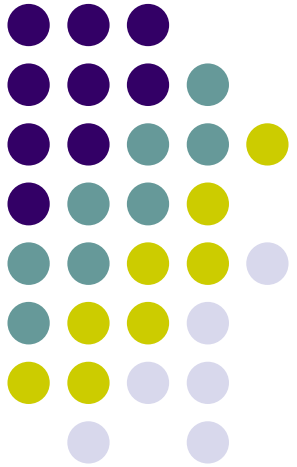
第十二章
大型程式的發展與常用的類別庫

學習如何分割檔案

認識類別庫以及取用類別庫裡的類別

建構package的階層關係

學習Java裡常用的類別庫



分割檔案的實作 (1/2)

- 以CCircle類別爲例，說明分割檔案的實作

1. 依序建立兩個類別檔案，並置於同一個資料夾內：

CCircle.java

```
01  // CCircle.java, 本檔案置於 c:\Java\pack1 資料夾內
02  class CCircle   // 定義類別 CCircle
03  {
04     public void show()
05     {
06        System.out.println("show() method called");
07     }
08  }
```

app12_1.java

```
01  // app12_1.java, 本檔案置於 c:\Java\pack1 資料夾內
02  public class app12_1
03  {
04     public static void main(String args[])
05     {
06        CCircle cir=new CCircle();
07        cir.show();
08     }
09  }
```

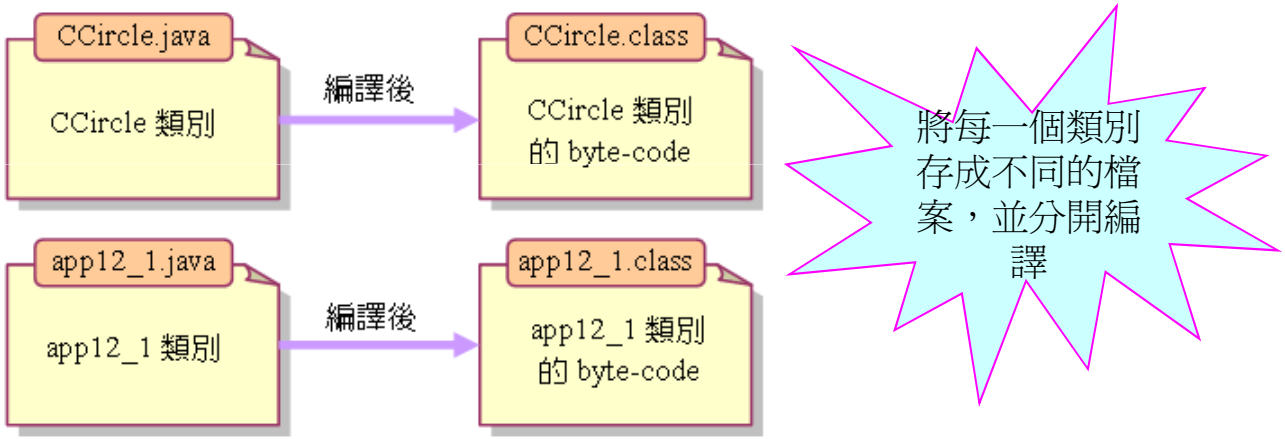
分割檔案的實作 (2/2)

2. 分別以下列的指令編譯CCircle.java與 app12_1.java：

```
C:\Java\pack1>javac CCircle.java

C:\Java\pack1>javac app12_1.java
```

編譯後會產生CCircle.class與app12_1.class兩個檔案：



3. 接下來鍵入：

```
C:\Java\pack1>java app12_1
```

即可執行此一程式，並得到下面的結果：

```
/* app12_1 OUTPUT-----

show() method called
----------------------*/
```

package的基本概念

- 把package想像成類別庫，是專門用來收納類別的地方
- 不同的package內可以擁有名稱相同的類別
- package使用格式如下：

package的宣告

```
package package名稱;
```

package的使用範例一 (1/3)

