الباب الأول

أساسيات البرمجة المعتمدة على الكائنات
ولغة فيجوال بيسك.نت
OBJECT-ORIENTATION AND
VB .NET FUNDAMENTALS

- الفصل الأول: تطوير نظم الحاسب المعتمدة على الكائنات بواسطة لغة فيجوال بيسك .نت
- الفصل الثاني: بيئة تطوير برامج فيجوال ستديو .نت
- الفصل الثالث: أساسيات لغة VB .NET
- الفصل الرابع: البرمجة بلغة VB .NET باستخدام مكتبة الأصناف الجاهزة
- الفصل الخامس: التحليل والتصميم الموجه لتقنية الكائنات

الفصل الأول

تطوير نظم الحاسب المعتمدة على الكائنات
بواسطة لغة فيجوال بيسك.نت
OBJECT-ORIENTED SYSTEM DEVELOPMENT WITH VB .NET

أهداف الفصل:

- تطوير نظم الحاسب بتقنية الكائنات الموجهة (OO development).
- التعرف على لغة فيجوال بيسك .نت (VB .NET).
- التعرف على مفاهيم التقنية المعتمدة على الكائنات.
- معرفة أهمية وفوائد التطوير التقنية المعتمدة على الكائنات.
- تقديم الأسلوب المتبع داخل هذا الكتاب لتطوير النظم بواسطة الكائنات.

تشمل عملية تطوير نظم المعلومات المعتمدة على الكائنات (Objectoriented) على مراحل التحليل والتصميم والبرمجة مستخدماً لغات برمجة وتقنيات تعتمد على الكائنات الموجهة، وعادة نطلق على هذه العملية الاسم المختصر OO أو تقنية OO. أصدرت شركة مايكروسوفت مؤخراً إطار عمل .نت (.NET framework) الذي يشجع على تطوير النظم بتقنية OO ولكي يثبت أيضاً أن تقنية OO أصبحت الاتجاه السائد في مجال تطوير نظم المعلومات. ومما لا شك فيه أن إعادة تصميم لغة الفيجوال بيسك وتطويرها لإصدار لغة الفيجوال بيسك .نت سوف يسهل على العديد من المبرمجين استخدام تقنية OO في تطوير نظم المعلومات لأن لغة الفيجوال بيسك .نت تحتوي الآن على جميع المقومات والمفاهيم التي تمكن المبرمج من تطوير نظم معلومات معتمدة على الكائنات التي كانت تُقدم فقط في السابق بواسطة لغات قوية مثل لغة C++ ولغة Java.

كثيراً ما يختلف مفهوم كل من تقنية OO والتطوير بواسطة تقنية OO (OO development) لدى العديد من المبرمجين وذلك بناء على اختلاف خبراتهم وخلفياتهم حيث يرى البعض أن تقنية OO تعني واجهة المستخدم الرسومية (GUI) ، ويرى البعض أن تقنية OO تعني أي شيء له علاقة بنظم العميل /الخـادم (client/server systems) أو تطبيقات الإنترنت. ويرى آخرون أن تقنية OO تعني نظم الحاسب التي يتم تطويرها بلغات برمجة تعتمد على الكائنات مثل كل من لغة C++ ولغة Java بغض النظر عن التطبيق ذاته.

في الحقيقة إن تقنية OO تعني استخدام تقنية الكائنات (Object-Oriented) في كل من مرحلة تحليل النظم OO Analysis (OOA) ومرحلة التصميم OO Design (OOD) ومرحلة البرمجة OO Programming (OOP) ، ومن ثم إن تقنية OO لا تعني تطوير واجهة مستخدم رسومية (GUI) أو علاقة العميل بالخادم أو استخدام لغة مثل لغة C++ (أي أن تعلم لغة برمجة تعتمد على الكائنات غير كافٍ لكي تجعلك مبرمجاً في مجال تقنية الكائنات). وأخيراً إن تطوير البرامج باستخدام تقنية OO يعني أسلوب تفكير، وبمعنى آخر إن تقنية OO تربط جميع مراحل تطوير نظم المعلومات من تحليل وتصميم وبرمجة معاً.

سوف تعرض داخل صفحات هذا الكتاب عملية تطوير نظم المعلومات المعتمدة على الكائنات بشكل شامل ومتكامل بغرض التعرف على تقنية الكائنات مع التركيز على لغة فيجوال بيسك .نت حيث نفترض أن لدى القارئ خلفية بسيطة في مجال البرمجة ، ولا يتطلب الأمر معرفة أي خبرة عن تقنية الكائنات أو عن لغة الفيجوال بيسك ، وبالتأكيد إذا توفر لدى القارئ معرفة عن مجال التحليل والتصميم وإدارة قواعد البيانات سيكون مفيداً جداً.

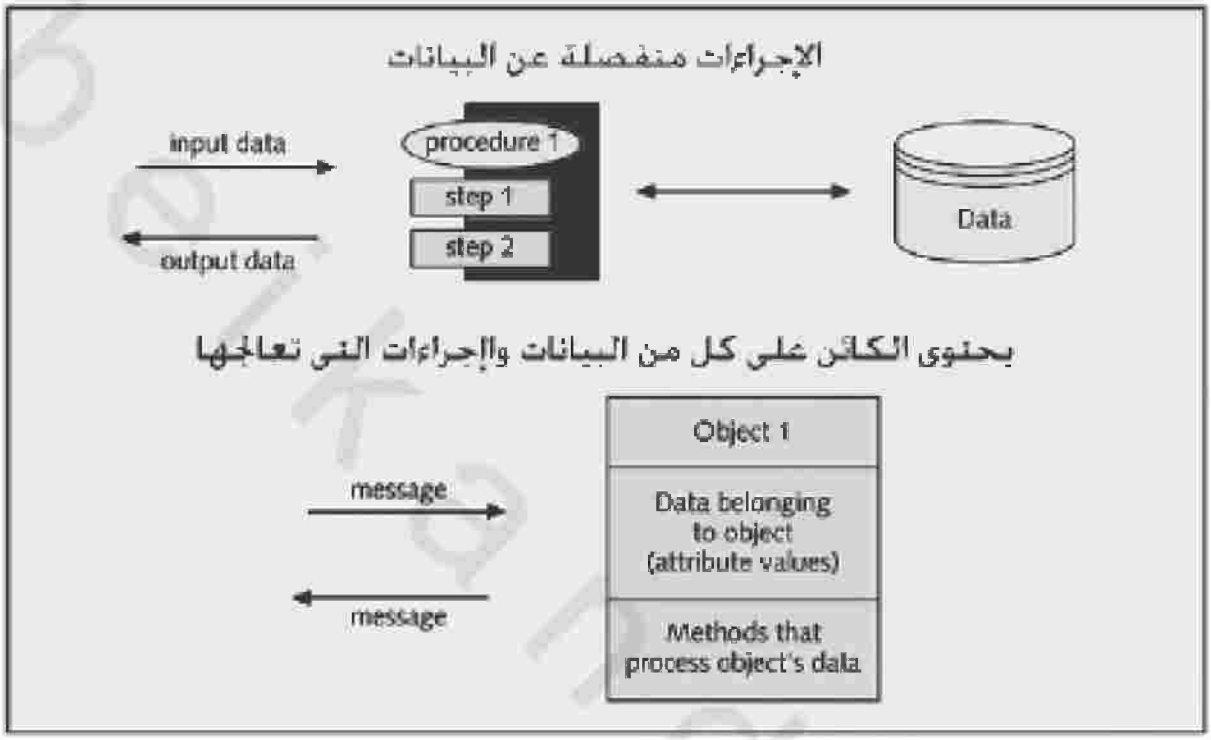
عزيزي القارئ سوف تتعلم كيفية تطوير نظم المعلومات من البداية إلى النهاية مستخدماً تقنية الكائنات (Object-Oriented) حيث يقدم الكتاب أساساً لكل من عملية التحليل (Analysis Process) وعملية التصميم (Design Process) بواسطة الكائنات ، كما يقدم أيضاً عملية البرمجة (Programming Process) من خلال لغة فيجوال بيسك .نت.

التعرف على تطوير النظم باستخدام تقنية الكائنات ولغة VB .NET

Understanding OO Development and VB .NET

عادة ما يتم مقارنة تطوير النظم مستخدماً تقنية OO مع الطريقة التقليدية (Traditional Approach) لتطوير نظم المعلومات والتي يطلق عليها عملية التطوير الإجرائي (Procedural Development) والتي تعتمد على الإجراءات (Procedures). يتم تعريف نظام المعلومات من وجهة نظر تقنية OO على أنه مجموعة من الكائنات (Objects) التي تعمل سوياً لإنجاز مهام النظام حيث أن الكائن يستطيع أن ينفذ بعض المهام عندما يتم طلبها منه بواسطة كائن آخر ، مع العلم أن كل كائن يحتفظ ببياناته الخاصة. وعلى الجانب الآخر ، فإن الطريقة التقليدية تُعرِّف النظام على أنه

مجموعة من الإجراءات التي تتعامل مع البيانات المخزنة بعيداً عن الإجراءات حيث يتم تعديل بيانات أو إنشائها عند تنفيذ أحد هذه الإجراءات. يوضح الشكل رقم (١.١) الفرق بين كل من تقنية OO والتقنية الإجرائية.



الشكل رقم (١.١)- مقارنة بين تقنية الكائنات الموجهة والتقنية الإجرائية.

وكما هو واضح من الشكل رقم (١.١) فإن الإجراء يستقبل البيانات ثم يقوم بمعالجتها، ومن ثم تعديل البيانات المخزنة في ملفات منفصلة. هذا على العكس من تقنية OO حيث يستقبل الكائن رسالة (Message) من كائن آخر، ثم يقوم بمعالجة بياناته الداخلية وتعديلها. سيتم توضيح أهمية هذا الفرق من خلال مراحل تطوير النظام (التحليل والتصميم والبرمجة) في الفصول المقبلة.

البرمجة المعتمدة على الكائنات
Object-Oriented Programming

يعتقد معظم الناس أن تطوير البرامج مستخدماً الكائنات اتجاهاً حديثاً، ولكن على العكس من ذلك فإن هذا الاتجاه يرجع إلى عقود مضت حيث بدأ هذا الاتجاه في حقبة الستينيات في النرويج بتطوير لغة برمجة سيمولا (Simula) وهي أول لغة تم تصميمها لتنفيذ برامج المحاكاة(Simulation Programs) لأن برامج المحاكاة تتطلب أسلوب

برمجة مختلفاً عن الطريقة الإجرائية (Procedural Process). عادة ما تتعلق المحاكاة بالكائنات مثل السفن والطائرات ومحاكاة العملاء الذين ينتظرون في صفوف داخل بنك لإنهاء معاملاتهم. ولذلك يجب أن يتم تصنيف هذه الكائنات إلى أصناف (Classes) والتي تحدد صفات كل كائن (بيانات) والسلوك الذي يسلكه كل كائن بشكل مستقل في برامج المحاكاة.

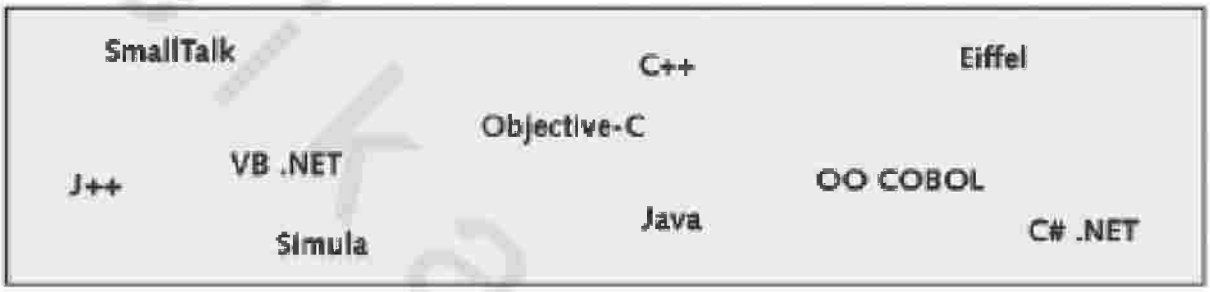
فعلى سبيل المثال، إن محاكاة عمل بنك ما يتطلب تعريف الصنف Customer (العميل) حيث أن جميع العملاء الذين يتعاملون مع البنك ما هم إلا كائنات من هذا الصنف، ومن ثم يستطيعون أن يدخلوا البنك، وينتظروا في صف، ويتقدمون في الصف واحداً تلو الآخر في اتجاه موظف البنك، وكذلك تعتبر صفوف الانتظار نوعاً آخر من الكائنات (كائنات من الصنف Row) والتي لديها القدرة على إضافة عميل وتحريك العملاء في اتجاه شباك الموظف. يجب تحديد بعض قيم الاحتمالات قبل بداية تشغيل برامج المحاكاة مثل عدد العملاء المتوقع دخولهم البنك في الساعة الواحدة والزمن المتوقع لخدمة العميل الواحد. يدخل العملاء البنك عند تشغيل برنامج المحاكاة، ثم ينتظرون في أقصر الصفوف طولاً، ثم يتقدمون في الصفوف بناء على أرقام عشوائية يتم إصدارها بواسطة دالة التوزيع الاحتمالي.

