第 13 章

套件（Packages）

13-1 程式的切割

- 當程式越來越大的時候,
  最簡單的整理方式就是讓每一個類別單獨儲存在一個原始檔中,
  並以所含的類別名稱為檔名,
  然後將程式所需的檔案全部放在同一個目錄中。

程式的切割

程式 UsingOtherClasses.java 只含主程式的類別

```
01 public class UsingOtherClasses {
02   public static void main(String[] argv) {
03     // 使用的類別都存放在同一資料夾的其它檔案中
04     Rectangle re = new Rectangle(1,3,5,7);
05     Circle    ci = new Circle(3,6,9);
06     Cylinder  cr = new Cylinder(2,4,6,8);
07
08     System.out.println(re.toString());
09     System.out.println(ci.toString());
10     System.out.println(cr.toString());
11   }
12 }
```

程式的切割

使用單獨檔案來儲存個別類別有以下好處：

- 管理類別更容易
  只要找到與類別同名的檔案，
  就可找到類別的原始程式。
- 編譯程式更簡單
  只要編譯包含有 main( ) 方法的原始檔，
  編譯器就會根據使用到的類別名稱，
  在同一個資料夾下找出同名的原始檔，
  並進行必要的編譯。

13-2 分享寫好的類別

- 有些情況下,撰寫的類別也可給其他人使用,可讓開發人員不需重複撰寫類似功能的類別。最簡單的方式,就是將該類別的 .java 或 .class 檔案複製給別人使用,不過此時對方的程式中,就不能有任何類別或介面和提供的類別同名。

分享寫好的類別

- Java 提供套件(Packages) 這種可將類別包裝起來的機制來解決上述問題。
  將類別包裝成套件時,
  就相當於替類別貼上一個標籤 (套件名稱), 而使用到套件中的類別時,
  即需指明套件名稱,
  因此可與程式中自己撰寫的同名類別區隔開來而不會混淆,也沒有必須重新命名的問題。

13-2-1 建立套件

- 要將類別包裝在套件中，
  必須在程式最開頭使用 package 敘述，
  標示出類別所屬的套件名稱。

建立套件

程式 Shape.java 將類別包裝在套件中

```
01 package flag; // 將類別包裝在 flag 套件中
02
03 public class Shape {     // 代表圖形原點的類別
04   protected double x,y; // 座標
05
06   public Shape(double x,double y) {
07     this.x = x;
08     this.y = y;
09   }
10
11   public String toString() {
12     return "圖形原點：(" + x + ", " + y + ")";
13   }
14 }
```

13-2-2 編譯包裝在套件中的類別

當類別使用 package 敘述指明所屬的套件後，在編譯類別時必須有額外的處理，可選擇以下兩種方式來整理及編譯這些類別：

- **自行建立套件資料夾：**

  在Java 中，同一套件的類別必須存放在與**套件名稱相同的資料夾中**。

- **由編譯器建立套件資料夾：**

  在指定的路徑下建立與套件同名的子資料夾，並將編譯結果存到該資料夾中。

```
javac -d C:\Ch13\13-2 Shape.java
```

若改用 *.java 則可一次編譯所有的檔案

13-2-3 使用套件中的類別

- 要使用套件中的類別時,必須以其完整名稱 來表示,格式為『套件名稱.類別名

程式 UsingPackage.java 使用套件中的類別

```
01 public class UsingPackage {
02   public static void main(String[] argv) {
03     flag.Rectangle re = new flag.Rectangle(1,3,5,7);
04     flag.Circle    ci = new flag.Circle(3,6,9);
05     flag.Cylinder  cr = new flag.Cylinder(2,4,6,8);
06
07     System.out.println(re.toString());
08     System.out.println(ci.toString());
09     System.out.println(cr.toString());
10   }
11 }
```

套件與程式檔在同一資料夾下

- 若套件和使用到套件的程式都放在同一資料夾下,仍以一般的方式編譯、執行即

```
C:\Example\Ch13\13-2>dir
......
2016/05/09  下午 07:50    <DIR>          flag
2016/05/09  下午 08:03               408 UsingPackage.java
......

C:\Example\Ch13\13-2>javac UsingPackage.java

C:\Example\Ch13\13-2>java UsingPackage
圖形原點：(1.0, 3.0)
圖形原點：(3.0, 6.0)
圖形原點：(2.0, 4.0)
```

