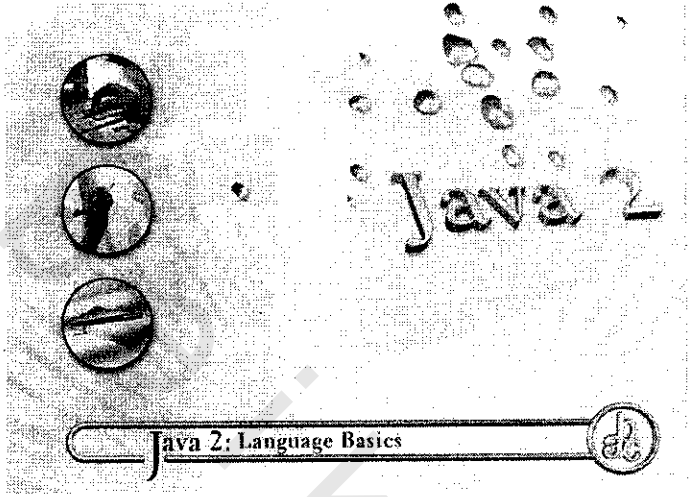
الفصل السابع



الوراثة Inheritance

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:

- خاصية التوريث Inheritance.
- التوريث Inheritance في لغة Java.
- نسخ الدوال Overriding Methods.
- أمر التجريد abstract.
- متى نستعمل خاصية التوريث Inheritance ومتى لا نستعملها؟
- مقارنة المعدل النهائي final modifier ومعدل التجريد abstract modifier.

خاصية التوريث Inheritance :

- هي خاصية قوية جداً في مفهوم البرمجة باستعمال الأهداف OOP ، وتستعمل لتقليل إعادة كتابة الأوامر ، فيمكنك عن طريق استعمال خاصية التوريث Inheritance أن تقوم بإنشاء فصيلة class تحتوى على خصائص Properties ودوال methods ثم يتم استعمال هذه الفصيلة class كأساس لفصائل classes أخرى ، وبالتالي لا نحتاج لكتابة ما كتبناه في تعريف الفصيلة class الأولي (والتي تسمى الفصيلة الأساسية base class) مع كل فصيلة class جديدة .
- وتسمى الفصائل classes الجديدة التي تستورث فصيلة class قديمة باسم فصائل مشتقة derived classes ، وغالباً يتم التعامل مع الفصائل classes بأن نستورث فصيلة classes ونكمل عليها.
- ولتوضيح ذلك نضرب لك المثال التالي.
- نفرض أنك تريد تعريف فصيلتين classes ، الأولي لتعريف بيانات الطالب والثانية لتعريف بيانات المدرس ، فبدون استعمال خاصية التوريث Inheritance سنضطر لإنشاء الفصيلتين classes بالرغم من تشابه الفصيلتين classes واحتوائهما على بيانات متشابهة ، فمثلاً فصيلة class الطالب تحتوى على البيانات التالية: الكود ، الاسم ، العنوان ، التليفون ، المؤهل ، تاريخ الميلاد ، وكثير من البيانات الأخري.
- وتحتوى على الدوال methods التالية: دالة تسجيل بيانات الطالب ، دالة حذف بيانات الطالب ، دالة تعديل بيانات الطالب ودوال أخرى.
- وعند النظر إلي فصيلة class المدرس سوف تجد أنها تشتمل علي كثير من البيانات والدوال methods الأعضاء في فصيلة class الطالب بالإضافة لبعض الأعضاء members الجديدة.
- وبالتالي باستعمال خاصية التوريث Inheritance ، فعلينا توريث فصيلة class الطالب وفصيلة class المدرس من فصيلة class أساسية تحتوي علي البيانات المتشابهة ، ويتم إضافة الجديد فقط في فصيلة class الطالب وفصيلة class المدرس ، مما يوفر علينا إعادة كتابة كل شيء.

- ويمكن تكرار هذه الفكرة مع فصائل classes جديدة أخرى مما يوفر الكثير من الوقت ، بالإضافة إلي أن هذا الأسلوب يسمح لك باستعمال فصائل classes تم إعدادها وتم اختبارها.