وتظهر فائدة البرمجة المعتمدة على الكائنات عندما يعمل برنامج المحاكاة في ظروف مختلفة لكي يحدد أكثر صنف انتظار عدداً وأطول فترة زمنية ينتظرها العميل، فعندئذ نرى أن الكائنات تتفاعل بعضها مع بعض بشكل مختلف. ومن ثم يستطيع متخذو القرار بالبنك اتخاذ قرارات مناسبة طبقاً لنتائج برنامج المحاكاة مثل تحديد عدد موظفي البنك خلال فترات البنك المختلفة.

يجب الاعتراف أن البداية الحقيقية لتقنية OO بدأت عند تطوير لغة البرمجة SmallTalk بواسطة العالم "آلان كي" Alan Kay وزملائه في مركز أبحاث شركة زيروكس (زيروكس بارك) في أوائل السبعينيات حيث كانت هذه اللغة أول لغة تدعم تقنية OO والتي يتم استخدامها في تطوير برامج مختلفة الأغراض. ركز العالم "كي" في هذا الوقت على إيجاد بيئة تطوير برامج تعتمد على الكائنات (لغة SmallTalk) والتي جعلت الكائنات تتفاعل مع المستخدم وتتفاعل مع بعضها لإنجاز مهام النظام. قد تم استخدام هذه اللغة في تطوير كل من واجهة المستخدم الرسومية (GUI)، ونظم المعلومات مثل نظم معالجة أعمال البنوك التي تساعد العملاء على إجراء عملية إيداع الأموال إلى حساباتهم، وإجراء عملية سحب الأموال من حساباتهم الجارية حيث اشتملت هذه النظم على كائنات العملاء، وكائنات الحسابات، إلخ.

بعد ذلك ظهرت العديد من لغات البرمجة والتي تعتمد على الكائنات مثل لغة Objective-C، ولغة Eiffel، واللغة الأكثر شهرة لغة C++ والتي تعتبر امتداداً للغة C الإجرائية التي كانت مستخدمة بكثرة مما ساهم في انتقال العديد من المبرمجين إلى البرمجة بأسلوب البرمجة المعتمدة على الكائنات، وهذا جعل لغة C++ اللغة

الرائدة للبرمجة المعتمدة على الكائنات، وذلك بالرغم أن لغة C++ تدعم أيضاً أسلوب البرمجة الإجرائية التي ما زال يتبعها بعض المبرمجين في تطوير برامجهم. أصدرت شركة مايكروسوفت إصدار C++ كجزء من إصدار فيجوال ستوديو (Visual Studio) من عدة سنوات والذي يحتوي على العديد من اللغات والبرامج الأخرى، كما أصدرت عدة شركات أخرى العديد من اللغات ذات الغرض الواحد مثل الإصدار الخاص من لغة COBOL الذي يدعم البرمجة المعتمدة على الكائنات. يوضح الشكل رقم (١.٢) لغات البرمجة المختلفة التي تدعم البرمجة المعتمدة على الكائنات.



الشكل رقم (١.٢). بعض لغات تقنية الكائنات الموجهة.

وبعد ذلك أصدرت شركة صن مايكروسيستمس (Sun Microsystems) عام ١٩٩٥م لغة Java التي تعتمد فقط على الكائنات والتي تشبه لغة C++ في صيغة كتابة الأوامر، ولكن تتميز بصفات أخرى تجعلها مناسبة لتطوير تطبيقات الإنترنت (Web-based Applications) مثل تحميل برامج Applets على شبكة الإنترنت وتشغيلها على أي بيئة حاسب آلي، ثم أصدرت شركة مايكروسوفت إصداراً من لغة Java أطلقت عليه اسم J++. ولأن لغة Java تعتمد على الكائنات فقط ويمكن أن تستخدم لتطوير برامج تعمل على أي بيئة بما في ذلك تطبيقات الإنترنت، فإن لغة Java أصبحت خياراً ممتازاً لتعليم تطوير التطبيقات المعتمدة على الكائنات.

أصدرت شركة مايكروسوفت في الآونة الأخيرة إصداراً آخر منافساً للغة Java كجزء من إصدار "فيجوال ستوديو .نت" أطلقت عليه اسم C# (ينطق سي شارب)، هذا بالإضافة إلى إصدار لغة VB .NET التي تعتمد على الكائنات مما جعل شركة مايكروسوفت تأمل أن تسيطر على كلٍّ من تطوير التطبيقات المعتمدة على الكائنات وتطبيقات الإنترنت، ومن ثم فإن لغة VB .NET أصبحت خياراً ممتازاً أيضاً لتعليم تطوير التطبيقات المعتمدة على الكائنات.

إطار عمل .نت ولغة فيجوال بيسك .نت لشركة مايكروسوفت

The Microsoft .NET Framework and VB .NET

احتوى الإصدار فيجوال ستوديو .نت الجديد (Visual Studio .NET) التي أصدرته شركة مايكروسوفت على إطار عمل .نت (.NET Framework) والذي وصفته الشركة على أنه منصة تطوير جديدة ليسهل عملية تطوير البرامج وخصوصاً في البيئة ذات الطبيعة الموزعة (Distributed Environment) مثل شبكة الإنترنت. يتكون هذا الإطار من مكونين أساسيين اللذين يقدمان للمبرمج مرونة عالية أثناء عملية التطوير :

أولاً: مكون تنفيذ اللغات المشترك (Common Language Runtime) ويختصر بالمصطلح CLR والذي يدير شفرة البرنامج أثناء التنفيذ، وأيضاً يقدم خدمات جوهرية مثل إدارة الذاكرة، والتحقق من أمان شفرة البرنامج، وترجمة البرنامج إلى لغة الآلة (Machine Language)، هذا بالإضافة إلى خدمات أخرى عديدة والتي من شأنها أن توفر وقت المبرمج وتساعده على إنتاج برامج ذات جودة عالية مع المكون CLR. يمكن تطوير أجزاء مختلفة من البرنامج باستخدام عدة لغات (مدعومة بإطار عمل .نت) أثناء وقت تصميم البرنامج (Design time)، ولذلك يستطيع المبرمجون أن يتعاونوا في كتابة برنامج كبير بعدة لغات مختلفة من اختيارهم اعتماداً على خبراتهم مثل لغة VB .NET، ولغة C++، ولغة C# .NET، ولا يزالون يستفيدون من إمكانات مكون CLR الذي يضمن لهم تكامل أجزاء البرنامج وتشغيلها بشكل صحيح.

يرجع هذا التكامل إلى إنشاء لغة وسيطة (Intermediate Language) بين لغة البرمجة ولغة الآلة حيث يتم ترجمة جميع لغات البرمجة التي تعمل تحت بيئة .نت (مثل لغة فيجوال بيسك .نت) إلى هذه اللغة بواسطة مترجم كل لغة برمجة على حده، ثم تتم ترجمة اللغة الوسيطة الناتجة بواسطة المكون CLR إلى لغة الآلة. ولتنفيذ ذلك يستخدم مكون CLR مرحلة تالية من الترجمة أثناء تنفيذ البرنامج تسمى الترجمة أثناء التنفيذ (Just-in-time compilation) والتي تحول اللغة الوسيطة الناتجة بواسطة مترجم اللغة إلى لغة الآلة، ومن ثم يؤدي استخدام اللغة الوسيطة إلى تكامل لغات البرمجة المختلفة داخل البرنامج الواحد، كما يقدم المكون CLR أيضاً ميزة تنفيذ البرنامج على أكثر من بيئة الحاسب الآلي (Platform) دون إعادة ترجمة البرنامج مرة أخرى.

ثانياً: مكتبة أصناف إطار عمل .نت التي تقدم للمبرمج عدداً هائلاً من الأصناف سابقة التعريف (Predefined classes) والتي يمكن استخدامها في العديد من التطبيقات المتوافقة مع المكون CLR. يستطيع المبرمج بأي لغة من لغات .نت أن يستفيد من هذه المكتبة لإضافة وظائف هائلة للبرنامج الخاص به. تتنوع هذه الأصناف طبقاً لوظيفتها حيث توجد الأصناف المسؤولة عن إدارة النصوص، وأصناف معالجة المجموعات، وأصناف الاتصال بقواعد البيانات، وأصناف إدارة الملفات. عندما يتعلم المبرمج كيفية استخدام أحد أصناف هذه المكتبة مستخدماً لغة ما، فيمكن تطبيق هذه الخبرة مع أي لغة أخرى تعمل مع بيئة .نت. كما تقدم هذه

الأصناف وظائف خاصة مثل أصناف برامج Console، وأصناف برامج الويندوز ذات الواجهة الرسومية (GUI)، وتطبيقات الإنترنت.

لقد تحقق طموح شركة مايكروسوفت في تقديم إطار عمل .نت داخل إصدار المنتج التسويقي Microsoft Visual Studio .NET في عام ٢٠٠٢م، حيث تكامل كل من مكون CLR ومكون مكتبة إطار عمل .نت مع البيئة المتكاملة لتطوير فيجوال ستوديو .نت، وذلك لتوفير مجموعة متكاملة من اللغات والأدوات للمبرمج من أجل تطوير تطبيقات تعتمد على تقنية الكائنات وتطوير تطبيقات الإنترنت.

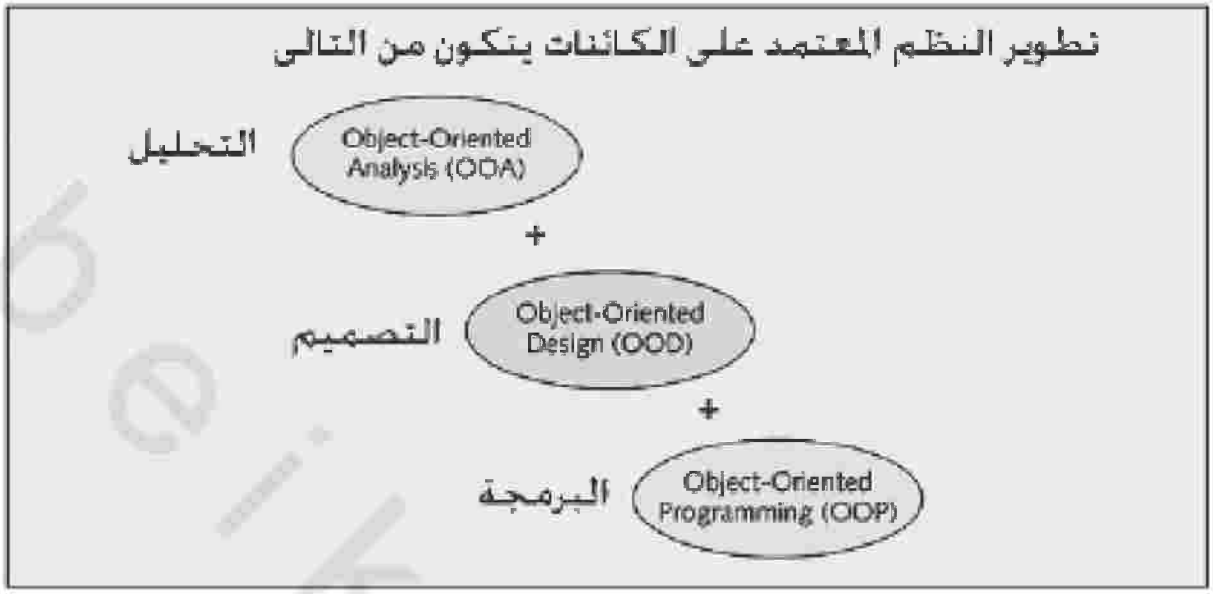
لقد قررت شركة مايكروسوفت أن تشمل لغة فيجوال بيسك (Visual Basic) مع الإصدار .نت، ولكن تحتاج هذه اللغة المشهورة والبسيطة (وهي امتداد للغة البيسك ولغة كويك بيسك التي ظهرت عام ١٩٩١م) إلى تغيير جذري لتتناسب مع الفكر المقدم في بيئة عمل .نت حيث كان في السابق تصدر هذه اللغة بشكل منفصل. تميزت لغة فيجوال بيسك بسهولة تطوير تطبيقات الويندوز لاحتوائها على عناصر واجهة الويندوز مثل النماذج (Forms) والأزرار (Buttons) والعناوين (Labels)، والتي يتم التعامل معها عن طريق سحبها ووضعها داخل نماذج الويندوز. كما اشتهرت أيضاً هذه اللغة بتقنية البرمجة المعتمدة على الأحداث (Event-driven programming) لمعالجة أحداث بيئة الويندوز مثل حدث الضغط على الزر الأيسر للماوس (OnClick).

