Java Nested Class

巢狀類別

本章重點

- 巢狀類別(Nested Class)簡介
- 內部類別(Inner Class)
- 靜態巢狀類別
- 區域類別(Local Class)
- 匿名類別(Anonymous Class)

巢狀類別(Nested Class)簡介

巢狀類別就是將類別定義在另一個類別中，所以巢狀類別是外圍類別(Enclosing Class)的成員

語法：

```
//外圍類別
[public] [final|abstract] class <類別識別子>
  [extends 父類別][implements 介面1 [,介面2]]{

    //巢狀類別
    [private|protected|public][static] [final|abstract] class <巢狀類別識別子>
      [extends 父類別]
      [implements 介面1 [,介面2]]{
        //巢狀類別中可以宣告屬性、方法、建構式、巢狀類別...
    }
}
```

內部類別(Inner Class)

Non-static的巢狀類別又被稱為內部類別(Inner Class)

檔案名稱：Enclosing1.java

```
public class Enclosing1 {
    private int var;
    //宣告內部類別
    public class Inner {
        public int getVar() {
            return var;
        }
    }
    public void testMyInner() {
        Inner i = new Inner();
        i.getVar();
    }
}
```

內部類別(Inner Class)(續)

因為內部類別是外圍類別 (Enclosing Class)的非靜態成員，所以

1. 必須先將外圍類別建立成為物件
2. 藉由這個物件才能將內部類別建立成為物件

內部類別中不得宣告static Member；但有一個例外: 內部類別中可以宣告static final變數(且須在宣告時給初值)

內部類別－範例

檔案名稱：Enclosing2.java

```
public class Enclosing2 {
    private int var;
    public class Inner {
        public int getVar() {
            return var;
        }
    }
}
```

內部類別－範例(續)

檔案名稱：TestEnclosing2.java

```
public class TestEnclosing2 {
 public static void main(String[] args) {

   Enclosing2 enclosing = new Enclosing2();
   Enclosing2.Inner inner =
              enclosing.new Inner();
   System.out.println(inner.getVar());
 }
}
```

內部類別(Inner Class)(續)

類別中與巢狀類別中可以有同名的Non-static成員變數

檔案名稱：Enclosing3.java

```
public class Enclosing3 {
 private int var;

 public class Inner {
  private int var;

  public void printData(int var) {
   var++;                    //參數
   this.var++;               //內部類別的成員
   Enclosing3.this.var++;  //外圍類別的成員
  }
 }
}
```

靜態巢狀類別(Static Nested Class)

靜態巢狀類別又提升回Top-level類別

靜態巢狀類別不得參考外圍類別的非靜態成員

```
public class EnclosingStaticNested {
 private static int var;
 public static class Nested { //靜態巢狀類別
  private static int var;
  public static void printData(int var) {
    var++;              //參數
    Nested.var++;
    EnclosingStaticNested.var++;
  }
 }
}
```

靜態巢狀類別－範例

檔案名稱：TestStaticNested.java

```
public class TestStaticNested{
  public static void main(String[] args) {
    //不需先建立外圍類別物件
    EnclosingStaticNested.Nested nested =
      new EnclosingStaticNested.Nested();
    nested.printData(1);
  }
}
```

區域類別(Local Class)

- 將類別定義在方法中即稱為區域類別(Local Class)
- 區域類別不得使用所在方法中的區域變數或參數，但可以使用所在方法中的final區域變數

區域類別－範例

```
public class Enclosing4 {
 private int var1 = 5;
 public Object makeTheLocal(int var) {
  final int finalVar = (int)(var*0.6);

  //這是定義在方法中的區域類別
  class Local {
    public String toString() {
      return (
       ":\n 外圍類別的var1為:" + var1 +
       //“\n 區域變數為:” + var + //編譯失敗
       "\n final區域變數為: " + finalVar);
    }
    public void test(){
      System.out.println("Test");
    }
  }
  Local local = new Local();
  local.test();
  return local;
}
```

區域類別－範例(續)

```
public class TestEnclosing4 {
  public static void main(String[] args) {
    Enclosing4 outer = new Enclosing4();
    Object obj = outer.makeTheLocal(82);
    System.out.println("這個物件" + obj);
    // obj.test();   //Uncomment會編譯失敗
  }
}
```

匿名類別(Anonymous Class)

- 在new Class(…)後面直接加上大括號{…}，並在大括號中宣告新的類別成員，這就產生了一個匿名類別(Anonymous Class)，並為這個匿名類別建立了物件
- 定義再方法中的匿名類別也可說是一種區域類別，與區域類別有相同的限制

匿名類別－範例

```
public class Enclosing5 {
 private int var1 = 5;
 public Object makeTheLocal(int var) {
  final int finalVar = (int)(var*0.6);

  //這是定義在方法中的區域類別
  Object obj = new Object {
    public String toString() {
      return (
      ":\n 外圍類別的var1為:" + var1 +
      //“\n 區域變數為:” + var + //編譯失敗
      "\n final區域變數為: " + finalVar);
    }
    public void test(){
      System.out.println("Test");
    }
  }
  obj.test();
  return obj;
}
```