- app12_2是package使用的範例

app12_2.java

```
01  // app12_2, package 的使用(一),此檔案置於 pack2 資料夾內
02  package pack2;   // 宣告以下程式碼所定義的類別均納入 package pack2 中
03  class CCircle    // CCircle 類別已納入 package pack2 中
04  {
05     public void show()
06     {
07        System.out.println("show() method called");
08     }
09  }
10  public class app12_2    // app12_2 類別也納入 package pack2 中
11  {
12     public static void main(String args[])
13     {
14        CCircle cir=new CCircle();
15        cir.show();
16     }
17  }
```

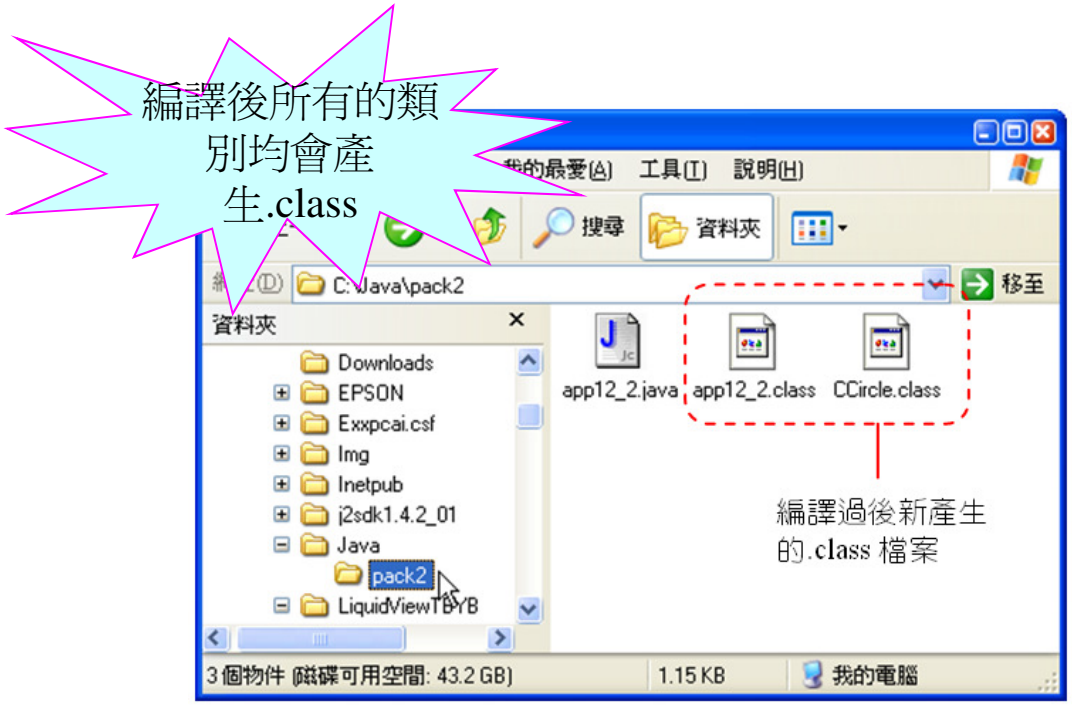
存放app12_2.java的資料夾名稱必須與package名稱相同

package的使用範例一 (2/3)

- 編譯app12_2.java：

```
C:\Java>javac pack2\app12_2.java
```

  - 此時在pack2資料夾內會產生兩個檔案：



- 執行app12_2.java：

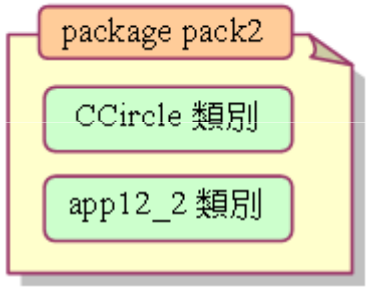
```
C:\Java>java pack2.app12_2
```

- 執行結果：

```
/* app12_2 OUTPUT-----
show() method called
---------------------*/
```

package的使用範例一 (3/3)