ظلت لغة فيجوال بيسك اللغة الأسرع تعلماً والأسرع تطويراً لتطبيقات الويندوز، ولكن لم تعتمد لغة فيجوال بيسك حتى إصدار لغة فيجوال بيسك ٦ (أي حتى عام ٢٠٠٢م) بشكل كامل على تقنية الكائنات، كما ظلت لغة فيجوال بيسك تشتمل على بعض القيود والمشاكل التي عانى منها مبرمجو كل من لغة C++ ولغة C.

أما الآن فإن لغة فيجوال بيسك .نت تعتمد على تقنية الكائنات بشكل كامل وتستفيد من مكون CLR ومكتبة أصناف إطار عمل .نت مثلها مثل بقية لغات إصدار .نت، ولكن لكي تحتوي لغة فيجوال بيسك .نت على جميع هذه الإمكانات، تم إعادة بنائها من البداية وتغيير العديد من أجزائها والتي جعلت مبرمجي فيجوال بيسك يستطيعون أن يطوروا برامجهم معتمدين على تقنية الكائنات مثلهم مثل مبرمجي لغة Java ولغة C++، ولذلك يبدو أن مستقبل لغة فيجوال بيسك .نت سيكون أكثر ازدهاراً.

التحليل والتصميم المعتمد على الكائنات
Object-Oriented Analysis and Design

لقد ذكرنا سابقاً أن تطوير النظم المعتمدة على الكائنات لا يتعلق فقط بمرحلة البرمجة فحسب بل يتعلق بكلٍ من مرحلة التحليل ومرحلة التصميم ومرحلة البرمجة كما هو واضح في الشكل رقم (١.٣).



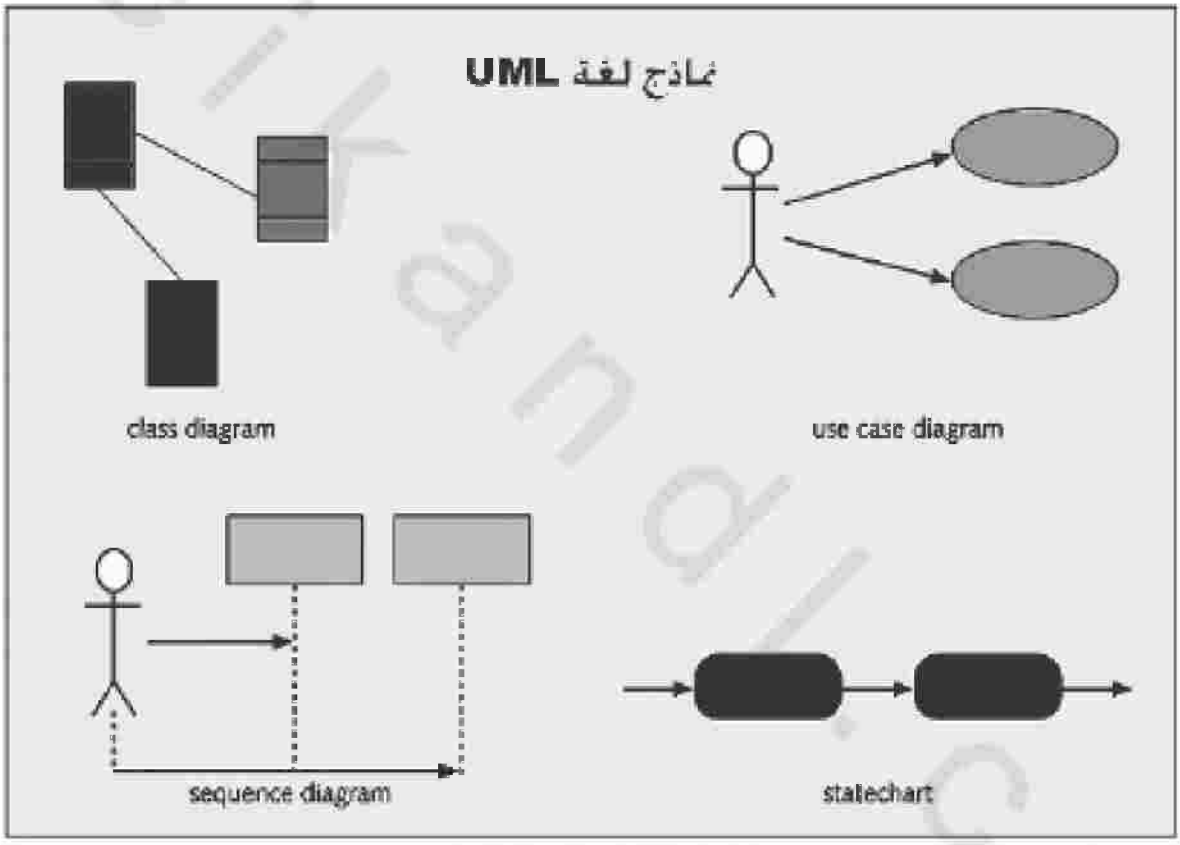
الشكل رقم (١,٣). تطوير النظم المعتمد على تقنية الكائنات الموجهة.

إن زيادة الاهتمام بتطوير نظم الحاسب مستخدماً تقنية OO في بدايات الثمانينات أدى إلى زيادة الاحتياج إلى أساليب وتقنيات تساعد مصممي النظم على تحليل تلك النظم وتصميمها حيث ظهر في نهاية حقبة الثمانينات ما يسمى التحليل المعتمد على الكائنات (Object-Oriented Analysis) والتصميم المعتمد على الكائنات (Object-Oriented Design). اتجه كثير من العلماء لوضع مفاهيم لمساعدة مصممي النظم في تحليل النظم وتصميمها حيث اتجه العالم "إد يوردن" Ed Yourdon إلى وضع أساليب وتقنيات لكل من التحليل التركيبي والتصميم التركيبي ، واتجه العالم "جايمس مارتن" James Martin إلى هندسة تقنيات تطوير نظم المعلومات كمحاولة لتحسين كل من التحليل والتصميم التركيبي ، وكذلك قام كل منهما بتأليف كتب في مجال التحليل المعتمد على الكائنات ومجال التصميم المعتمد على الكائنات. كما اقترح علماء آخرون أساليب أخرى للتحليل والتصميم المعتمد على الكائنات من واقع عملهم في مجال الصناعة حيث أنشئوا اللغة المثالية المستخدمة في تحليل نظم المعلومات وتصميمها والمعتمدة على الكائنات وعرفوها ، وأطلقوا عليها اسم لغة Unified Modeling Language (تختصر باسم UML).

عزيزي القارئ سوف تتعلم في هذا الكتاب كيفية تصميم بعض نماذج لغة UML ، وذلك من خلال شرح تطوير نظم المعلومات. تعتمد لغة UML على تقنية "الاعتماد على النماذج" (Model-driven approach) في كل من مرحلة التحليل والتصميم حيث يجب على المصمم أن ينشئ نماذج رسومية لكل من متطلبات النظام وتصميمه. تتكون هذه النماذج من أشكال هندسية مثل الخط والمربع والشكل البيضاوي التي توضح كيف

ينجز النظام مهامه ومما يتكون. إن توحيد هذه الأشكال يسهل على مطوري نظم المعلومات التواصل أثناء عملية تطوير النظم.

مع العلم أن الطريقة التقليدية لكل من التحليل والتصميم تستخدم أيضاً تقنية "الاعتماد على النماذج" (Modeldriven approach) حيث يجب إنشاء نماذج رسومية مثل نماذج تدفق البيانات (Data Flow Diagrams)، وعلى أي حال فإن لغة UML تستخدم أشكال نماذج مختلفة عن النماذج السابقة مثل نموذج Use case diagrams، ونموذج Class diagrams، ونماذج أخرى كما هو واضح في الشكل رقم (١.٤).



الشكل رقم (١.٤). نماذج لغة UML.

إن تحليل نظم المعلومات المعتمدة على الكائنات وتصميمها لا يتطلب فقط تصميم نماذج، بل إن دورة حياة تطوير النظم تتطلب إطار عمل لإدارة المشاريع والتي تحدد مراحل إنجاز النظام والمهام الموجودة داخل كل مرحلة. تشمل إدارة مشروع تطوير نظام المعلومات على مرحلة التخطيط، ومرحلة التحليل، ومرحلة التصميم، ومرحلة البرمجة، وأخيراً مرحلة الدعم. وبالإضافة إلى اتباع هذه المراحل فإن مطوري النظم المعتمدة على الكائنات

يتبعون أسلوب التكرار (Iterative approach) وذلك في كل من التحليل والتصميم والبرمجة. التكرار هنا يعني الانتهاء من بعض التحليل، ثم الانتهاء من بعض التصميم، ثم الانتهاء من بعض البرمجة، وهكذا.

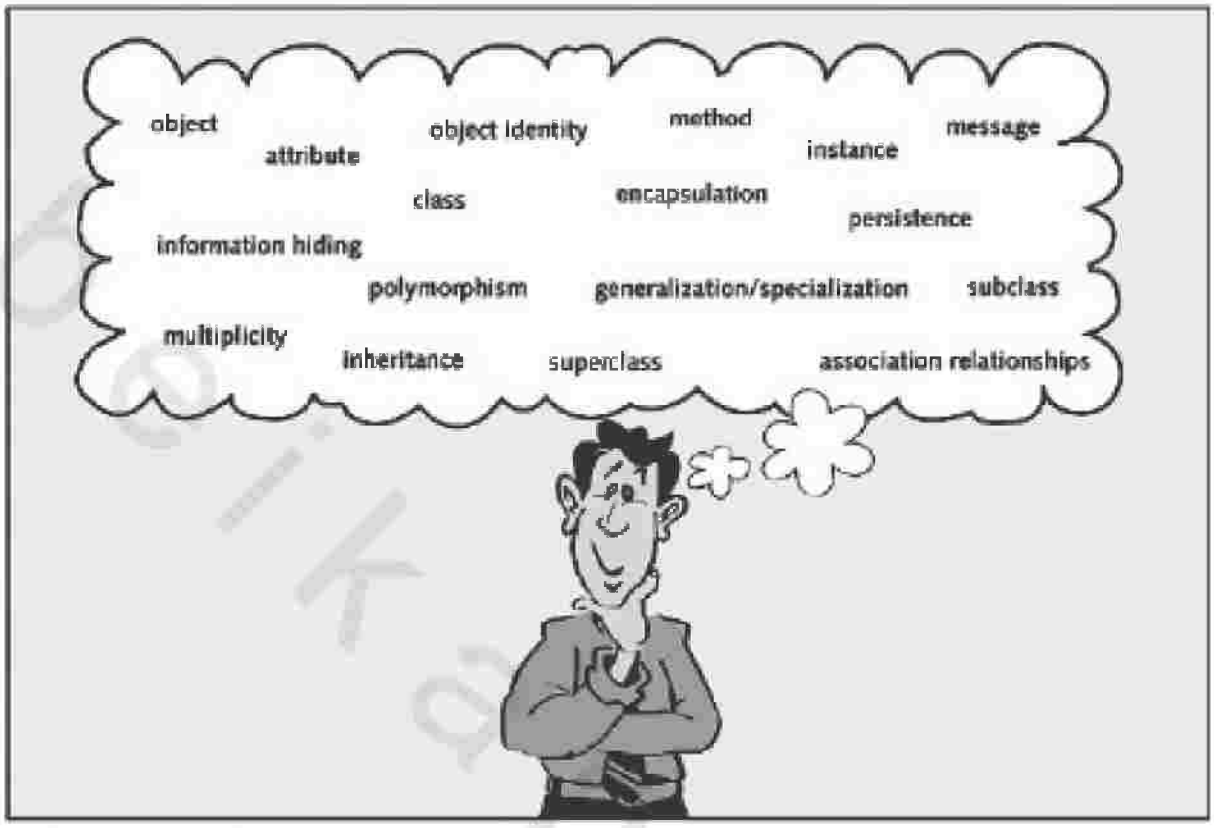
ربما تعتمد عملية تطوير التطبيقات المعتمدة على الكائنات على كل من تقنية "النمذجة الفعالة" (Prototyping) وتقنية "التطوير المشترك للنظم" (Joint Application Development). إن تقنية Prototyping تعني إنشاء برنامج مصغر كنموذج لواحد أو أكثر من أجزاء النظام لكي نعطي المستخدم فرصة التعديل والتقييم على شيء محسوس وملموس له. يعمل كل من مطوري النظام ومتخذي القرار سوياً أثناء جلسات تقنية "التطوير المشترك للنظم" لتحديد متطلبات النظام في فترة زمنية قصيرة. كما أن التطوير باستخدام تقنية OO يتطلب إدارة المشروع، وإجراء مقابلات، وجمع البيانات، وتصميم واجهة النظام، واختبار النظام كما هو متبع في الطريقة التقليدية.

