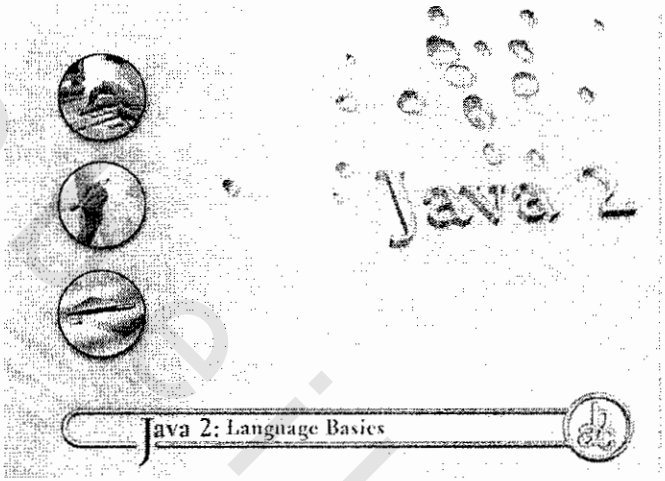
الفصل السادس



مفاهيم البرمجة بواسطة الأهداف
Object Oriented Programming

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

- مفاهيم البرمجة بواسطة الأهداف Object Oriented Programming.
- البرمجة بواسطة الأهداف Object Oriented Programming (OOP).
- معنى الفصيلة class.
- دوال البناء Constructors.
- خاصية التوريثInheritance .
- التحميل الزائد للدوالMethod Overloading .
- إنشاء واستعمال الفصائل Creating & Using Classes.
- قواعد يفضل الالتزام بها عند كتابة البرامج.
- ملاحظات يجب مراعاتها عند تصميم الفصائل classes.

مفاهيم البرمجة بواسطة الأهداف Object Oriented Programming:

- تعتبر البرمجة بواسطة الأهداف Object Oriented Programming من الاتجاهات الحديثة للبرمجة ، ومن أوائل اللغات التي طبقت هذا المفهوم لغة C++ ، وكذلك من اللغات التي تلتزم بمفهوم البرمجة بواسطة الأهداف OOP لغة Java.
- ولتوضيح ذلك تابع معنا شرح هذه الفقرة حيث نتناول النقاط التالية:

1. معنى البرمجة بواسطة الأهداف Object Oriented Programming.
2. معنى الفصيلة class.
3. دالة البناء Constructor.
4. خاصية التوريث Inheritence.
5. التحميل الزائد للدوال Method Overloading.

البرمجة بواسطة الأهداف Object Oriented Programming (OOP):

- تعتمد فكرة البرمجة بواسطة الأهداف (OOP) علي استعمال الفصيلة Class كوحدة برمجة ، حيث كانت فكرة البرمجة التقليدية تعتمد علي استعمال الدوال Functions لبناء البرنامج ، وهذا كان يجعل المبرمج مضطراً لإعادة كتابة الأوامر في كل مرة لتحقيق فكرة معينة ، ولكن جاءت البرمجة بواسطة الأهداف (OOP) لتجعل وحدة بناء البرمجة عبارة عن فصيلة class ، وبالتالي يتم إعداد مجموعة من الفصائل العامة classes التي تلبي معظم متطلبات المبرمج لإعداد برنامج ، حتى أن بعض المبرمجين يشبه البرمجة بواسطة الأهداف OOP بالبناء باستعمال المباني الجاهزة ، والبرمجة بالطريقة التقليدية القديمة تشابه البناء باستعمال الأدوات الأولية وبالتالي الفرق بينهما في السرعة كبير جداً.