套件與程式檔在同一資料夾下

- 如果一切無誤, 就可以正常執行, 否則就會出現編譯錯誤。

```
C:\Example\Ch13\13-2>javac UsingPackage.java
UsingPackage.java:4: cannot find symbol
    flag.Rectangle re = new flag.Rectangle(1,3,5,7);
       ^
  symbol   : class Rectangle
  location: package flag
UsingPackage.java:4: cannot find symbol
    flag.Rectangle re = new flag.Rectangle(1,3,5,7);
                            ^
  symbol   : class Rectangle
  location: package flag
2 errors
```

套件與程式在不同資料夾下

- 套件所在的資料夾通常並非使用該套件的程式所在的資料夾, 此時在編譯及執行程式時, 就必須告訴 Java 編譯器以及 Java 虛擬機器套件所在的位置。

```
C:\Example\Ch13\13-2>javac -cp C:\Allpackages UsingPackage.java

C:\Example\Ch13\13-2>java -cp .;C:\Allpackages UsingPackage
```

套件與程式在不同資料夾下

- -cp 選項意指 classpath, 也就是類別所在的路徑, Java 編譯器以及虛擬機器會到此選項指定的資料夾中尋找所需的類別檔案。
- 若需要列出多個路徑, 可用分號分隔之, 如此 Java 編譯器以及虛擬機器就會依照指定的順序, 依序到各個資料夾中找尋所需的類別, 如果程式使用到了位於不同資料夾的套件或是類別, 就必須如此指定。

套件與程式在不同資料夾下

- -cp 選項也可以使用全名寫成 -classpath, 兩者效用相同。若常會使用到某個資料夾下的套件或是類別, 也可設定環境變數 classpath, 就不需在編譯或是執行時額外指定 -cp 或是 -classpath 選項。

```
C:\Example\Ch13\13-2>set classpath=.;c:\Allpackages

C:\Example\Ch13\13-2>javac UsingPackage.java

C:\Example\Ch13\13-2>java UsingPackage
圖形原點：(1.0, 3.0)
圖形原點：(3.0, 6.0)
圖形原點：(2.0, 4.0)
```

13 - 3 子套件以及存取控制關係

- 當程式越來越大時, 單一套件中可能會包含多個類別。要做到這一點, 只要將 Sort 類別以及伴隨的 ICanCompare 介面分別加上 package 敘述再存檔編譯即可：

程式 ICanCompare.java 將介面加入套件中

```
01 package flag;
02
03 public interface ICanCompare {
04   int compare(ICanCompare i); // 進行比較
05 }
```

子套件以及存取控制關係

程式 Sort.java 將 Sort 類別加入 flag 套件中

```
01 package flag;
02
03 public class Sort { // 提供排序功能的類別
04   public static void bubbleSort(ICanCompare[] objs) {
05     // 氣泡排序法
06     for(int i = objs.length - 1;i > 0;i--) {
07       for(int j = 0;j < i;j++) {
08         if(objs[j].compare(objs[j + 1]) < 0) {
09           ICanCompare temp = objs[j];
10           objs[j] = objs[j + 1];
11           objs[j + 1] = temp;
12         }
13       }
14     }
15   }
16 }
```

子套件以及存取控制關係

程式 Sorting.java 使用套件中的 Sort 類別進行排序

```
01 abstract class Land implements flag.ICanCompare { // 父類別
02   abstract double area(); // 計算面積
03   public int compare(flag.ICanCompare i) { // 實作 compare() 方法
04     Land l = (Land) i;
05     return (int)(this.area() - l.area());  // 依據面積比較大小
06   }
07 }
08
09 class Circle extends Land { // 圓形的土地
10   int r; // 半徑（單位：公尺）
11
12   Circle(int  r) { // 建構方法
13     this.r = r;
14   }
```

子套件以及存取控制關係

```
15
16    double area() { // 多重定義的版本
17      return 2 * 3.14 * r * r;
18    }
19
20    public String toString() {
21      return "半徑：" + r + ",面積：" + area() + "的圓";
22    }
23  }
24
25  class Square extends Land { // 正方形的土地
26    int side; // 邊長（單位：公尺）
27
28    Square(int  side) { // 建構方法
29      this.side = side;
30    }
31
```