التعرف على مفاهيم تقنية الكائنات الموجهة
Understanding Object-Oriented Concepts

كما ذكرنا سابقاً أن نظام المعلومات المعتمد على الكائنات ما هو إلا مجموعة من الكائنات تتفاعل بعضها مع بعض لإنجاز مهام النظام، ولكي تتمكن من تطوير مثل تلك النظم لابد أن تتعرف على المفاهيم الأساسية المتعلقة بتقنية الكائنات والموضحة في الشكل رقم (١.٥). سوف يتم تلخيص هذه المفاهيم خلال هذه الفقرة، ثم يتم شرحها بالتفصيل داخل صفحات هذا الكتاب.

الكائنات والصفات والإجراءات
Objects, Attributes, and Functions

يعرف الكائن (Object) في مجال الكمبيوتر كما يعرف في الحقيقة حيث أن الكائن هو أي شيء له صفات وله سلوك. يحتوي نظام المعلومات على العديد من الكائنات مثل كائنات واجهة المستخدم الرسومية (GUI objects) والكائنات المتعلقة بالمشكلة التي يحلها البرنامج والتي تمثل محور البرنامج مثل كائنات الصنف Student وكائنات الصنف Teacher في نظام معلومات تسجيل مواد طلاب داخل جامعة ما. فعلى سبيل المثال فإن كائنات واجهة البرنامج (مثل النماذج Forms والأزرار Buttons والعناوين Labels) لها عدة خصائص (صفات الكائن) مثل اللون والحجم والموقع التي تحدد مظهر الكائن وذلك بواسطة إسناد قيمة لكل من هذه الصفات. كما تحتوي هذه الكائنات أيضاً على إجراءات (Methods) وأحياناً يطلق عليها سلوك (Behavior) والتي تصف ما يمكن أن يفعله الكائن عند حدوث أحداث محددة.



الشكل رقم (١.٥). المفاهيم الأساسية لتقنية الكائنات.

فعلى سبيل المثال فإن الزر (كائن الصنف Button) يستجيب عندما يتم الضغط عليه بتنفيذ إجراء ما، والعنوان (كائن الصنف Label) يتغير مظهره ومحتواه عند تغيير الصفات المتعلقة بذلك، كما يمكن تغيير حجم النموذج (كائن الصنف Form) أو مظهره أو حتى إخفاءه ثم إظهاره عند تغيير الصفات المتعلقة بذلك. يوضح الشكل رقم (١.٦) أمثلة من كائنات واجهة المستخدم (GUI objects)، والصفات (Attributes)، والإجراءات (Methods) الخاصة بها.

من السهل على كل من المستخدم والمبرمج أن يفهما كائنات واجهة المستخدم (GUI objects) لأنهما يتفاعلان معها ويشاهدونها، ولكن تحتوي نظم المعلومات أيضاً على نوع آخر من الكائنات يطلق عليها اسم "كائنات مجال المشكلة" (Problem Domain Objects) والتي تتعلق بموضوع المشكلة، فعلى سبيل المثال فإن نظم معالجة طلبات العملاء تحتوي على كائنات من الصنف Order (تمثل طلبات العملاء)، وعلى كائنات من الصنف Product (تمثل المنتجات)، وبالمثل فإن كائنات مجال المشكلة تحتوي على صفات (Attributes) وإجراءات (Methods) كما هو واضح في الشكل رقم (١.٧).

كائنات واجهة الإستخدام	الصفات	الإجراءات
Button	size, shape, color, location, caption	click, enable, disable, hide, show
Label	size, shape, color, location, text	set text, get text, hide, show
Form	width, height, border style, background color	change size, minimize, maximize, appear, disappear

الشكل رقم (١,٦). صفات وإجراءات كائنات واجهة المستخدم.

كائنات واجهة الإستخدام	الصفات	الإجراءات
Customer	name, address, phone number	set name, set address, add new order for customer
Order	order number, date, amount	set order date, calculate order amount, add product to order, schedule order shipment
Product	product number, description, price	add to order, set description, get price

الشكل رقم (١,٧). صفات وإجراءات كائنات مجال المشكلة.

عزيزي القارئ لاحظ أن صفات كائنات مجال المشكلة تشبه حقول جداول قواعد البيانات، فعلى سبيل المثال فإن صفات العميل (كائن الصنف Customer) تشمل الاسم (Name) والعنوان (Address) ورقم الهاتف (Phone)، بالإضافة إلى أن هذه الكائنات تحتوي على إجراءات (Methods) تمكنها من إدارة المهام الخاصة بها، فمثلاً يستطيع العميل (كائن الصنف Customer) أن يسند قيمة لكل من الصفة Name والصفة Address بواسطة تنفيذ كل من الإجراء ( )SetName، والإجراء ( )SetAddress، كما يستطيع أن يضيف طلباً جديداً (كائن من الصنف Order) له بواسطة تنفيذ الإجراء ( )AddNewOrder. أما إجراءات كائن الصنف Order فهي مسؤولة عن إسناد قيمة تاريخ الطلب بواسطة تنفيذ الإجراء ( )SetOrderDate، وحساب القيمة الكلية للطلب بواسطة تنفيذ الإجراء ( )CalculateOrderAmount.

تفاعل الكائنات وإرسال الرسائل
Object Interactions and Messages

تتفاعل الكائنات بعضها مع بعض بإرسال الرسائل (Messages) حيث يستجيب الكائن المستقبل للرسالة بتنفيذ أحد الإجراءات (Method) المناظرة لهذه الرسالة. ولتوضيح ذلك نفترض وجود كائن من الصنف Customer اسمه Bill استقبل رسالة لإضافة كائن من الصنف Order (الطلب Order143) فيستجيب الكائن Bill بإضافة هذا الطلب إلى نفسه، ثم يطلب الكائن Bill من الكائن Order143 أن يحسب القيمة الكلية للطلب بواسطة إرسال رسالة له، وهكذا.

عزيزي القارئ لابد أن تعي جيداً أن فهم تقنية OO والوصول إلى تحليل وتصميم جيد يتطلب فهم العبارة التالية :

"إن مهام النظام تنجز بواسطة تفاعل الكائنات سوياً وذلك بإرسال رسائل ومن ثم تنفيذ الإجراءات المناظرة"

يوضح الشكل رقم (١.٨) نظام معالجة طلبات العملاء (يحتوي على الكائن Bill) كمجموعة من الكائنات المتفاعلة. كما يتفاعل المستخدم مع النظام من خلال كائنات واجهة الاستخدام (GUI objects) التي بدورها تتفاعل مع كائنات مجال المشكلة (Problem Domain Objects) بواسطة إرسال الرسائل (Messages)، ويمكن تلخيص عمل هذا النظام كما يلي :

١) يقوم المستخدم بكتابة بيانات المنتج داخل صندوق نصي (كائن الصنف Textbox)، ثم يضغط على الزر (كائن الصنف Button) مما يعني إرسال رسالة لهذا الزر من المستخدم.

٢) يعلم الزر أنه يجب أن يطلب من الكائن Bill إضافة طلب جديد لنفسه عندما يتم الضغط عليه، ولذلك سيرسل الكائن Button رسالة إلى الكائن Bill بهذا الخصوص.

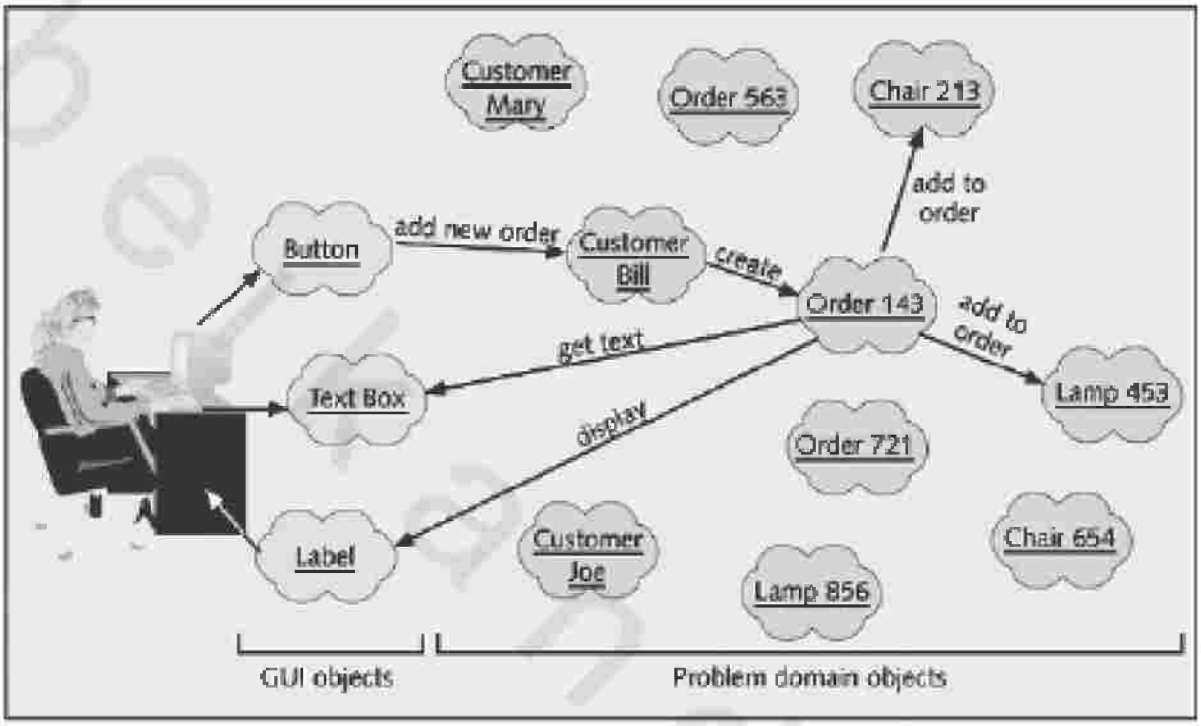
٣) يعلم الكائن Bill جيداً كيف يضيف طلباً جديداً لنفسه لأنه يحتوي على إجراء خاص بذلك، ولتنفيذ ذلك سيرسل الكائن Bill رسالة بغرض إنشاء كائن من النوع Order.