معنى الفصيلة class:

- الفصيلة class هي أساس البرمجة بواسطة الأهداف (OOP) وهي التي يبنى عليها البرنامج ، وفكرة الفصيلة Class جاءت من الواقع ، فكل عنصر من عناصر الحياة عبارة عن فصيلة class ، فأنت تستطيع أن تطلق علي جميع السيارات أنها من فصيلة Car مع بعض الاختلافات بين كل سيارة وأخري ، ويمكنك أن تطلق على

الطيور فصيلة Bird، وهكذا تنتمى جميع العناصر إلى فصائل classes ، وكل فصيلة class تستطيع تمثيلها بعنصرين هما البيانات والدوال Methods ، فمثلاً فصيلة الموظف Employee بياناتها هي بيانات الموظف العامة مثل كود الموظف واسم الموظف وعنوان الموظف وتليفون الموظف وباقي بياناته ، وكذلك الدوال Methods هي الوظائف أو العمليات التي يمكن أن تتم على الموظف مثل عملية إضافة موظف جديد وحذف موظف موجود وتعديل بيانات موظف موجود.

- إذن الفصيلة class هي مجموعة من السطور التي تمثل عنصراً من العناصر تمثيلاً تاماً من حيث بيانات العنصر والتي تسمى خصائص properties ، وكذلك دوال العنصر التي تسمى methods ، وبتقسيم البرنامج إلي فصائل classes يصبح البرنامج أكثر نظاماً وأسرع في الإعداد.
- وقد قامت شركة صن Sun في لغة Java بإعداد مجموعة كبيرة من الفصائل classes التي تلبي متطلبات المبرمج ، وما عليك عند إعداد البرامج إلا أن تدرس مكتبة الفصائل class library الموجودة لتعرف المتوفر منها ، وكذلك لتتعرف علي بيانات proprrties ودوال Methods الفصيلة class.
- والسؤال الآن هو كيف أستطيع إنشاء فصيلة Class والتعامل معها وتطبيق مفاهيم OOP؟
- الإجابة: هذا ما سوف نوضحه في هذا الفصل بعد الانتهاء أولاً من توضيح المفاهيم الأساسية لمفهوم OOP.

دوال البناء Constructors:

- دوال البناء Constructors هي دوال Methods يتم تنفيذها تلقائياً عند استعمالك للفصيلة class، وهذا يتم عند تعريف عنصر object ، حيث يمكن استعمال دالة البناء Constructor في تسجيل أي قيم ابتدائية Initial Values لمتغيرات الفصيلة class.

التوريث Inheritance :

- معنى خاصية التوريث Inheritance هو إنشاء فصيلة تعرف باسم الفصيلة الأساسية Base class وهي عبارة عن فصيلة class قديمة موجودة بالفعل ، ثم إنشاء فصيلة class جديدة ترث من الفصيلة الأساسية Base class ، وهذا يعني

أنه ستتم إضافة تركيب الفصيلة الأساسية Base class القديمة من بيانات ودوال Methods إلي الفصيلة class الجديدة ثم نكمل عليها في الفصيلة class الجديدة بعض البيانات والدوال Methods الخاصة بهذه الفصيلة class الجديدة.

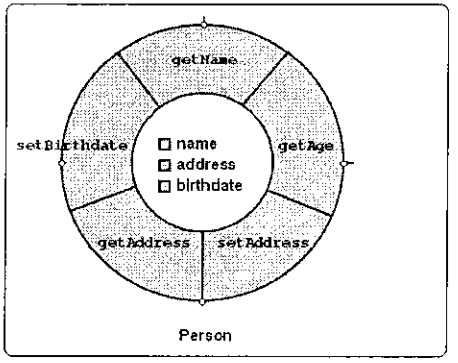
- وهذه الخاصية من أهم خصائص مفهوم OOP ، فعلى أساسه تبنى مكتبات الفصائل Class library ، حيث يتم بناء فصيلة أساسية Base class ثم ترثها الفصيلة class الثانية والثالثة ، وتأخذ الفصائل classes من بعضها البعض حتى تتكون مكتبة فصائل Class library عبارة عن شجرة فصائل Class Tree ، وهذا هو الحال في مكتبة الفصائل الشهيرة MFC الخاصة بميكروسوفت Microsoft فى لغة Visual C++ وكذلك مكتبة فصائل لغة Java المعروفة بالاسم Java Foundation Classes (JFC).