- app12_2.java是把兩個類別（CCircle和app12_2）放在同一個package內

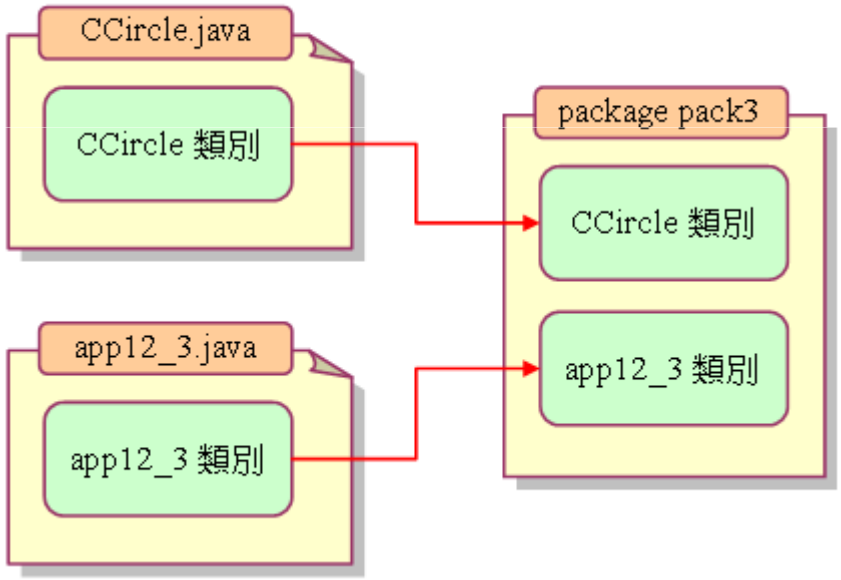


- 在原始檔案中若沒有指明package，則會被視爲

「沒有名稱的package」

package的使用範例二 (1/3)

- 儲存於不同的檔案中的類別，也可以隸屬於同一個package：



package的使用範例二 (2/3)

- 下面是將不同檔案的類別納入同一個package的範例：

CCircle.java

```
01  // CCircle.java, package 的使用(二),此檔案置於 pack3 資料夾內
02  package pack3;   // 宣告下面所定義的類別均納入 package pack3 中
03  class CCircle    // 將 CCircle 類別納入 package pack3 中
04  {
05     public void show()
06     {
07        System.out.println("show() method called");
08     }
09  }
```

app12_3.java

```
01  // app12_3.java, package 的使用(二),此檔案置於 pack3 資料夾內
02  package pack3;        // 宣告下面所定義的類別均納入 package pack3 中
03  public class app12_3 // 將 app12_3 類別納入 package pack3 中
04  {
05     public static void main(String args[])
06     {
07        CCircle cir=new CCircle();
08        cir.show();
09     }
10  }
```

package的使用範例二 (3/3)

- 請依下面的指令來編譯它們：

```
C:\Java>javac pack3\app12_3.java
C:\Java>javac pack3\CCircle.java
```

- 鍵入下面的指令來執行它：

```
C:\Java>java pack3.app12_3
```

- 執行結果：

```
/* app12_3 OUTPUT----
show() method called
---------------------*/
```

只要在每個檔案前面加上package名稱，便可將它們歸屬於同一個package

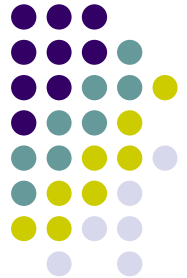
類別在不同package裡的修改

- 如果類別分別屬於不同的package時，在某個類別要存取到其它類別的成員時，必須做下列的修改：

  (1) 若某個類別需要被存取時，此類別必須宣告成public

  (2) 要存取不同package內某個public類別的成員時，必須的指明「被存取package的名稱.類別名稱」

簡單的範例 (1/3)