٤) يسند الكائن Order الجديد رقم طلب لنفسه (Order143)، ثم يطلب رقم الطلب من صندوق النص الذي يحتويه، وعندئذ يستخدم الكائن Order143 هذا النص لكي يعرف أي المنتجات المطلوبة بواسطة المستخدم.

٥) يرسل الكائن Order143 رسالة add-to-order لكل منتج موجود في النص طالباً منه إضافة نفسه في الطلب مع العلم أن المنتجات المطلوبة في هذا المثال هي المنتج "Chair 213" والمنتج "Lamp 453".

٦) يضيف كلاً من المنتج "Chair 213 " والمنتج "Lamp 453" بياناتهما (السعر والكمية المتاحة) إلى الطلب.

٧) أخيراً ينهي الكائن Order143 المهمة بواسطة حساب القيمة الكلية للطلب وإرسال رسالة للعنوان (كائن الصنف Label) طالباً منه إظهار معلومات الطلب للمستخدم.



الشكل رقم (١,٨). تفاعل كائنات نظام معالجة الطلبات بواسطة إرسال الرسائل.

تغليف عناصر الكائنات وإخفاء المعلومات
Encapsulation and Information Hiding

كما ذكرنا سابقاً أن الكائنات تحتوي على صفات (Attributes) التي تمثل بيانات الكائن وإجراءات (Methods) التي تمثل سلوك الكائن عند استقباله رسائل (Messages) من كائنات أخرى. إن احتواء الكائن على كل من البيانات والإجراءات في كيان واحد متكامل يعني تغليفها داخل وحدة متكاملة ومستقلة. يعتبر هذا المفهوم أهم مفاهيم تقنية OO، والذي يسمى مفهوم Encapsulation وهو الذي يجعل المبرمج يستفيد من إمكانات كائن ما (إرسال رسائل له لتنفيذ الإجراءات المناظرة) دون أن يعرف تركيبه الداخلي حيث يجب أن يعرف ما يمكن أن يقدمه له هذا الكائن من سلوك (Methods) فقط، ولأن مفهوم Encapsulation يخفي التركيب الداخلي للكائن فهو يحمي بياناتها من التغيير المباشر بقيم غير صحيحة من قبل الكائنات الأخرى وهو ما نطلق عليه مفهوم "إخفاء المعلومات" (Information Hiding).

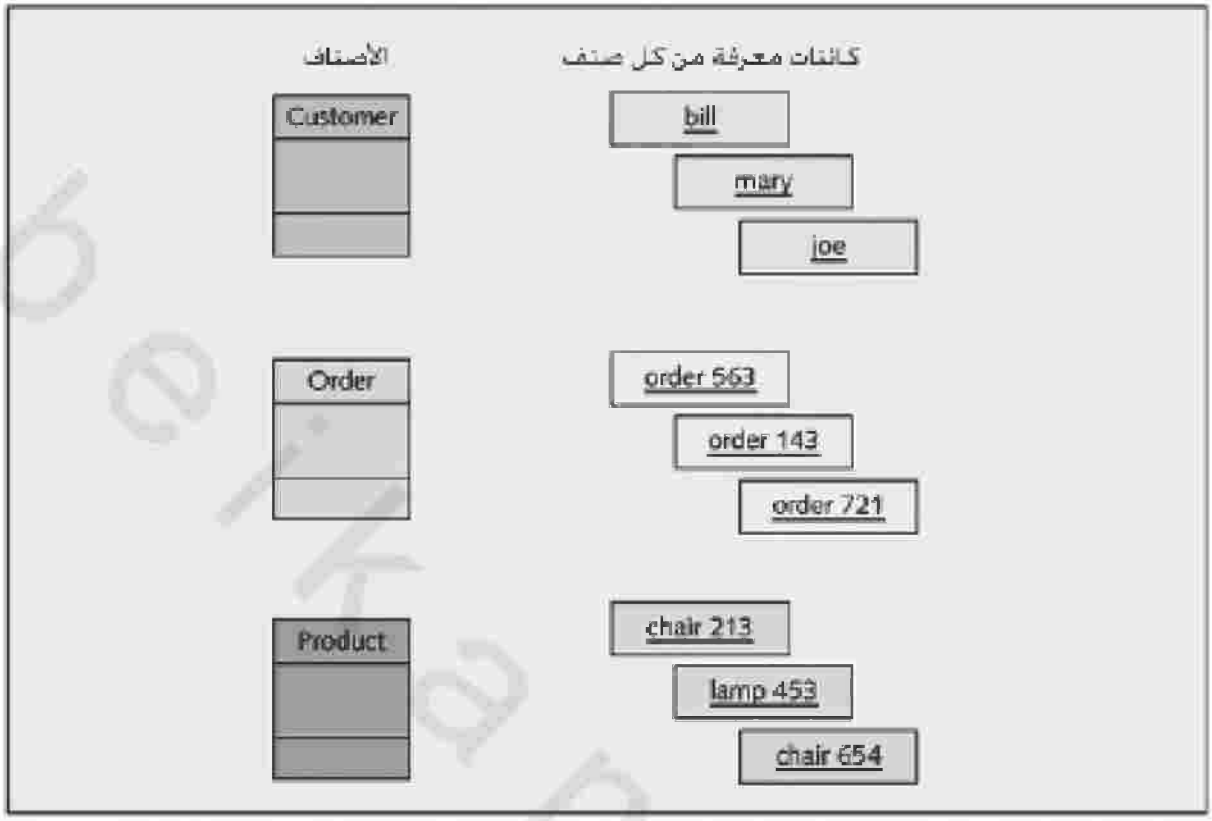
يجب تعريف هوية (Identity) لكل كائن (Object) لتعريفه عن غيره، والتي من خلالها يمكن إيجاده وإرسال الرسائل له. لذا يجب أن تعرف هوية الكائن قبل أن ترسل له رسالة، وعادة ما تخزن هوية الكائنات كعناوين في الذاكرة. يتم استخدام كائن ما لفترة زمنية أثناء تنفيذ أحد مهام النظام لحين الانتهاء من تنفيذ هذه المهمة، ثم حذف هذا الكائن من الذاكرة، وللتغلب على هذه المشكلة لابد من الإبقاء على هذه الكائنات في قاعدة بيانات أو ملفات نصية، وفي هذه الحالة نطلق عليها كائنات دائمة (Persistent Objects)، ولذلك إذا عرف النظام مئات العملاء (كائنات من الصنف Customer)، وتم تعريف طلبات لهم، فيمكن تخزين هذه الكائنات داخل قاعدة بيانات لتصبح كائنات دائمة، ومن ثم استرجاع بياناتهم في أي وقت لاحق.

الأصناف والكائنات وعلاقة الترابط
Classes, Instances, and Associations

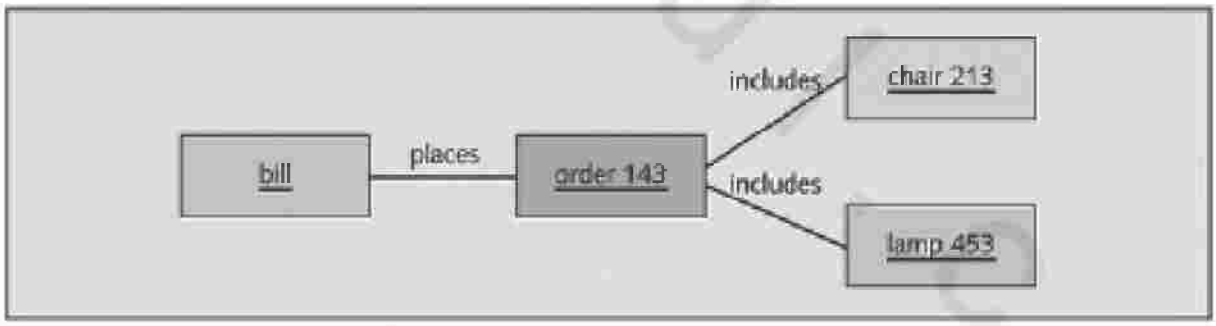
كما هو واضح من الشكل رقم (١,٩)، يوجد العديد من العملاء في نظام معالجة طلبات العملاء والتي تنتمي إلى صنف واحد (الصنف Customer)، وعندما نتحدث عن الصنف Customer فكأننا نتحدث عن جميع الكائنات المنتمية إلى هذا الصنف. يوجد فرق كبير بين الصنف (Class) وبين الكائن (Object)، فإن الصنف (Class) يعرف عناصر الكائنات من صفات وإجراءات، أما الكائن فهو نسخة (Instance) من الصنف ولذلك فإن كلاً من كلمة Object وكلمة Instance تشير لنفس المعنى، وعندما نقوم بتعريف كائن من صنف ما، فإن هذه العملية تسمى Instantiation أي إنشاء نسخة من صنف.

لكي تتمكن الكائنات من إرسال رسائل إلى بعضها بعضاً فلابد أن تحتفظ هذه الكائنات بعلاقات ترابط فيما بينها (Association relationships)، فعلى سبيل المثال فإن العميل (كائن الصنف Customer) يجب أن تربطه علاقة بالطلبات (كائنات الصنف Order) التي طلبها لكي يتمكن من الاستفسار عن طلب ما، وإضافة طلب جديد، وهكذا، كما يجب أن يرتبط الطلب بالمنتجات التي يشملها. وكما هو واضح من الشكل رقم (١,١٠)، تسمى العلاقة التي تربط العميل بالطلبات "Places" (وضع طلب)، وتسمى العلاقة التي تربط الطلب بالمنتجات "Includes" (تشمل). تشبه العلاقات بين الكائنات في تقنية الكائنات العلاقات بين الجداول في قواعد البيانات.

ربما تكون بعض هذه العلاقات من النوع واحد إلى واحد (one-to-one) مثل علاقة كائن الصنف Order بكائن الصنف Customer (أي أن الطلب يتم طلبه بواسطة عميل واحد فقط والعميل يطلب طلباً واحداً)، وأحياناً تكون العلاقة واحداً إلى العديد (one-to-many) مثل علاقة الترابط بين كائن الصنف Order بكائنات الصنف Product، أي أن الطلب ربما يشمل العديد من المنتجات. تشير لغة UML إلى درجة هذه العلاقات بكلمة Multiplicity على العكس في مجال قواعد البيانات يشار إليها بكلمة Cardinality.



الشكل رقم (١,٩). أصناف مجال المشكلة وأمثلة للكائنات المعرفة.



الشكل رقم (١,١٠). ربط الكائنات بواسطة علاقات الترابط.

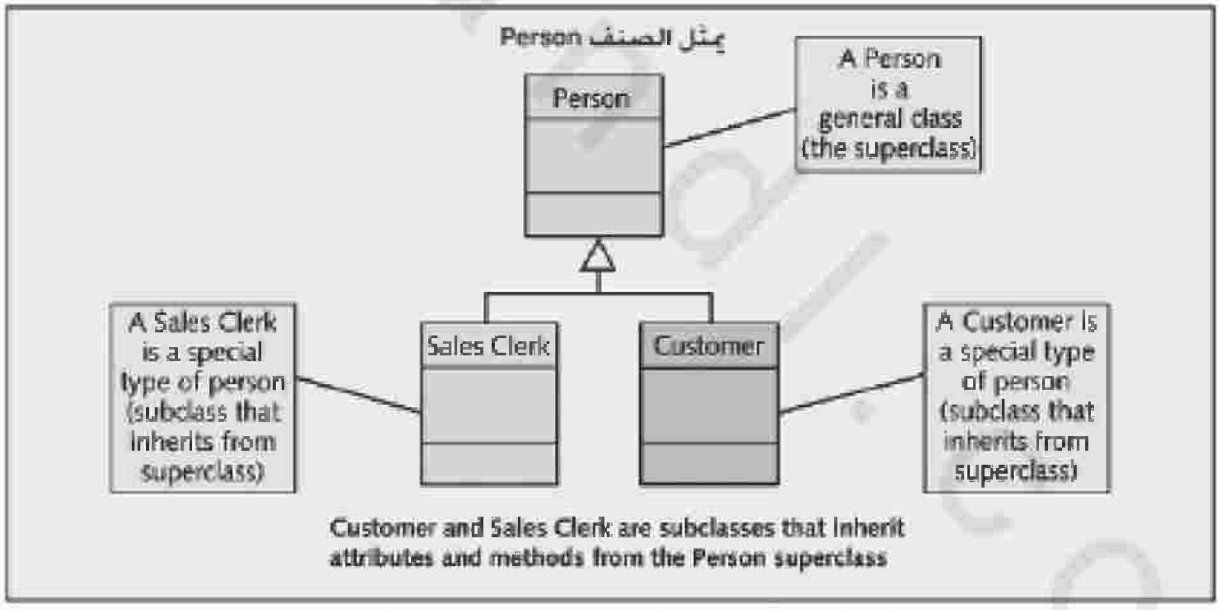
التوارث وتعدد الأشكال

Inheritance and Polymorphism

ربما يكون أكثر المفاهيم أهمية عند مناقشة تقنية الكائنات (Object-Oriented) هو مفهوم التوارث (Inheritance) والذي يجعل صنفاً ما يرث خصائص صنف آخر وإجراءاته، ثم يزيد عليهما بعناصر أخرى. فعلى سبيل المثال، إن الصنف Customer ربما يكون امتداداً لصنف آخر أكثر عمومية اسمه Person، فلو

افترضنا أن الصنف Person قد تم تعريفه سابقاً، فإن الصنف Customer يمكن تعريفه كأحد ورثة الصنف Person لكي يرث جميع عناصره من صفات (Attributes) وإجراءات (Methods)، ثم يزيد عليه ما يريد من صفات وإجراءات خاصة به.

