجدول والكائن TableCell والذى يمكننا من إضافة خلية إلى الصف .

و سوف نقوم فى السطور التالية بشرح بعض خصائص الأداة Table.

الخاصية BorderColr

وهى متعلقة بلون الحدود الخاصة بالجدول .

الخاصية BorderStyle

وهى متعلقة بنوع الحدود الخاصة بالجدول ولكن كى تظهر أهمية كلا الخاصيتين. فلابد من شرح خاصية هامة جدا وهى الخاصية GridLines.

الخاصية GridLines

وهى الخاصة برسم الحدود بين أعمدة الجدول وصفوفه

ومن المواقع المهمة جدا لتعلم سكربتات الإنترنت الموقع

http://www.w3schools.com/

الذى يوفر شرح لجميع سكربتات الانترنت كما فى الصفحة الرئيسية فى الشكل

حاول تصفح هذا الموقع المهم وجرب تتبع الدروس التعليمية للموضوعات المختلفة مثل

HTML Tutorials
Learn HTML
Learn XHTML
Learn CSS
Learn TCP/IP

XML Tutorials
Learn XML
Learn XSL
Learn XSLT
Learn XSL-FO
Learn XPath
Learn XQuery
Learn XLink