التحميل الزائد للدوال Method Overloading :

- معنى خاصية التحميل الزائد للدوال Method Overloading هو إمكانية إنشاء أكثر من دالة Method بنفس الاسم مع تغيير عدد المعاملات Parameters أو نوع البيانات ، وهذا يفيد بإنشاء أكثر من دالة Method لنفس الوظيفة بمعاملات Parameters مختلفة بنفس الاسم.
- فمثلاً يمكننا إنشاء دالتين Methods بالاسم ( )findEmployee إحداهما تأخذ معاملاً Parameter رقمياً يمثل كود الموظف ، والثانية تأخذ معاملاً Parameter عبارة عن نص String يمثل اسم الموظف ، وفي الحالتين تبحث الدالة Method عن الموظف المطلوب إيجاده ، وبالتالي يشعر المبرمج كأنها دالة Method واحدة ولكنهما في الواقع دالتين Methods يتم استدعاء كل واحدة تلقائياً حسب المعامل Parameter المرسل لها.

عزيزي القارئ:
حاول أن تنظر حولك وستجد أن كل شئ حولك هو هدف Object من فصيلة class.
حاول مراجعة كل العناصر من حولك وتحديد الفصيلة class التابع لها.

وتأخذ الفصيلة class الشكل العام الواضح فى الشكل (6-1).



(الشكل 6-1)

- فى هذا الشكل نفترض أن لدينا فصيلة class باسم Person تمثل بيانات Data وعمليات Methods لشخص ما ، البيانات هى الاسم name والعنوان address وتاريخ الميلاد birthdate ، والدوال methods هى مجموعة من الدوال methods التى تستطيع تسجيل قيم هذه المتغيرات وتبدأ بـ set ومجموعة أخرى لإعادة قيم هذه المتغيرات وتبدأ بـ get.
- وبالتالى تلاحظ أن التركيب العام للفصيلة class يأخذ الشكل التالى:

```
class ClassName
{
  constructor1
  constructor2
  . . .
  method1
  method2
  . . .
  field1
  field2
  ...
}
```

- فى هذه السطور يتم تعريف الفصيلة class باستخدام كلمة class ثم يليها اسم الفصيلة class ثم الأقواس ثم المحتويات وهذا ما سوف نتناوله فى الفقرات التالية.

إنشاء واستعمال الفصائل Creating & Using Classes:

- بعد شرح بعض المفاهيم المهمة لمفهوم OOP ، تعال معنا نشرح عملياً كيفية إنشاء الفصائل Classes وكيفية تحقيق هذه المفاهيم.
- الفصيلة class هى أساس مفهوم OOP ، لذلك أول ما نتعلمه فى هذا المفهوم هو كيفية إنشاء واستعمال الفصيلة class ، ولتحقيق ذلك تابع معى المثال التالي.

مثال (1): إنشاء الفصائل Classes:

- قم بانشاء تطبيق جديد.
- اكتب سطور البرنامج كما فى الشكل (6-2).

FirstClass.java

```
1  public class FirstClass {
2
3      public static void main(String[] args) {
4
5          System.out.println("this the main class ");
6      }
7  }
```

(الشكل 6-2)

- فى هذه السطور تلاحظ أنها أبسط أشكال برامج لغة Java ، ويتكون من فصيلة class واحدة وهى الفصيلة الرئيسية ، وبالتالى لابد أن يحتوى البرنامج في لغة Java على فصيلة class واحدة على الأقل وبها الدالة ()main التى يبدأ منها تنفيذ البرنامج.
- فى السطر رقم 1 يبدأ تعريف الفصيلة class بالاسم FirstClass مع استعمال الكلمة المحجوزة class لتعريف هذه الفصيلة class.
- فى السطر رقم 3 نجد الدالة الرئيسية ()main وبالتالي تم إنشاء أبسط شكل للفصيلة class ، وسنقوم في الأمثلة القادمة بتوضيح المزيد من محتويات الفصيلة class.

مثال (2): إنشاء الفصائل Classes:

المثال التالى يتناول موضوع الفصائل classes بشكل أكثر إيضاحاً. قم بكتابة الكود كما فى (الشكل 6-3).