- 下面的範例說明如何存取在不同package裡的類別：

CCircle.java

```
01  // CCircle.java, package 的使用(三),此檔案存放在 pack4b 資料夾內
02  package pack4b;
03  public class CCircle      // 將 CCircle 類別納入 package pack4b 當中
04  {
05     public void show()
06     {
07        System.out.println("show() method called");
08     }
09  }
```

```
01  // app12_4.java, package 的使用(三),此檔案存放在 pack4a 資料夾內
02  package pack4a;
03  public class app12_4 // 將 app12_4 類別納入 package pack4a 當中
04  {
05     public static void main(String args[])
06     {
07        pack4b.CCircle cir=new pack4b.CCircle();
08        cir.show();
09     }
10  }
```

簡單的範例 (2/3)

- 請依下面的指令編譯：

```
C:\Java>javac pack4a\app12_4.java
C:\Java>javac pack4b\CCircle.java
```

  - 鍵入下面的指令來執行它：

```
C:\Java>java pack4a.app12_4
```

  - 執行結果：

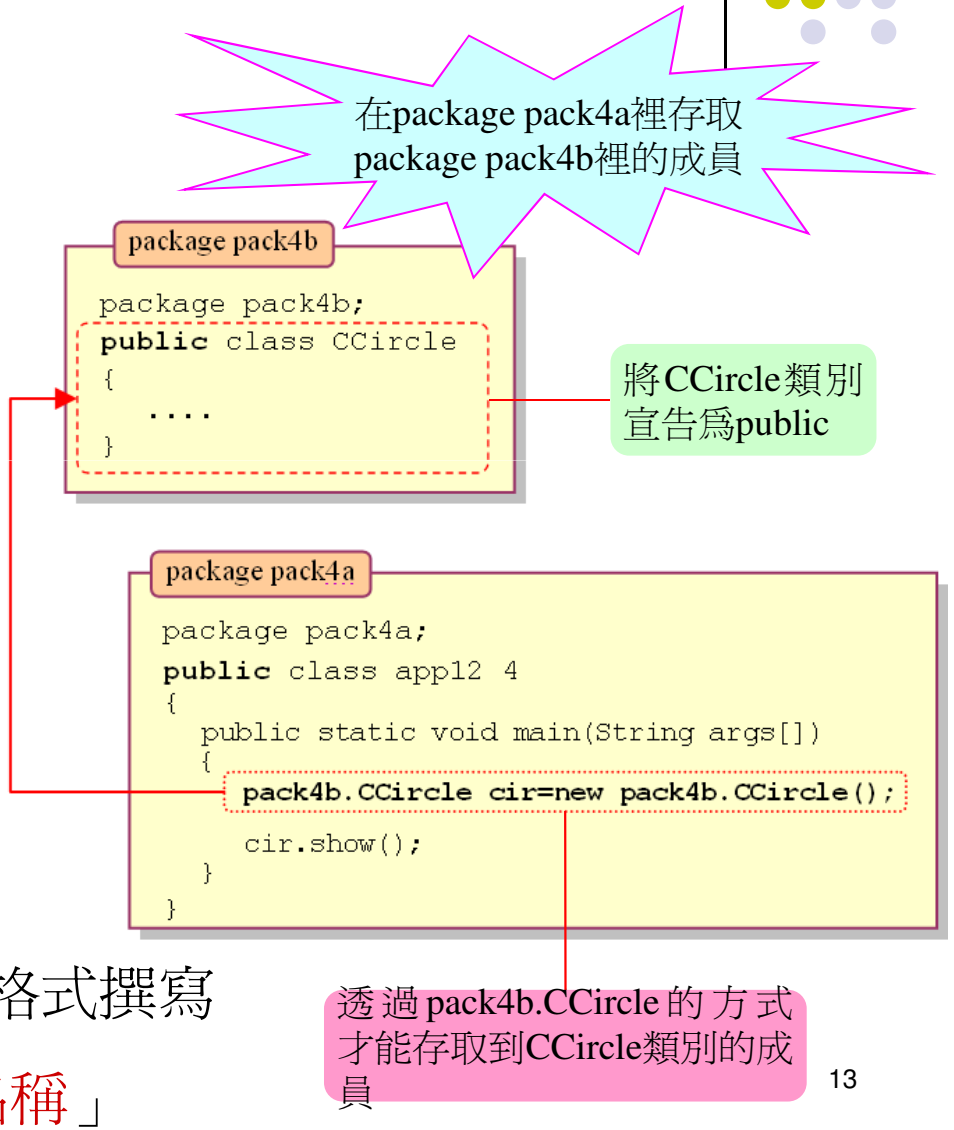
```
/* app12_4 OUTPUT----
show() method called
---------------------*/
```