子套件以及存取控制關係

```
32    double area() { // 多重定義的版本
33      return side * side;
34    }
35
36    public String toString() {
37      return "邊長：" + side + ",面積：" + area() + "的正方形";
38    }
39  }
40
41  public class Sorting {
42
43    public static void main(String[] argv) {
44      Land[] Lands = {
45              new Circle(5),
46              new Square(3),
47              new Square(2),
```

子套件以及存取控制關係

```
48                  new Circle(4)
49          };
50
51          for(Land l : Lands) {
52                  System.out.println(l);
53          }
54
55          flag.Sort.bubbleSort(Lands);
56          System.out.println("排序後...");
57
58          for(Land l : Lands) {
59                  System.out.println(l);
60          }
61      }
62  }
```

13-3-1 在套件中建立子套件

- 當套件中的類別、介面愈來愈多時,
  為了分類管理套件,
  可進一步在現有套件中建立子套件,
  集合相關類別, 歸於某一子套件,
  而這個子套件則和其他的類別或是子套件同屬於一個上層套件中。

在套件中建立子套件

程式 Shape.java 將類別包裝在套件中

```
01 package flag.math;          // 將類別包裝在 flag.math 子套件中
02
03 public class Shape {        // 代表圖形原點的類別
04   protected double x,y;     // 座標
05
06   public Shape(double x,double y) {
07     this.x = x;
08     this.y = y;
09   }
10
11   public String toString() {
12     return "圖形原點：(" + x + ", " + y + ")";
13   }
14 }
```

在套件中建立子套件

- 子套件類別的 .class 檔也一樣要放在與子套件同名的資料夾下。
- 由於牽涉多層資料夾, 因此在編譯時建議使用前面介紹過的 -d 選項, 讓 Java 編譯器依據套件的結構建立對應的資料夾。

```
javac -d . Shape.java
          ↑
          └── 小數點代表目前的路徑
```

在套件中建立子套件

- 將 Shape 等相關類別放入子套件後, 參考這些類別時所用的『完整名稱』就必須包含子套件名稱。

程式 UsingSubPackage.java 測試子套件

```
01 public class UsingSubPackage {
02   public static void main(String[] argv) {
03     flag.math.Rectangle re = new flag.math.Rectangle(1,3,5,7);
04     flag.math.Circle    ci = new flag.math.Circle(3,6,9);
05     flag.math.Cylinder  cy = new flag.math.Cylinder(2,4,6,8);
06
07     System.out.println(re.toString());
08     System.out.println(ci.toString());
09     System.out.println(cy.toString());
10   }
11 }
```

13-3-2 使用 import 敘述

- 如果已確定自己程式中的類別名稱都不會與套件中的類別名稱相衝突，那麼就可以利用 import 敘述，改善程式總是出現一長串名稱的情形。
- 使用 import 敘述的方式可分為兩種：
  匯入單一類別名稱、
  或匯入套件中所有的類別名稱。

匯入指定套件中的單一類別

- 直接匯入套件中的單一類別

程式 OnlyImportClass.java 匯入單一類別的名稱

```
01 import flag.math.Rectangle;
02
03 public class OnlyImportClass {
04   public static void main(String[] argv) {
05     Rectangle r = new Rectangle(1,3,5,7);
06     flag.math.Circle c = new flag.math.Circle(3,6,9);
07
08     System.out.println(r.toString());
09     System.out.println(c.toString());
10   }
11 }
```

匯入指定套件中所有的類別名稱

- 如果程式同時要用到套件中的多個類別，可用萬用字元 *，表示要匯入指定套件中所有的類別名稱。

程式 ImportPackage.java 匯入套件中的所有類別名稱

```
01 import flag.math.*;
02
03 public class ImportPackage {
04   public static void main(String[] argv) {
05     Rectangle r = new Rectangle(1,3,5,7);
06     Cylinder c = new Cylinder(2,4,6,8);
07
08     System.out.println(r.toString());
09     System.out.println(c.toString());
10   }
11 }
```