فمثلاً ربما يكون الصنف Person يحتوي على كل من الصفة Name والصفة Address، وبما أن الصنف Customer هو نوع خاص من الصنف Person فإنه يختص ببعض الصفات الأخرى مثل الصفة ShippingAddress (عنوان الشحن) والصفة CreditCardInformation (معلومات بطاقة الائتمان)، وبنفس الطريقة فإن الصنف SalesClerk (موظف المبيعات) هو امتداد للصنف Person حيث أن الصنف SalesClerk يحتوي على صفات إضافية مثل الصفة JobTitle (المسمى الوظيفي)، والصفة PayRate (معدل الدفع). تطلق على الصنف Person اسم "super class" أي الصنف الأساسي، وتطلق على كل من الصنف Customer والصنف SalesClerk اسم "subclasses " أي الأصناف الفرعية. يوضح الشكل رقم (١,١١) علاقة التوارث بين كل من الصنف Person وكل من الصنف Customer، والصنف SalesClerk.



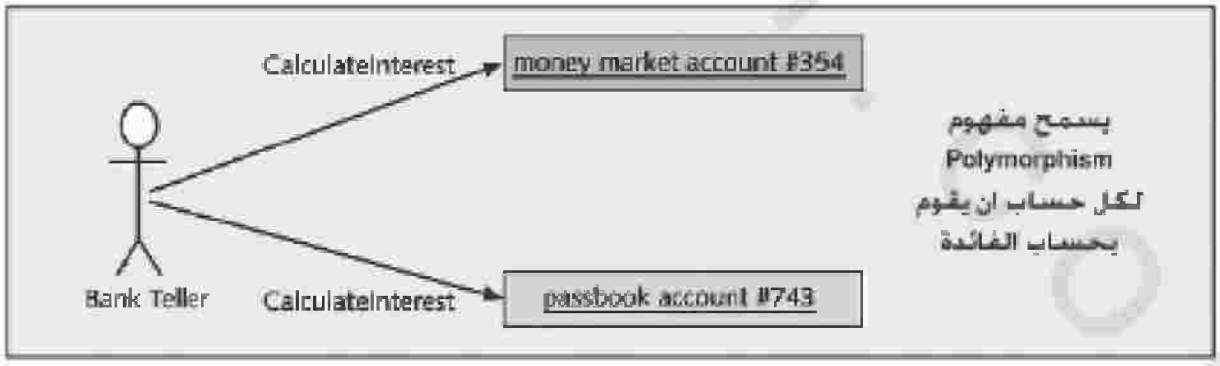
الشكل رقم (١,١١). الأصناف الأساسية والأصناف الفرعية.

تصنيف كائنات النظام إلى أصناف يساعد في تحديد الأصناف الفرعية المنبثقة من أصناف أساسية والتي تعتبر نوعاً خاصاً منها، وما ينتج عن هذا التقسيم يسمى "Generalization/Specialization Hierarchy" (شجرة التعميم/التخصيص)، وأحياناً أخرى يسمى "Inheritance Hierarchy" (شجرة التوارث)، وعندما يرث صنف ما

صنفاً آخر فإنه يرث جميع خصائص هذا الصنف من صفات (Attributes) وإجراءات (Methods) وعلاقات (Associations).

يوجد مفهوم آخر مرتبط بمفهوم التوارث وهو مفهوم Polymorphism ويعني تعدد الأشكال، حيث يشير هذا المفهوم إلى أن الكائنات المختلفة (من أصناف مختلفة) تستجيب بسلوك مختلف عند استقبالها نفس الرسالة، فعلى سبيل المثال، إن الكائن DialogBox (نافذة)، والكائن NetworkConnection (اتصال شبكة)، والكائن Document (وثيقة) تستجيب بأساليب مختلفة للرسالة ( )Close حيث يعلم كل من هذه الكائنات كيفية الاستجابة لهذه الرسالة بطريقته الخاصة، مع العلم أن المرسل لا يحتاج أن يعرف ما نوع الكائن المستقبل للرسالة ولا كيفية استجابته لها، والذي يريد أن يعرفه فقط هو هل الكائن المستقبل سيستجيب لهذه الرسالة أم لا (هل هذا الكائن يحتوي على إجراء يناظر هذه الرسالة أم لا).

عزيزي القارئ لكي يتضح مفهوم Polymorphism تعال معي لنعرض مثالاً آخر. افترض أن هناك بنكاً ما يحتوي على أنواع عديدة من الحسابات (Accounts) مثل حساب الوديعة وحساب الدفتر الجاري وكل منها يحسب فوائد الحساب بأسلوب مختلف طبقاً لقواعد الفائدة في هذا الحساب. فإذا أراد كائن آخر أن يعرف فوائد حساب كل منها، فيجب عليه أن يرسل لهم الرسالة ( )CalculateInterest، وفي هذه الحالة فإنه لا يحتاج أن يعرف ما هو نوع الحساب ولا قواعد حساب الفائدة، وكل ما يريد أن يعرفه هو هل هذا الكائن يستجيب لهذه الرسالة أم لا كما هو واضح في الشكل رقم (١.١٢). وكما هو ملاحظ أن المعالجة بهذا المفهوم أصبحت أسهل حيث تستطيع جميع حسابات البنك أن تستقبل نفس الرسالة وتؤدي الوظيفة بشكل صحيح وكامل.



الشكل رقم (١.١٢). تطبيق مفهوم تعدد الأشكال لنوعين من حسابات البنك.

عزيزي القارئ لا تقلق من عدم استيعاب تقنية الكائنات بشكل جيد، فإنه من المتوقع مع نهاية هذا الكتاب وتكملة حلول أسئلة لغة فيجوال بيسك .نت أنه سيكون لديك خبرة ممتازة بخبايا هذه التقنية.

فوائد التطوير بتقنية الكائنات

Recognizing the Benefits of OO Development

كما ذكرنا سابقاً أن تقنية الكائنات ظهرت في البداية من أجل برامج المحاكاة، ثم استخدمت بكثرة لعمل واجهات استخدام رسومية (GUI)، ولكن ما سبب استخدامها في تطوير نظم المعلومات. يوجد هناك سببان رئيسان لاستخدام تقنية الكائنات في تطوير نظم المعلومات. السبب الأول هو إعادة الاستخدام (Reusability)، والسبب الثاني هو توافق طبيعة التقنية مع الواقع (Naturalness).

تقنية الكائنات أكثر توافقاً مع الواقع

Objects Are More Natural

تتوافق طبيعة الكائنات مع الواقع بشكل كبير حيث يتعامل الإنسان مع عالمه الخارجي على أنه مجموعة من الكائنات، وبذلك فإنه من السهل على الناس أن تتحدث عن أعمالهم وتناقش متطلبات نظم المعلومات مستخدمين لغة الكائنات. كما أن مراحل التطوير المختلفة (التحليل، والتصميم، والبرمجة) ترتكز على استخدام الكائنات، ولذلك سيظل مفهوم الكائنات هو محور التركيز أثناء مرحلة التطوير.

يتجنب بعض المبرمجين ذي الخبرة العالية استخدام تقنية الكائنات معتقدين أنها ليست طبيعية كالبرمجة الإجرائية، ويرجع هذا في الحقيقة إلى أن هؤلاء المبرمجين يجدون صعوبة في تعلم تقنية الكائنات وذلك لأن تقنية البرمجة الإجرائية تمثل الخبرة الأولى بالنسبة لهم والتي يصعب تغييرها.

على العكس فإن المبرمجين الجدد يجدون صعوبة في تعلم البرمجة الإجرائية ويرون أن تقنية الكائنات أكثر طبيعة من البرمجة الإجرائية وذلك في تطوير نظم المعلومات. ويجب التنويه في هذا السياق على أن قلة من المبرمجين يستطيعون كتابة شفرة معقدة مستخدمين البرمجة الإجرائية.

بالإضافة إلى ذلك فالمستخدم النهائي يتعامل مع كائنات النظام بشكل تلقائي وطبيعي، ولذلك يستطيع المستخدمون أن يناقشوا كائنات النظام بسهولة ويسر. تمثل مفاهيم الكائنات (الأصناف والكائنات) الطريقة الطبيعية للناس التي تمكنهم من تنظيم معرفتهم.

إعادة استخدام أصناف الكائنات

Classes of Objects Can Be Reused

بالإضافة إلى توافق تقنية الكائنات مع الواقع فإن هذه التقنية توفر إمكانية إعادة استخدام الأصناف والكائنات في العديد من التطبيقات، وإعادة الاستخدام تعني تطوير أصناف الكائنات مرة واحدة، ثم استخدامها داخل العديد من التطبيقات. فعلى سبيل المثال، إذا قام المبرمج بتطوير الصنف Customer في نظام ما، فيمكن إعادة

استخدام هذا الصنف في العديد من التطبيقات التي تحتوي على عملاء (كائنات من الصنف Customer)، وإذا كان التطبيق الجديد يحتوي على عملاء من نوع معين، فيمكن تعريف صنف جديد يرث خصائص الصنف Customer، ثم يزيد عليها عناصر خاصة به. يمكن إعادة استخدام الأصناف بهذا الأسلوب في كل من مرحلة التحليل، والتصميم، والبرمجة.

لا يحتاج المبرمج أثناء مرحلة البرمجة أن يطلع على شفرة صنف معين حتى ولو كان هذا المبرمج سيقوم بتطوير صنف آخر يرث هذا الصنف، وهذا المفهوم يسهل عملية التطوير ويبسطها. تأتي لغات البرمجة المعتمدة على الكائنات بمكتبة تحتوي على عدد هائل من الأصناف سابقة التعريف والتي يستخدمها معظم المبرمجين لتطوير برامجهم، وتعتبر مكتبة إطار عمل .نت مثالاً على أحد هذه المكتبات. فأصناف واجهة الاستخدام الرسومية (GUI) مثل الصنف Button، والصنف Label، والصنف TextBox، والصنف CheckBox تأتي مع مكتبة إطار عمل .نت، ولذلك لا يحتاج المبرمج لتعريفها من البداية، ولكن يستخدمها فقط عن طريق إنشاء كائنات من هذه الأصناف لتطوير واجهة استخدام البرنامج. تحتوي أيضاً مكتبة إطار عمل .نت على أصناف هامة مثل تلك التي تستخدم للاتصال بقاعدة البيانات والاتصال بشبكة ما.

تعليم تطوير نظم المعلومات باستخدام تقنية الكائنات

Learning OO Development

يعتبر هذا الكتاب مرشداً شاملاً لتطوير نظم المعلومات مستخدماً تقنية الكائنات حيث يشمل كلاً من مرحلة التحليل (OOA)، ومرحلة التصميم (OOD)، ومرحلة البرمجة (OOP). سوف تُستخدم لغة VB .NET كأحد لغات البرمجة المعتمدة على الكائنات لتطبيق مفاهيم هذه التقنية، مع العلم أن مفاهيم هذه التقنية نفسها يمكن تطبيقها مستخدماً أي لغة برمجة تعتمد على الكائنات. نقدم خلال هذه الفقرة خريطة تنظيم الكتاب.

