١٧/٠٦/٣٤

اخر محاضره
١٢ ليلي بمشير
مقدمة في البرمجة

[٦ صفحات]

حجب الطرق Method Overriding :

← عندما يتطابق الاسم ونمط التوقيع بطريقتين (2 methods) إحداهما تابعة للصف الفرعي (sub class) والأخرى تابعة للصف الفوقي (suber class) فان الطريقة الفرعية تحجب الطريقة الفوقية.

← جملة super تمكن من التغلب على حجب أعضاء الصف الفوقي:

- يستطيع الصف الفرعي استدعاء بانية يعرفها صفه الفوقي باستخدام super الشكل التالي:

super(paramerters);

- يمكن استخدام super للنفاذ الى اعضاء صف فوقي بالشكل:

Super.memper

والعضو member كما هو معروف يمكن أن يكون متغير أو طريقة.

+ مفهوم حجب الطرق Method Overriding (مهم)

مثال prog58 :

```
/* المثال يوضح حجب الطرق
   حيث تحجب الطريقة الفرعية الطريقة الفوقية التي
   تطابقها في الاسم .
*/
class A{
    int i,j;
    A (int a , int b) {  i=a;       j=b;
    }// constructor A
    void show(){
    System.out.println("i and j :"+i+"   "+j);
    }// end method show()
}// end class A

class B extends A{   int k;
    B(int a, int b, int c) {
        super(a,b);
        k=c; } // end of constructor B
    void show(){
            System.out.println("k="+k);
    }// end method show()
}// end class B

class override{
  public static void main(String[] args){
        B b;
        b=new B(1,2,3);
        b.show();
}  }
```

النتيجة:

```
K= 3
```

مثال prog59 :

```
/* استخدام super للوصول للطريقة
   الموجودة الموجودةShow()
   في الصف الفوقي
*/
 class A{
   int i,j;
   A (int a , int b) { i=a;   j=b;
   }// constructor A
   void show(){
System.out.println("i and j :"+i+" "+j);
    }// end method show()
}// end class A
```

```
class B extends A{   int k;
   B(int a, int b, int c) {
      super(a,b);
      k=c; } // end of constructor B
   void show(){
         super.show();
         System.out.println("k  :"+k);
    }// end method show()
}// end class B
class override2{
  public static void main(String[] args){
      B b;
       b=new B(1,2,3);
       b.show();  } }
```

```
i and j: 1 2
K: 3
```

مثال prog60

```
// التحميل الزائد للطريقة show بسبب اختلاف التوقيع
class A{
    int i,j;
    A(int a,int b){     i=a;      j=b;
    } // constructor A
    void show(){
System.out.println("i and j :"+i+"   "+j);
    } // end method show()
} // end class A
class B extends A{          int k;
    B(int a, int b, int c) {
          super(a,b);
          k=c;   } // end of constructor B
    void show(String msg){

         System.out.println(msg +k);
    } // end method show()
} // end class B
```

```
class override{
    public static void main(String[] args){
      B b;
          b=new B(1,2,3);
          b.show("k: ");
          b.show();
}   }
```

```
K: 3
i and j: 1 2
```

+ مفهوم التعدد الشكلي Polymorphism

٣م

التعدد الشكلي polymorphism:

مثال prog61

```
//                                  التعدد الشكلي
class A{
   void who(){
   System.out.println(" I am A .");
   } // end method who ()
} // end class A
class B extends A{
void who(){
   System.out.println(" I am B .");
} // end method who ()
} // end class B
class D extends A{
void who(){
   System.out.println(" I am D .");
} // end method who ()
} // end class D
```

```
class run{
   public static void main(String[]
   args){
      A x;
      A a;      a=new A();
      x=a;
      x.who();
      B b;      b=new B();
      x=b;
      x.who();
      D d;      d=new D();
      x=d;
      x.who();
} //main end
} // class run end
```

```
I am A.
I am B.
I am D.
```

في prog61 اعتمدنا على اختلاف الكائن object (A,B,C) who
في تحديد أي ~~show~~ سيتم تنفيذها، بينما في prog60 اعتمدنا على اختلاف التوقيع (ما بين القوسين) في تحديد show .

البريمجات Applets :

تستطيع البريمجات نشر برامج جافا على صفحات الويب طبقي الأمثلة prog69 و prog70 بالمعمل.

م٢ محاضرة ١٠
م/ ليلى بشير
مقدمة في البرمجة

Java Applets

البريمجات

وهي نوع من التطبيقات التي صممت خصيصا للإنترنت. حيث يقوم المطور (Developer) بإعداد هذا البريمج Applet ثم يتم استدعاءه من خلال ملف HTML. فيتم عرض هذا التطبيق من خلال متصفح يدعم لغة الجافا عندما يستدعي المستخدم صفحة الـ Html.

App1.java مثال لبريمج applet:

```
import java.applet.Applet;
import java.awt.Graphics;
```

أول سطرين في المثال هما لاستيراد اثنان من ال classes المستعملات في ال applet، و هما Applet و Graphics.

عملية استيراد ال Applet و ال Graphics تمكن البرنامج من الرجوع اليهم بدون ان يكون هناك تعريف لهم في بداية البرنامج.

ال java.applet. و ال java.awt. تخبر المترجم compiler في اي حزمة package يجب ان يبحث عن Applet و Graphics.

إن كل من java.applet و java.awt هما packages، و هما جزء من ال API و الذي يوجد في كل بيئة جافا.

ان ال (java.applet) package يحتوي على ال classes الضرورية لل applet. و ال (java.awt) package يحتوي على ال classes المستعملة في AWT و التي تستعمل لعمل الواجهة الرسومية لبرامج الجافا GUI.

السطر الثالث:

```
public class app1 extends Applet {
```

تعريف للصف app1 على أنه صف فرعي (subclass) يتوسع من ال class الفوقي Applet.

السطر الرابع:

```
public void paint(Graphics g) {
```

يجب على كل applet ان يعرف على الاقل واحد او اكثر من الmethods (init,start,paint).

لقد ادخلنا ال Graphics الى ال paint method و معنى هذا اننا سنحصل على صورة او نص على شاشة المتصفح.

ان أول argument من Graphics هو drawstring method و الذي سيقوم برسم النص على الشاشة، ال argument الثاني و الثالث هما (x,y) وهو موقع النص، اي الطرف الايسر السفلي من النص.

كيفية تشغيل ال applet:

١- نقوم بترجمة App1.java لنحصل على App1.class الخالي من الأخطاء.
٢- ندرج رابط ال applet في ملف HTML لكي نستطيع تشغيله على المتصفح كما بالسطر:

```
<APPLET CODE="App1.class" WIDTH=150 HEIGHT=25>
```

ان الكود الذي قمنا بإدراجه عمله ان يوجه المتصفح الى البحث عن الملف App1.class في نفس ال folder الذي يحتوي على صفحة ال html و يقوم بتشغيله.

البرنامج كامل
بالصفحة الثالثة ←

App1.java

```
import java.applet.Applet;
import java.awt.Graphics;
public class app1 extends Applet {
public void paint(Graphics g) {
g.drawString("Hello world!", 50, 25);
}
}
```

بعد ترجمة البرنامج نحصل على app1.class نقوم بإدراجه في كود صفحة الـ html لكي نستطيع تشغيله على المتصفح :

```
<HTML>
<HEAD>
<TITLE>A Simple Program</TITLE>
</HEAD>
<BODY>
Here is the output of my program:
<APPLET CODE="app1.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```