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
20 public class Class2 {
21
22    public static void main(String[] args) {
23
24    }
25 }
26
```

(الشكل 6-3)

شرح السطور:

- في السطر رقم 1 تم إنشاء فصيلة بالاسم MathClass وذلك باستعمال الكلمة المحجوزة class حتى يمكن التعامل مع الفصيلة class.
- في السطر رقم 3 تم الإعلان عن المتغيرات a,b,c من النوع int.
- في السطر رقم 5 تم إنشاء دالة Method بالاسم printABC() مع وضع كلمة public قبلها حتى يمكن التعامل مع هذه الدالة Method مباشرة من خارج الفصيلة class وسوف يتم توضيح هذه النقطة لاحقاً.
- داخل سطور هذه الدالة Method تم تحديد قيم للمتغيرات a,b,c كما في السطور 7,8,9 كما تم استعمال System.out.println() ثلاث مرات لطباعة قيم المتغيرات a,b,c كما في السطور 10,11,12.
- وبهذا تم إنشاء فصيلة class جديدة بالاسم MathClass التي تحتوي على المتغيرات (a,b,c) والدالة printABC() كأعضاء لهذه الفصيلة class.
- من فضلك قم بمراجعة سطور إنشاء الفصيلة MathClass جيداً وحاول استيعاب كيفية إنشاء وكتابة سطور هذه الفصيلة class.

- والخطوة التالية هى كيفية استعمال هذه الفصيلة class ، ولتوضيح ذلك تابع الخطوات التالية.
- عد إلى البرنامج ثم اكتب سطور استعمال الفصيلة MathClass داخل الدالة الرئيسية main() كما فى (الشكل 6-4).

```
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  }
```

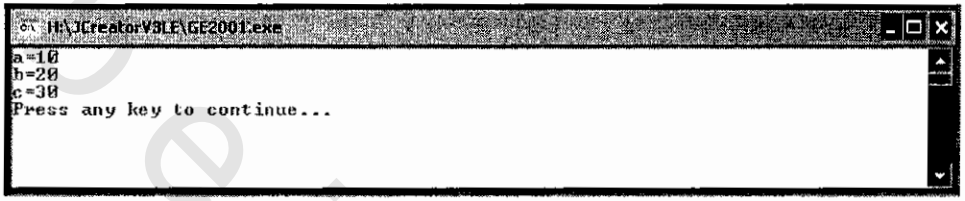
(الشكل 6-4)

- فى هذا السطور تم استعمال الفصيلة MathClass كما يلى.
- فى السطر رقم 22 تم الإعلان عن متغير بالاسم obj1 من نوع الفصيلة MathClass فهو فى هذه الحالة لا يسمى متغيراً بل يسمى هدفاً Object مع استعمال الكلمة new لإنشاء هدف object بالذاكرة ، وبالتالى تكون أول خطوة لاستعمال الفصيلة class هى تعريف هدف object من الفصيلة class كما فى السطر التالى.

```
Myclass Obj1=new Myclass();
```

- وبالتالى فإن أى تعامل مع الفصيلة class لا يتم من خلال اسمها بل يتم من خلال هدف object منها مع ملاحظة إمكانية تعريف أكثر من هدف object وبالتالى تغيير القيم فى كل هدف object.

- فى السطر رقم 23 تم استدعاء الدالة printABC() للهدف obj1 وبالتالي يتم تنفيذ هذه الدالة method ويتم طباعة قيم المتغيرات.
- قم بتنفيذ البرنامج وستلاحظ استدعاء السطور المكتوبة داخل الدالة الرئيسية main() حيث يتم طباعة قيم المتغيرات a,b,c باستدعاء الدالة printABC() كما فى الشكل (6-5).



(الشكل 6-5)

عزيزي القارئ:

حاول التدرب كثيراً على إعداد الفصائل classes بما تحتويه من بيانات ودوال Methods وكذلك التدرب على استعمال هذه الفصائل classes بتعريف هدف object ثم استدعاء الدوال Methods. ولزيادة التوضيح تابع معنا المثال التالى.

مثال (3): الأهداف Objects:

- قم بانشاء تطبيق جديد بالخطوات المعتادة كما سبق.
- قم بكتابة سطور إنشاء فصيلة class جديدة كما فى الشكل (6-6).

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 }
```