التصميم بأسلوب ثلاثي الطبقات

Introducing Three-Tier Design

تم تقسيم الكتاب طبقاً لتقنية تسمى Three-Tier Approach والتي تعني تقسيم أصناف النظام وكائناته إلى ثلاثة طبقات، وهذا يتطلب تصنيفها إلى ثلاثة طبقات من الأصناف وهي أصناف مجال المشكلة (Problem Domain Classes)، وأصناف واجهة المستخدم الرسومية (GUI Classes)، وأصناف التعامل مع البيانات (Data Access Classes). كما ذكرنا سابقاً فإن أصناف مجال المشكلة تعني الأصناف المتعلقة بالمشكلة التي يحاول النظام حلها مثل الصنف Customer، والصنف Order، والصنف Product. أما أصناف واجهة المستخدم فتمثل الأصناف المسؤولة عن تكوين واجهة البرنامج مثل الصنف Form (نموذج)، والصنف Button (زر)، والصنف TextBox (صندوق النص)، وأخيراً أصناف التعامل مع

البيانات فهي المسؤولة عن التعامل مع برامج إدارة قواعد البيانات لتخزين معلومات كائنات مجال المشكلة بغرض استخدامها لاحقاً، وهذا ما يجعل كائنات مجال المشكلة كائنات دائمة (Persistent Objects).

وكما ذكرنا فإن هذا الأسلوب يتطلب من المبرمج أن يقوم بتعريف ثلاثة طبقات من الأصناف عندما يصمم نظم المعلومات ويطورها. أولاً يحدد المطور أصناف مجال المشكلة، ثم يحدد أصناف واجهة البرنامج التي تمكن المستخدم النهائي من التفاعل مع كائنات أصناف مجال المشكلة، وأخيراً يحدد أصناف التعامل مع البيانات التي تسمح بتخزين كائنات أصناف مجال المشكلة واسترجاعها من قواعد البيانات.

تم تقسيم هذا الكتاب طبقاً لتقنية Three-Tier Approach حيث يقدم الباب الأول من هذا الكتاب المفاهيم الأساسية لكل من تقنية الكائنات ولغة VB .NET، أما الأبواب الثاني إلى الخامس فتتبع عملية التطوير بناء على هذه التقنية، ولذلك يشرح الباب الثاني أصناف مجال المشكلة (Problem Domain Classes) حيث أنها تمثل محور مرحلة التحليل (OOA).

يحتوي الباب الثالث على شرح وافٍ لأصناف واجهة المستخدم الرسومية (GUI Classes)، ويحتوي الباب الرابع على شرح وافٍ لأصناف التعامل مع البيانات (Data Access Classes)، ثم نقدم شرحاً وافياً لعملية دمج أنواع الأصناف الثلاثة من خلال تطوير برنامج متكامل يعمل تحت بيئة الويندوز وبرنامج آخر يعمل تحت بيئة الإنترنت.

الباب الأول: أساسيات كل من تقنية الكائنات ولغة VB .NET

Part 1: Object-Orientation and VB .NET Fundamentals

يقدم الباب الأول كلاً من مفاهيم تقنية الكائنات وأساسيات لغة VB .NET حيث قدم الفصل الأول خلفية عامة لمفاهيم هذه التقنية، ثم عرض مميزات تطوير النظم المعتمدة على تقنية الكائنات. أما الفصل الثاني والثالث والرابع فسوف تركز على تعليم لغة VB .NET حيث يشرح الفصل الثاني البيئة المتكاملة للتطوير إصدار فيجوال ستوديو .نت (Visual Studio IDE)، ثم يشرح الفصل الثالث كيفية كتابة أوامر لغة VB .NET وقواعدها (المتغيرات، وأوامر اتخاذ القرار، وأوامر التكرار). فإذا كان لديك عزيزي القارئ فكرة عن هذه الأمور مستخدماً لغة فيجوال بيسك، فيمكنك أن تأخذ فكرة سريعة عن هذا الفصل. أما إذا لم يكن لديك خبرة بلغة فيجوال بيسك، فإن هذا الفصل سيشرح لك بالتفصيل هذه المواضيع.

يقدم الفصل الرابع أصناف مكتبة إطار عمل .نت بلغة VB .NET حيث تعرض كلاً من مفاهيم تقنية الكائنات الموجهة من خلال لغة VB .NET وأهمية تفاعل كائنات النظام لإنجاز مهامه، ونعرض أيضاً فائدة إعادة استخدام أصناف مكتبة إطار عمل .نت (تطبيق مفهوم التوارث Inheritance)، وأخيراً نعرض مفهوم تعريف كائنات من أصناف (Instantiation).

يقدم الفصل الخامس مفاهيم كل من التحليل (OOA)، والتصميم (OOD) مستخدماً تقنية OO، كما يقدم الفصل مقدمة عن نظام المعلومات المستخدم خلال جميع أبواب الكتاب (نظام معلومات لشركة تأجير مراسي ومراكب Bradshaw Marina)، وسوف يقدم هذا الفصل معظم نماذج مرحلة التحليل والتصميم بلغة UML مشتملاً على كل من نموذج Use case diagram، ونموذج Class diagram.

الباب الثاني: تطوير أصناف مجال المشكلة

Part 2: Developing Problem Domain Classes

نقوم في هذا الباب بتوضيح كيفية تطوير الأصناف المتعلقة بالمشكلة المطلوب حلها بواسطة تطوير نظام معلوماتي، فمثلاً يحتوي نظام معلومات شركة "برادشو مارينا" (المسؤول عن إدارة بيانات العملاء والمراكب والعقود) على الصنف Customer (العملاء)، والصنف Boat (المراكب)، والصنف Slip (المراسي) كأصناف متعلقة بمجال المشكلة. يوضح الفصل السادس كيفية تطوير أحد أصناف مجال المشكلة من خلال تعريف عدد قليل من الصفات (Attributes) والإجراءات (Methods)، ثم نقوم بإنشاء برنامج اختبار لتعريف كائن من هذا الصنف واستدعاء أحد إجراءاته (إرسال رسالة له). يستمر الفصل السابع في تطوير أصناف مجال المشكلة وذلك بشرح موضوع التحقق من صحة البيانات وتقديم أسلوب معالجة الأخطاء (Exception Handling)، ومناقشة مواضيع أخرى.

يقدم الفصل الثامن مفهوم التوارث (Inheritance) بين أصناف مجال مشكلة نظام شركة "برادشو مارينا" حيث يرث كلاً من الصنف الفرعي SailBoat (مركب شراعي) والصنف الفرعي PowerBoat (مركب آلي) الصنف الأساسي Boat (مركب). وأخيراً يوضح الفصل التاسع كيفية إنشاء علاقات الترابط (Association Relations) بين الكائنات حيث يوجد هناك نوعان من العلاقات، أولاً: علاقة ترابط من الرتبة "واحد–إلى–واحد" مثل علاقة الترابط بين العميل والمركب، وثانياً: علاقة ترابط من الرتبة "واحد–إلى–عديد" مثل علاقة الترابط بين المرسى والرصيف.

الباب الثالث: تطوير أصناف واجهة الاستخدام

Part 3: Developing GUI Classes

يوضح الباب الثالث كيفية تطوير أصناف واجهة الاستخدام (GUI Classes) التي يتعامل معها المستخدم النهائي، والتي بدورها تتفاعل مع أصناف مجال المشكلة (Problem Domain Classes). يقدم الفصل العاشر جميع المكونات المتاحة لتصميم النماذج وتطويرها (Forms)، ثم يقدم الفصل الحادي عشر النماذج متعددة النوافذ التي تُمكّن المستخدم النهائي من التعامل مع نظام معلومات شركة "برادشو مارينا" بشكل أكثر مرونة. وأخيراً يوضح الفصل الثاني عشر كيفية تصميم نماذج الويب وتطويرها ومن ثم تطوير موقع لشركة "برادشو مارينا" باستخدام كلٍ من لغة ASP .NET ولغة HTML.

الباب الرابع: تطوير أصناف التعامل مع البيانات
Part 4: Developing Data Access Classes

يغطي الباب الرابع الطبقة الثالثة من أصناف التطبيق (طبقا لتقنية Three-Tier Approach) والتي تمثل أصناف التعامل مع البيانات، والتي تستخدم لإدارة التعامل مع قواعد البيانات لجعل كائنات أصناف مجال المشكلة كائنات دائمة (أي يمكن استرجاعها والتعامل معها في أي وقت). يوضح الفصل الثالث عشر أمثلة عن استخدام الملفات (Files)، وقواعد البيانات العلاقية (Relational Database)، والنشر المتسلسل للكائنات (Object Serialization) لجعل الكائنات دائمة. إن معظم مطوري البرامج الآن يستخدمون قواعد البيانات العلاقية لإنجاز هذا الغرض، ولذلك سوف يُخصص الفصل الرابع عشر لشرح كل من إدارة قواعد البيانات العلاقية ولغة SQL المشهورة (Structured Query Language) بالتفصيل.

الباب الخامس: تطوير التطبيقات المعتمدة على تقنية ثلاثي الطبقات
Part 5: Developing the Three-Tier Application

سوف يوضح الباب الخامس كيفية توظيف كل من أصناف مجال المشكلة، وأصناف واجهة الاستخدام، وأصناف التعامل مع البيانات معاً لتطوير نظم العميل/الخادم (client/server systems). سوف يوضح الفصل الخامس عشر كيفية تحقق عملية تطوير نظم المعلومات المعتمدة على الكائنات من خلال مثال شركة "برادشو مارينا"، وأخيراً سوف يعرض الفصل السادس عشر تكنولوجيا مواقع الإنترنت ويناقش كيفية نشر موقع شركة "برادشو مارينا" على الإنترنت مستخدماً نماذج الويب، ولغة ASP .NET، ولغة XML.

عزيزي القارئ سوف تتعلم من خلال قراءتك لهذا الكتاب كيفية تحليل نظم المعلومات وتصميمها مستخدماً تقنية الثلاث طبقات (Three-Tier) التي تسهل كلاً من عملية التحليل (OOA)، وعملية التصميم (OOD)، وأخيراً كيفية تطوير كل من تطبيقات الويندوز والإنترنت مستخدماً لغة VB .NET.

ملخص الفصل
Chapter Summary

- تشمل عملية تطوير نظم المعلومات المعتمدة على الكائنات (Object-Oriented) على كل من مرحلة التحليل المعتمد على الكائنات (Object-Oriented Analysis)، ومرحلة التصميم المعتمد على الكائنات (Object-Oriented Design)، ومرحلة البرمجة المعتمدة على الكائنات (Object-Oriented Programming).
- يمكن رؤية أنظمة المعلومات المعتمدة على الكائنات على أنها مجموعة من الكائنات المتفاعلة فيما بينها لإنجاز مهام النظام حيث يرجع استخدام تقنية الكائنات الموجهة إلى عقود مضت حيث بدأ هذا الاتجاه في حقبة

الستينيات بتطوير لغة برمجة سيمولا (Simula)، ثم تلا ذلك لغات حديثة مثل لغة C# ولغة Java ولغة Visual Basic .NET. يمثل إطار عمل .نت (.NET Framework) منصة تطوير جديدة ليسهل عملية تطوير البرامج وخصوصاً في البيئة ذات الطبيعة الموزعة (Distributed Environment) مثل شبكة الإنترنت.