التوريث Inheritance في لغة Java:

- يتم إنشاء فصائل classes كاملة بكل دوالها methods ومتغيراتها (تعتبر كفصائل أساس Base class) ثم يتم توريث هذه الفصائل classes بكل دوالها methods ومتغيراتها لفصائل classes جديدة ، وذلك يوفر عليك الكثير من الوقت.
- ويمكنك تنفيذ هذه الفكرة كما في المثال التالي.

مثال (1): التوريث Inheritance:

اكتب سطور تعريف فصيلة class بالاسم Person كما فى الشكل (7-1).

```
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 InhertApp1 {

   public static void main(String[] args) {
    | }
}
```

(الشكل 7-1)

شرح السطور:

- في هذه السطور تم إنشاء فصيلة class بالاسم Person وتم تعريف متغيرات فيها حيث تم كتابة فصيلة class متكاملة تحتوى على المتغيرات Paddress و Pname وتحتوى على الدالة Set_Data() التى تقوم بتسجيل بيانات الفصيلة class ، كما تحتوى على الدالة Get_Name() التى تعيد قيمة المتغير Pname الخاص بالفصيلة class والذي يمثل اسم الشخص ، وكذلك الدالة Get_Address() التى تعيد قيمة المتغير Paddress الذي يمثل عنوان الشخص.
- وعند إنشاء فصيلة class جديدة ترث من الفصيلة Person ، يتم استعمال متغيراتها ودوالها methods مباشرة دون الحاجة إلي إنشائها مرة أخرى.
- وتوريث هذه الفصيلة class لفصيلة class جديدة يتم كما يلي.
- اكتب سطور فصيلة class جديدة بالاسم Student كما يوجد فى الشكل (7-2).

```
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  }
40
```

(الشكل 7-2)

شرح السطور:

- تم إنشاء فصيلة class جديدة بالاسم Student وتم استعمال العبارة extends Person في تعريف الفصيلة class ومعناها قم بالوراثة من الفصيلة Person بكل

ما فيها للفصيلة class الجديدة Student ، وبالتالي أصبحت نفس أعضاء members الفصيلة class القديمة Person أعضاء فى الفصيلة class الجديدة Student ، وهذا يسمح لك باستعمالها مباشرة وذلك كما يلي.

- فى الدالة الرئيسية ()main ، قم بكتابة سطور استعمال كلاً من الفصيلتين Person, Student كما فى الشكل (7-3).

```
Mathoperations.java | ShapesApp.java | InhertApp1.java
42  public class InhertApp1 {
43
44      public static void main(String[] args) {
45          String vname,vaddress;
46          Person no=new Person();
47          no.Set_Data("Azab","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  }
```

(الشكل 7-3)

شرح السطور:

- في السطر رقم 45 تم تعريف المتغيرات vname, vaddress.
- في السطر رقم 46 تم تعريف المتغير no الذي يمثل هدفاً object من الفصيلة Person.
- فى السطور 47 و 48 و 49 تم التعامل مع هدف object الفصيلة Person باستدعاء الدوال methods والتعامل معها بطريقة عادية.

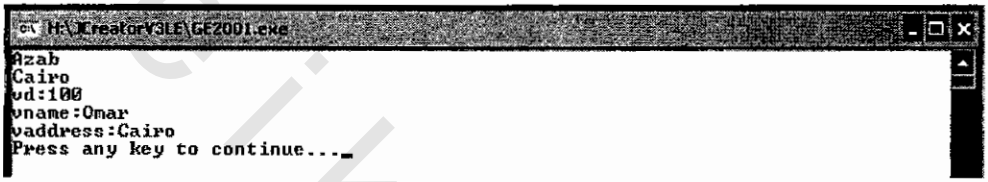
- في السطر رقم 53 تم تعريف المتغير st من الفصيلة Student .
- في السطر رقم 54 تم استدعاء الدالة method الجديدة المضافة للفصيلة التي لها الاسم Set_Degree().
- في السطر رقم 55 و 56 تم الإعلان عن بعض المتغيرات.
- في السطر رقم 58 تم استدعاء الدالة Set_Data() المعرفة داخل الفصيلة class الأساسية Person ، وبالرغم من عدم وجود الدالة method داخل الفصيلة class الجديدة إلا أننا استعملناها لأنها معرفة فى الفصيلة class التى ورثنا منها باستعمال الأمر extends ، وبالمثل يتم ذلك في السطر رقم 60 و 61.
- من هذا المثال تلاحظ استعمال الخصائص المستورثة Inherited Members من الفصيلة Person مع الفصيلة Student بالرغم من عدم تعريفها أو عدم الإعلان عنها.
- والسطور التالية تعرض النص الكامل للبرنامج.

```
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 InhertApp1 {
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);
}
}
```

نفذ البرنامج تحصل على نتيجة التنفيذ كما فى الشكل (7-4).