(الشكل 6-6)

- فى هذا الشكل تم انشاء فصيلة class جديدة بالاسم operations وذلك باستعمال الكلمة المحجوزة class.
- داخل سطور الفصيلة class الجديدة operations قم بتعريف متغيرات جديدة كما فى السطر التالى.

public double a, b, c;

- داخل سطور الفصيلة operations قم بإنشاء دوال methods العمليات الحسابية ، فقم بتعريف دالة method لعملية الجمع وأخرى لعملية الطرح وعملية الضرب وعملية القسمة لتصبح الفصيلة operations كما فى (الشكل 6-7).

Mathoperations.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) {

   }
}
```

(الشكل 6-7)

- راجع هذه السطور وستلاحظ أننا أنشأنا الفصيلة operations وبها مجموعة دوال methods العمليات الحسابية التى أشرنا إليها.
- قم باستعمال الفصيلة class الجديدة كما فى (الشكل 6-8).

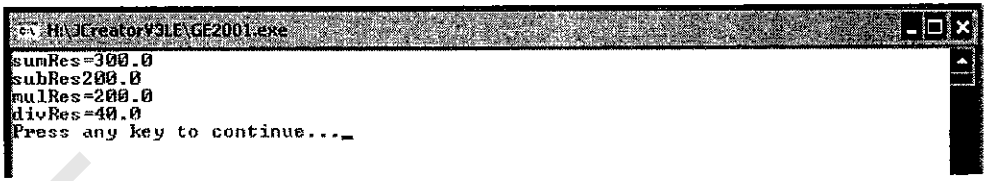
```
Mathoperations.java *
 1  class operations
 2  {
 3    public double a,b,c;
 4    public double sum(double v1,double v2)
 5      {
 6      double res;
 7      res=v1+v2;
 8      return res;
 9          }
10      public double sub(double v1,double v2)  {
11      double res;
12      res=v1-v2;
13      return res;
14          }
15      public double mul(double v1,double v2)
16    {
17      double res;
18      res=v1*v2;
19      return res;
20        }
21       public double div(double v1,double v2)
22    {
23      double res;
24      res=v1/v2;
25      return res;
26        }    }
27     public class Mathoperations {
28       public static void main(String[ ] args) {
29    operations obj=new operations();
30    double sumRes,subRes,mulRes,divRes;
31      sumRes=obj.sum(100,200);
32    subRes=obj.sub(300,100);
33    mulRes=obj.mul(10,20);
34    divRes=obj.div(200,5);
35     System.out.println("sumRes="+sumRes);
36    System.out.println("subRes"+subRes);
37    System.out.println( "mulRes="+mulRes);
38    System.out.println("divRes="+divRes);
39      }}
```

(الشكل 6-8)

شرح السطور:

- فى السطر رقم 29 تم تعريف هدف object بالاسم obj من الفصيلة operations وذلك باستعمال كلمة new كما سبق أن شرحنا.
- في السطر رقم 30 يتم الإعلان عن ثلاثة متغيرات من النوع double.
- في السطر رقم 31 يتم استدعاء الدالة ()sum مع الهدف obj مع إرسال معاملين Parameters هما 100 و 200 لجمعهما ، ثم يتم وضع النتيجة فى المتغير sumRes.
- وبالمثل يتم استدعاء دالة method الطرح والضرب والقسمة ثم يتم طباعة ناتج هذه العمليات الأربعة.

قم بتشغيل البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (6-9).



(الشكل 6-9)

الجديد فى هذا البرنامج:

لم يكن الغرض من هذا البرنامج هو تحقيق عمليات الجمع والطرح والضرب والقسمة وإلا كنا حققناها فى أول الكتاب ببساطة وبدون استعمال فصائل classes ، وإنما الغرض من البرنامج هو توضيح كيفية إنشاء فصيلة class وكيفية إضافة بيانات ودوال methods لها ، فأرجو أن تكون الصورة واضحة.

دوال البناء Constructors:

من الخصائص المتوفرة فى مفاهيم البرمجة بالأهداف OOP هي فكرة وجود دالة البناء Constructor داخل الفصيلة class وهى دالة Method مثل أى دالة Method عادية ولكن الفرق فى أنها دالة Method تنفذ تلقائياً بدون استدعاء بمجرد تعريف هدف object من الفصيلة class.

وتتميز دالة البناء Constructor عن الدالة Method العادية في أن اسمها لا بد أن يكون بنفس اسم الفصيلة class التي تحتويها ، كما لا تقوم بإرجاع أي قيمة ولا حتي void.

والغرض من ذلك هو استعمال هذه الدالة method فى تنفيذ أى عمليات أولية للفصيلة class مثل إعطاء قيم ابتدائية Initial Values للمتغيرات.

ويتم تحديد دالة البناء Constructor داخل الفصيلة class بإنشاء دالة method بنفس اسم الفصيلة class أى تصبح كما فى الشكل (6-10).

Mathoperations.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 }
```