匯入指定套件中所有的類別名稱

- 萬用字元只代表該套件中的所有類別，而不包含其下的子套件。
  若將 ImportPackage.java 的第一行改成：

```
import flag.*;
```

- 表示只匯入 flag 中的類別或介面，但未匯入 flag.math 子套件中的類別或介面，因此編譯到第 5、6 行時就會發生錯誤。

13-3-3 套件與存取控制的關係

嚴格程度：高 ↑ 低

字符	說明
private	只有在成員所屬的類別中才能存取此成員
預設控制（不加任何存取控制字符）	只有在同一套件的類別中才能存取此成員
protected	只有在子類別中或是同一套件的類別中才能存取此成員
public	任何類別都可以存取此成員

套件與存取控制的關係

- 由於存取控制字符可以加諸在成員、方法或是類別上，因此必須特別小心。

程式 DefaultClass.java 存取控制抵觸

```
01 package flag;
02
03 class DefaultClass { // 預設只有同一套件中的類別才能使用
04   public static int i = 10;
05 }
```

套件與存取控制的關係

- DefaultClass 並未加上任何存取控制字符, 因此只有同一套件的類別才能使用, 即使成員變數 i 是 public, 套件外的程式也無法存取之：

程式 TestDefault.java 無法存取 public 成員

```
01 public class TestDefault {
02   public static void main(String[] argv) {
03     System.out.println(flag.DefaultClass.i);
04   }
05 }
```

套件與存取控制的關係

執行結果

```
TestDefault.java:3: flag.DefaultClass is not public in flag;
cannot be accessed from outside package
    System.out.println(flag.DefaultClass.i);
                               ^
1 error
```

套件與存取控制的關係

基本的使用規範如下：

- 如果類別是要給所有的類別使用，請標示為 public，否則就不要標示，那麼就只有同套件的類別可以存取。
- 如果類別的成員只給子類別或同一套件的其他類別使用，請將之標示為 protected。

套件與存取控制的關係

- 如果類別的成員只給同一套件的其他類別使用，就不要標示存取控制字符，採用預設存取控制。
- 如果成員類別的只給同類別中的其他成員使用，就請標示為 private。

13-3-4 預設套件

- 對所有未標示所屬套件的類別，
  Java都會將之視為是預設套件
  (Default Package)中的一員，
  這個預設套件相當於一個沒有名字的套件
- 在 UsingOtherClasses.java 程式中
  所用到同一資料夾中的其它類別，
  就都屬於同一個預設套件，
  所以程式可正常使用這些類別

13-4 綜合演練

- 13-4-1 加入新的類別到 flag 套件中

程式 ICanCompare.java 包在子套件中的介面

```
01 package flag.utility;
02
03 public interface ICanCompare {
04   int compare(ICanCompare i); // 進行比較
05 }
```

加入新的類別到 flag 套件中

程式 Sort.java 包在子套件中的類別

```
01 package flag.utility;
02
03 public class Sort { // 提供排序功能的類別
04   public static void bubbleSort(ICanCompare[] objs) {
05     // 氣泡排序法
06     for(int i = objs.length - 1;i > 0;i--) {
07       for(int j = 0;j < i;j++) {
08         if(objs[j].compare(objs[j + 1]) < 0) {
09           ICanCompare temp = objs[j];
10           objs[j] = objs[j + 1];
11           objs[j + 1] = temp;
12         }
13       }
14     }
15   }
16 }
```

13-4-2 Java 標準類別庫

- Java 預設就提供了許多的套件，可以幫助您處理多種工作
- java.lang 則提供了許多與 Java 語言本身有關的類別，像是對應於基礎型別的 Integer 等類別，就屬於此一套件。不過 Java 預設就會匯入 java.lang.*，所以我們不需在程式中自行匯入。
- 有關 Java 本身提供的這些套件，統稱為 Java 標準類別庫。

13-4-3 套件的命名

1. 每家公司 以其在網際網路上的網域名稱相反的順序為最上層套件的名稱。
2. 可以再依據部門或是工作單位名稱，在最上層套件中建立適當的子套件，以放置該部門所撰寫的所有類別，避免同一公司、不同單位的人使用相同的類別名稱
3. 在最上層套件中，再依據類別的用途建立適當的子套件