(الشكل 7-4)

نسخ الدوال Overriding Methods:

من الإمكانيات المتاحة في مفهوم البرمجة بواسطة الأهداف OOP خاصية تسمى نسخ الدوال Overriding Methods ، وهذه الخاصية تعنى إمكانية كتابة دوال methods جديدة في الفصائل classes الجديدة التي تم توريثها بنفس اسم الدوال methods الموجودة في الفصيلة الأساسية Base class وبنفس المعاملات Parameters ونفس نوع القيمة المرتجعة Return Type ، أي أن الدالة methods موجودة في الفصيلة الأساسية Base class ومع ذلك تم إنشاءها مرة أخرى بنفس الاسم في الفصيلة class الجديدة وبالتالي يتم إلغاء الدالة method من الفصيلة class القديمة واعتماد الدالة method الجديدة ، وهذا يسمى نسخ الدالة Overriding Methods ، ولتوضيح ذلك ننشئ فصيلة class بها دالة method ثم نستورث هذه الفصيلة class وننشئ نفس الدالة method السابقة في الفصيلة class الجديدة ، ويظهر ذلك كما فى الشكل (7-5).

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 }
```

(الشكل 7-5)

شرح السطور:

- في السطر رقم 1 تم تعريف فصيلة class جديدة بالاسم one.
- في السطر رقم 3 تم تعريف دالة method بالاسم Say() .
- في السطر رقم 9 تم إنشاء فصيلة class بالاسم Two ترث من الفصيلة One.
- في السطر رقم 11 تم إنشاء دالة method للفصيلة class الجديدة بنفس الاسم Say() وهو نفس اسم الدالة method الموجودة في الفصيلة الأساسية Base class التي تم توريثها وهي الفصيلة One.
- بهذا يصبح للفصيلة الأساسية One دالة method بالاسم Say() ويصبح للفصيلة class الجديدة Two دالة method بالاسم Say() أيضاً.
- وعند تعريف متغير من كل منهما واستدعاء الدالة Say() ، فإنه يتم استدعاء الدالة method الخاصة بكل منهما بالرغم من وجود خاصية التوريث Inheritance بينهما.
- ولتوضيح ذلك تابع ما يلي.

بعد كتابة سطور الفصيلتين classes ، اكتب السطور التالية فى الدالة الرئيسية للبرنامج ()main كما يوجد فى الشكل (7-6).

```
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     }
```

(الشكل 7-6)

شرح السطور:

- في السطر رقم 21 و 22 يتم تعريف المتغيرات a , b من الفصائل One, Two.
- في السطر رقم 23 يتم استدعاء الدالة ()Say مع المتغير a الذي من نوع الفصيلة One ، وبالتالي يتم استدعاء الدالة method الأولي وتظهر الرسالة الأولي.
- في السطر رقم 24 يتم استدعاء الدالة ()Say مع المتغير b الذي من نوع الفصيلة Two ، وبالتالي يتم استدعاء الدالة method الثانية وتظهر الرسالة الثانية ، وذلك بالرغم من توريث الفصيلة class الأولي المحتوية علي الدالة ()Say أيضاً ، الإ أن إنشاء دالة method جديدة بنفس الاسم والمعاملات Parameters ونوع القيمة المرتجعة Return Type أدي إلي إلغاء الدالة ()Say مع الفصيلة class الجديدة واستعمال الدالة method الخاصة بها.
- نفذ البرنامج تحصل على نتيجة التنفيذ كما فى الشكل (7-7).

```
H:\JCreatorV5LE\GE2001.exe
This Msg from class One
This Msg from class Two
Press any key to continue...
```

(الشكل 7-7)

أمر التجريد abstract:

- هذا الأمر عندما يوضع أمام دالة method في الفصيلة class فإنه يعني عدم إمكانية استدعاء هذه الدالة method مباشرة ولا يتم كتابة جسم الدالة method body ، بل لابد من تعريف دالة method جديده بنفس الاسم والمعاملات Parameters ونوع القيمة المرتجعة Return Type حتى يمكن استعمالها ، وبالتالي فهذه الدالة method تكتب للتعريف فقط ، أى لابد من تحقيق خاصية نسخ الدوالMethod Overriding .
- ويتضح ذلك من السطور التالية.