(الشكل 6-10)

- قم بتنفيذ البرنامج وستلاحظ ظهور رسالة دالة البناء Constructor بالرغم من عدم استدعائه حيث تم الاستدعاء تلقائياً.
- تأكد من الحصول على النتيجة كما فى الشكل (6-11).

```
H:\JCreatorV3LE\GE2001.exe
this is the Constructor >>> it run without calling
Press any key to continue...
```

(الشكل 6-11)

التحميل الزائد للدوال Method Overloading:

- من المفاهيم المشهورة فى البرمجة بالأهداف OOP مفهوم إنشاء أكثر من دالة method باسم واحد ويطلق عليه التحميل الزائد للدوال Method Overloading ، حيث يسمح هذا المفهوم بإنشاء أكثر من دالة method بنفس الاسم إذا دعت الحاجة لذلك مثل إنشاء مجموعة دوال رسم بالاسم draw() تستطيع رسم خط Line ومربع Square ودائرة Circle وغيره.
- ومثال أخر يمكن إنشاء أكثر من دالة method بحث بالاسم search() ، واحدة للبحث عن موظف بمعلومية رقم الموظف والثانية للبحث عن الموظف باسم الموظف ، ويتم استدعاء الأولى أو الثانية حسب المعامل Parameter الذى تم إرساله.

ولتوضيح فكرة التحميل الزائد للدوال Method Overloading ، تابع المثال التالى.

مثال (4): التحميل الزائد للدوال Method Overloading:

قم بإنشاء تطبيق جديد بالخطوات المعتادة كما سبق.

قم بتعديل سطور البرنامج وقم بإنشاء فصيلة class جديدة بالاسم Shapes كما فى الشكل (6-12).

Mathoperations.java ShapesApp.java

```
class shapes
{
   public void drawline ()
   {
      for(int i=0;i<10;i++)
      System.out.print("*");
      System.out.println("");
      }

   public void drawline (int n)
   {
      for(int i=0;i<n;i++)
      System.out.print("*");
      System.out.println("");
      }

      public void drawline (int n,char ch)
   {
      for(int i=0;i<n;i++)
      System.out.print(ch);
      System.out.println("");
      }

   }

public class ShapesApp {

   public static void main(String[] args) {
      shapes obj=new shapes();
      obj.drawline();
      obj.drawline(20);
      obj.drawline(30,'-');

   }
}
```

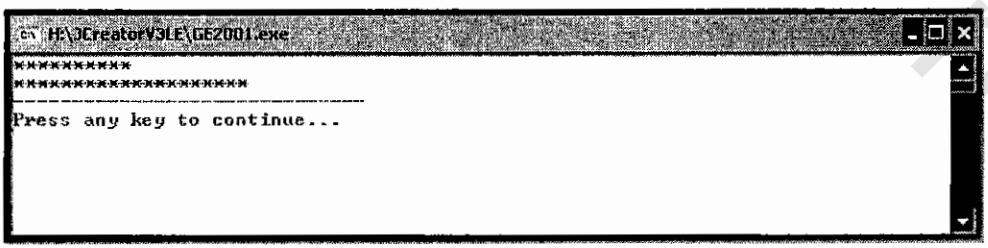
(الشكل 6-12)