- تشمل عملية تطوير نظم المعلومات المعتمدة على الكائنات (Object-Oriented) على أكثر من مرحلة البرمجة حيث تشمل على تقنية نمذجة متطلبات النظام وعلى تصاميم النظام مستخدماً لغة UML. كما يتم استخدام مبادئ وتقنيات أخرى داخل عملية التطوير.
- تشمل تقنية الكائنات الموجهة على العديد من المفاهيم الأساسية وهي: الكائنات (Objects)، والأصناف (Classes)، والرسائل (Messages)، وتغليف العناصر (Encapsulation)، وإخفاء المعلومات (Information Hiding)، والتوارث (Inheritance)، وتعدد الأشكال (Polymorphism)، وعلاقة الترابط (Association Relation).
- تعد أهم فوائد استخدام تقنية الكائنات الموجهة في تطوير نظم المعلومات هو إعادة الاستخدام (Reusability) وتوافق طبيعة التقنية مع الواقع (Naturalness).
- تم تقسيم الكتاب إلى خمسة أبواب وطبقاً لتقنية تسمى Three-Tier Approach والتي تعني تقسيم أصناف النظام وكائناتها إلى ثلاثة طبقات وهي: أصناف مجال المشكلة (Problem Domain Classes)، وأصناف واجهة المستخدم الرسومية (GUI Classes)، وأصناف التعامل مع البيانات (Data Access Classes).
- يقدم الباب الأول من هذا الكتاب المفاهيم الأساسية لكل من تقنية الكائنات الموجهة ولغة VB .NET، أما الأبواب من الثاني إلى الخامس فتتبع عملية التطوير بناء على هذه التقنية، ولذلك يشرح الباب الثاني أصناف مجال المشكلة (Problem Domain Classes) حيث أنها تمثل محور مرحلة التحليل (OOA). يحتوي الباب الثالث على شرح وافي لأصناف واجهة المستخدم الرسومية (GUI Classes)، ويحتوي الباب الرابع على شرح وافي عن أصناف التعامل مع البيانات (Data Access Classes)، ثم يقدم الباب الخامس شرحاً وافياً لعملية دمج أنواع الأصناف الثلاثة من خلال تطوير برنامج متكامل يعمل تحت بيئة الويندوز وبرنامج آخر يعمل تحت بيئة الإنترنت.

المصطلحات الأساسية

Key Terms

مكون تنفيذ اللغات المشترك (Common Language Runtime)	تقنية الاعتماد على النماذج (Model-driven Approach)
علاقة ترابط (Association Relation)	درجة العلاقة (Multiplicity)
صفة (Attribute)	كائن (Object)
لغة C#	التحليل المعتمد على الكائنات (Object-Oriented Analysis)

لغة C++	التصميم المعتمد على الكائنات (Object-Oriented Design)
درجة العلاقة بين الجداول (Cardinality)	تطوير النظم المعتمد على الكائنات (Object-Oriented Development)
صنف (Class)	كائن دائم (Persistent Object)
تغليف العناصر (Encapsulation)	تعدد الأشكال (Polymorphism)
شجرة التعميم/التخصيص (Generalization/Specialization Hierarchy)	كائن مجال المشكلة (Problem Domain Object)
هوية (Identity)	إعادة الاستخدام (Reusability)
إخفاء المعلومات (Information Hiding)	لغة Simula
التوارث (Inheritance)	لغة SmallTalk
شجرة التوارث (Inheritance Hierarchy)	الصنف الفرعي (Sub-Class)
لغة Java	الصنف الرئيس (Super Class)
رسالة (Message)	ثلاثي الطبقات (Three-Tier)
إجراء (Method)	لغة النمذجة UML

أسئلة المراجعة

Review Questions

١- تحليل برمجة نظم المعلومات وتصميمها باستخدام اللغة المعتمدة على الكائنات والتكنولوجيا والتقنيات تدعى :

(أ) تطوير الكائنات الموجهة.

(ب) التطوير الإجرائي.

(ج) الوراثة.

(د) هندسة المعلومات.

٢- يحتوي تطوير OO "الكائنات الموجهة" على :

(أ) فقط OOA و OOP.

(ب) فقط OOD و OOP.

(ج) OOA و OOD و OOP.

(د) التحليل والتصميم المنظم.

٣- أيٌّ من اللغات التالية ليست مثالاً على لغة الكائنات الموجه :

(أ) SmallTalk

(ب) Pascal

(ج) C++

(د) Java

٤- أول لغة برمجة تعتمد على الكائنات بشكل عام كانت :

(أ) SmalTalk

(ب) Pascal

(ج) C++

(د) Java

٥- أيٌّ من اللغات التالية ليست من مايكروسوفت :

(أ) C#

(ب) J++

(ج) VB .NET

(د) Java

٦- ما هما المكونان الرئيسان لإطار عمل .نت :

(أ) مكون تنفيذ اللغات المشترك ومكتبة أصناف إطار عمل .نت.

(ب) لغة فيجوال بيسك ولغة Quick Basic.Net.

(ج) Java virtual machine و JDK.

(د) servlets و applets.

٧- هل UML. اختصار لـ:

(أ) Uniform Model Limitation.

(ب) Unknown Meta Language.

(ج) Unified Modeling Language.

(د) Untyped Machine Language.

٨- لكي يتم تطوير "الاعتماد على النماذج" نحتاج إلى :

(أ) التطوير الإجرائي التقليدي فقط.

(ب) تقنية الكائنات فقط.

(ج) أنظمة الهواية الصغيرة "Small hobby systems".

(د) كل من التطوير الإجرائي التقليدي وتقنية الكائنات

٩- أيٌّ من التقنيات الآتية لا ينضم إلى تقنية الكائنات والتطوير الإجرائي التقليدي :

(أ) إدارة المشروع.

(ب) المقابلة.

(ج) اختبار البرنامج.

(د) الرسم التخطيطي للصنف.

١٠- أيٌّ من الآتي ليس رسم معرف UML :

(أ) تخطيط علاقة الكينونة.

(ب) الرسم التخطيطي للصنف.

(ج) تخطيط حالة الاستخدام "use case".

(د) التخطيط المتتالي.

١١- يعرف النظام المعتمد على الكائنات على أنه :

(أ) مجموعة الكائنات التي تقوم بإنجاز مهمة.

(ب) الأصناف والإجراءات المنفصلة عن البيانات.

(ج) تدفق بيانات التطبيقات التي تعالج البيانات إلى معلومات.

(د) أي نظام صُمم للإنترنت.

١٢- يطلق على الشيء الذي له صفات وسلوك :

(أ) السيناريو.

(ب) حالة الاستخدام.

(ج) الكائن.

(د) الطريقة.

١٣- يطلق على خاصية الكائن الذي يأخذ قيمة :

(أ) الصنف.

(ب) حالة الاستخدام.

(ج) الطريقة.

(د) الصفة.

١٤- ما هو الكائن الذي لديه القدرة على العمل من ناحية السلوك:

(أ) الصنف.

(ب) حالة الاستخدام.

(ج) الطريقة.

(د) الصفة.

١٥- يطلق على الكائنات المستخدمة في عمل واجهة استخدام النظام:

(أ) كائنات مجال مشكلة.

(ب) كائنات التعامل مع قاعدة البيانات.

(ج) كائنات واجهة الاستخدام GUI.

(د) الكائنات المرئية.

١٦- يطلق على الكائنات المخصصة للتطبيق العملي:

(أ) كائنات مجال مشكلة.

(ب) كائنات التعامل مع قاعدة البيانات.

(ج) كائنات واجهة الاستخدام GUI.

(د) الكائنات المرئية.

١٧- ماذا يطلق على الطلب المرسل لكائن ما لتنفيذ أحد إجراءاته:

(أ) الأمر.

(ب) الاستدعاء.

(ج) الصفة.

(د) الرسالة.

١٨- يسمى تغليف التركيب الداخلي للكائنات وحمايته:

(أ) إخفاء المعلومات.

(ب) تعدد الأشكال.

(ج) الوراثة.

(د) التعميم والتخصيص.

١٩- كل كائن له عنوان خاص يمكن إيجاده به أو الرجوع إليه أو إرسال رسالة إليه يدعى :

(أ) مفتاح الكائن.

(ب) الصنف المعرف.

(ج) قيمة الفهرس.

(د) هوية الكائن.

٢٠- الكائنات الدائمة هي :

(أ) متوفر للاستعمال بمرور الوقت.

(ب) أبداً لا يستجيب عندما يرسل رسالة.

(ج) مهمة عادة بما فيه الكفاية لكي تعطي أولوية.

(د) عمل واجهة الاستخدام.

٢١- الصنف والكائن هما :

(أ) مختلفان لأن الصنف نسخة والكائن فئة.

(ب) متشابهان.

(ج) مختلفان لأن الصنف مثل النسخة والكائن مثل علاقة الترابط.

(د) مختلفان لأن الصنف نوع من شيء والكائن نسخة محددة.

٢٢- أيٌّ من الآتي له علاقة مترابطة :

(أ) العميل كنوع خاص من الشخص.

(ب) الشخص الذي له صفة يطلق عليه اسم.

(ج) العميل الذي يسجل في برنامج الثمان.

(د) برنامج الثمان كنوع من الحساب.

٢٣- يطلق على رقم علاقة الترابط بين كائنات الأصناف :

(أ) تعدد الأشكال.

(ب) التعدد.

(ج) العلاقات.

(د) الطرق.

٢٤- ما هو الصنف المميز من الصنف سيارة:

(أ) سيارة فيسل.

(ب) السيارة الرياضية.

(ج) فيسل الرياضية.

(د) الشاحنة.

٢٥- ما هو الصنف الفرعي من صنف الشاحنة:

(أ) سيارة فيسل.

(ب) السيارة الرياضية.

(ج) عربة المحطة.

(د) شاحنة النفاية.

٢٦- طبقاً لمفهوم تعدد الأشكال، الخلاط والغسالة هما:

(أ) كلٌّ منهما كائنات ملموسة.

(ب) يمكن أن يكون كلاً منهما يدور بسرعة.

(ج) أنواع الأدوات المنزلية.

(د) كائنات مجال المشكلة لمتجر تجزئة.

٢٧- اثنان من منافع الكائنات الموجهة هما:

(أ) الطبيعة وإعادة الاستخدام.

(ب) الطرق والرسائل.

(ج) علاقة الترابط والتعميم والتخصيص.

(د) العميل والخادم.

٢٨- التصميم بثلاث طبقات يقسم النظام إلى الفئات التالية للأصناف:

(أ) أصناف واجهة المستخدم الرسومية، وأصناف قاعدة البيانات، وأصناف نظام التشغيل.

(ب) أصناف مجال المشكلة، وكائن نظام التشغيل، وأصناف واجهة المستخدم الرسومية.

(ج) أصناف مجال المشكلة، وأصناف واجهة المستخدم الرسومية، وأصناف التعامل مع البيانات.

(د) أصناف واجهة المستخدم الرسومية، والأصناف الإجرائية، وأصناف التعامل مع البيانات.

أسئلة المناقشة

Discussion Questions

١- يناقش هذا الفصل (بل مواضيع الكتاب جميعاً) أن عملية تطوير النظم لا تشتمل فقط على مرحلة البرمجة. اسرد وناقش أربع أنشطة/مهام أو أكثر لا تتعلق بمرحلة البرمجة. وهل يوجد هناك فرق بين عملية تطوير النظم المعتمدة على الكائنات والطريقة التقليدية في هذه الأمور؟

٢- لا تعتبر عملية تطوير النظم المعتمدة على الكائنات جديدة، ولكن مر بعض الوقت لانتشارها كتقنية في تطوير النظم. ناقش بعض الأسباب التي أدت إلى هذا. وهل ما زالت هناك أسباب تعوق استخدام هذه التقنية؟ وهل لغة VB .NET وإطار عمل مايكروسوفت .نت ساهمت بشكل كبير في انتشار هذه التقنية؟

مشاريع الفصل

Projects

١- تحدث مع أحد زملائك عن البرمجة الإجرائية والبرمجة المعتمدة على الكائنات، ثم قم بسرد ومناقشة خمس مواضيع تتعلق بكلا الأسلوبين؟

٢- تعد أصناف مجال المشكلة الجزء الأساسي أثناء مرحلة التحليل. ما هي أصناف مجال المشكلة المطلوبة لتطوير نظام تسجيل مواد داخل كلية ما؟ قم بتوسعة أحد الأصناف وذلك باستخدام كل من الأصناف الفرعية والأصناف الأساسية.

٣- قم بتحديد ثلاث علاقات ترابط على الأقل في المشروع السابق بين أصناف مجال المشكلة.

٤- تستخدم أصناف واجهة الاستخدام لبناء واجهة تطبيقات الويندوز حيث يحتوي نموذج الويندوز على عدة كائنات. قم بسرد العديد من أصناف واجهة الاستخدام وبعض أشجار التوارث لتلك الأصناف. ما هي بعض علاقات ترابط التي توجد بين تلك هذه الأصناف؟