```
abstract class WashingMachine
{
  public WashingMachine()
  {
    // Code to initialize the class goes here.
  }

  abstract public void Wash();
  abstract public void Rinse(int loadSize);
  abstract public long Spin(int speed);}
```

```
class MyWashingMachine extends WashingMachine
{
  public MyWashingMachine()
  {
    // Initialization code goes here.
  }

  override
// public void Wash()
  {
    // Wash code goes here.
  }
```

```
// override
public void Rinse(int loadSize)
  {
    // Rinse code goes here.
  }

// override
public long Spin(int speed)
  {
    // Spin code goes here.
  }
}
```

متى نستعمل خاصية التوريث Inheritance ومتى لا نستعملها؟

- هذا السؤال ربما يرد إلي ذهنك عندما تعمل مع الفصائل classes وإجابته كما يلي.
- إذا كانت الوظيفة المطلوبة تحقيقها ليست بكبيرة بحيث لا تحتاج فصيلة class جديدة تقوم فيها بإنشاء دوال method وتعريف متغيرات جديدة ، فلا تستخدم خاصية التوريث Inheritance ، بل تكتب الأوامر التي تحقق العملية فقط.
- مثال لذلك ، إذا كان لديك أداة نص Text Field وتريد تحويل لون الأرقام السالبة إلى اللون الأحمر ، فلا تقم بتوريث الفصيلة JTextField إلى فصيلة class جديدة وتقوم بتعريف دالة method لعملية تغيير اللون ، ولكن اكتب الأوامر التي تحقق العملية لأنها لا تحتاج إلي فصيلة class.
- إذا كانت الفصيلة class الجديدة (التي تستورث فصيلة class قديمة) لن تستفيد كثيراً من الفصيلة الأساسية (المستورثة) Base class ، حيث تريد فقط الوراثة منها لإنشاء دوالها methods من جديد بخاصية نسخ الدوال Overriding ، ففي هذه الحالة يكون التوريث غير جيد ، فالأفضل هو إنشاء interface ثم تنفيذها implement وبالتالي ننشئ الفصائل classes الجديدة حسب تركيب الـinterface وخاصة إذا كان هناك أكثر من فصيلة class سوف نشترك في التركيب فقط.

- إذا كانت الفصيلة class الجديدة (التي تستورث من الفصيلة class القديمة) تحتاج إلي دوال method الفصيلة class القديمة بنفس السطور المكتوبة فيها وتحتاج إلي الخواص والمتغيرات وتريد الإضافة إليها ، فهذه هي الحالة المناسبة للتوريث Inheritance وهو الاستفادة الكبيرة من الفصيلة الأساسية Base class حيث يتم إضافة الجديد بما يناسب العمل.

مقارنة المعدل النهائي final modifier ومعدل التجريد abstract modifier:

- الجدول التالي يلخص معني استخدام المعدل النهائي final modifier ومعدل التجريد abstract modifier مع كل من الفصائل classes والدوال methods والمتغيرات variables.

المعدل Modifier	مع الفصائل classes	مع الدوال methods	مع المتغيرات variables
المعدل النهائي final	لا يمكن الوراثة من هذه الفصيلة class	لا يمكن نسخ الدالة override	لا يمكن تغيير قيمة المتغير variable
معدل التجريد abstract	لا يمكن إنشاء هدف object من هذه الفصيلة class ويجب الوراثة منها	لا يوجد جسم body لهذه الدالة method ويجب نسخها override	لا يمكن استخدامه

ملخص الفصل:

- تعرضنا في هذا الفصل لشرح أحد أهم مبادئ البرمجة بواسطة الأهداف OOP وهو مبدأ الوراثة Inheritance.
- في الفصل القادم نستكمل - بإذن الله - شرح مبادئ البرمجة بواسطة الأهداف OOP حيث نتعرف علي موضوع قواعد التوصل للفصائل والمتغيرات Access Modifiers ، فتابع معنا الفصل القادم.