شرح السطور:

فى هذه السطور تم إنشاء فصيلة class بالاسم Shapes وداخل هذه الفصيلة class تم تعريف ثلاث دوال methods بالاسم ()drawLine كلها باسم واحد ولكن الملاحظ أن الفرق بينها هو المعاملات Paramters ، فالدالة method الأولى بدون

معاملات Parameters وتقوم بطباعة الحرف * عدد 10 مرات ، والدالة method الثانية بمعامل Parameter واحد هو n من نوع int ويستخدم لتحديد عدد مرات طباعة الحرف * ، والدالة method الثالثة لها معاملين Parameters ، الأول هو n من نوع int ويستخدم لتحديد عدد مرات طباعة الحرف ، والثانى ch لتحديد الحرف المطلوب طباعته.

- بعد إنشاء الدوال methods الثلاثة داخل الفصيلة Shapes ، قم بكتابة أوامر استعمال الفصيلة Shapes ودوالها methods الثلاثة داخل الدالة الرئيسية للبرنامج ()main كما فى السطور من 29 إلي 32.
- في السطر رقم 29 تم تعريف هدف Object من الفصيلة Shapes بالاسم obj كما سبق شرحه.
- فى السطر رقم 30 تم استدعاء الدالة ()drawLin مع الهدف obj ولكن الملاحظ أننا لم نكتب معاملات Parameters للدالة method ، لذلك فإننا نستدعى الدالة method الأولى التى لا تأخذ أي معاملات Parameters.
- في السطر رقم 31 تم استدعاء الدالة drawLine(20) مع الهدف obj ولكن الملاحظ أننا كتبنا معاملاً Parameter للدالة method ، لذلك فإننا نستدعى الدالة method الثانية التى تأخذ معاملاً Parameter واحداً.
- فى السطر رقم 32 تم استدعاء الدالة drawLine(30, "-") مع الهدف obj ولكن الملاحظ أننا كتبنا معاملين Parameters للدالة method ، لذلك فإننا نستدعى الدالة method الثالثة التى تأخذ معاملين Parameters.
- قم بتنفيذ البرنامج لتحصل على نتيجة التنفيذ كما فى الشكل (6-13).



(الشكل 6-13)

قواعد يفضل الالتزام بها عند كتابة البرامج:

- الشكل التالى يعرض التركيب المثالي 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 classes:

- حاول أن تحافظ على أن تكون البيانات من النوع private.
- حاول أن تعطى قيم ابتدائية Initial Values للمتغيرات.
- لا تستعمل متغيرات منفردة كثيراً ، فبدلاً من ذلك حاول تجميعها فى فصيلة class ، فمثلاً بالنظر الى المتغيرات التالية:

```
private String street;
private String city;
private String state;
private int zip;
```

- فيمكن تجميعها فى فصيلة class بالاسم Address.
- 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
```

- حاول تقسيم الفصائل classes حسب الوظائف وتجنب إنشاء فصيلة class تتناول أكثر من وظيفة ، 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 أنه احتوى على أكثر من وظيفة وهذا التصميم سئ ، فيجب فصل الوظائف بحيث لا تحتوى الفصيلة classes إلا على وظيفة واحدة محددة.
- وبالتالى يصبح التصميم الجيد كما فى السطور.

```
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;
}
```

- حاول إعطاء أسماء للفصائل classes والمتغيرات بحيث تكون معبرة عن الغرض منها ، كما يجب أن تلاحظ أنه من المشهور إنشاء دوال Methods تبدأ بالأسماء set و get حيث أن مجموعة دوال set تقوم بإعطاء قيم لمتغيرات الفصيلة class ، ومجموعة دوال get تقوم بإعادة قيم متغيرات الفصيلة class.

ملخص الفصل:

- تعرضنا في هذا الفصل لشرح أهم مبادئ البرمجة بواسطة الأهداف OOP.
- في الفصل القادم سوف نتعرف - بإذن الله- علي أحد أهم مبادئ البرمجة بواسطة الأهداف OOP وهو مبدأ التوريث Inheritance ، فتابع معنا الفصل القادم.