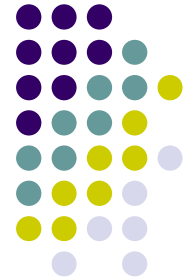
- 我們必須準備好兩個步驟：

(1) 把CCircle類別公開出來

(2) app12_4.java的第7行須以下面格式撰寫

「被存取package的名稱.類別名稱」





簡單的範例 (3/3)

- 存取不同package裡的類別時，被存取的類別必須宣告成public
- 拿掉CCircle.java第3行public時，編譯app12_4.java時會出現錯誤訊息：

```
pack4a\app12_4.java:7: pack4b.CCircle is not public in pack4b; cannot be accessed
from outside package
```

- public的注意事項
  - 一個Java檔案內，只能有一個類別宣告爲public
  - 檔案名稱必須與宣告成public的類別名稱相同
  - package中若有多個類別，可以將這些類別宣告成public再分別存檔

修飾子的角色

- 下面列出修飾子所扮演的角色 ：

表 12.3.1 類別與介面所使用的修飾子

修飾子	說　　明
沒有修飾子	只能讓同一個 package 裡的類別來存取
public	其它 package 裡的類別也可以存取此一類別裡的成員

表 12.3.2 成員與建構元所使用的修飾子

修飾子	說　　明
沒有修飾子	成員或建構元只能被同一個 package 內的程式所存取
public	如果所屬的類別也宣告成 public，則成員或建構元可被不同 package 內所有的類別所存取。若所屬類別不是宣告成 public，則成員或建構元只能被同一個 package 內的程式所存取
private	成員或建構元只在同一個類別內存取
protected	成員或建構元只能被位於同一 package 內的類別，以及它的子類別來存取

匯入packages

- 存取存放在不同package裡的類別，可以透過

  被存取的package名稱.類別名稱

  的語法

參考app12_4.java
的第7行

被存取的 package 名稱

```
pack4b. CCircle cir = new pack4b.CCircle();
```

類別名稱

- 匯入package裡某個類別的指令：

匯入package裡的某個類別

```
import package名稱.類別名稱;
```

- 透過import指令可將某個package內的特定類別匯入，「被存取的package名稱」的指定方式可省略

import指令

app12_5.java

```
01  // app12_5.java, package 的使用(四),此檔案置於 pack5a 資料夾內
02  package pack5a;
03  import pack5b.CCircle;   // 載入 pack5b package 裡的 CCircle 類別
04
05  public class app12_5
06  {
07     public static void main(String args[])
08     {
09        CCircle cir=new CCircle();    // 不用再寫 package 的名稱
10        cir.show();
11     }
12  }
```

import指令的範例

CCircle.java

```
01  // CCircle.java, package 的使用(四),此檔案置於 pack5b 資料夾內
02  package pack5b;
03  public class CCircle  // 將 CCircle 類別納入 package pack5b 當中
04  {
05     public void show()
06     {
07        System.out.println("show() method called");
08     }
09  }
```

CCircle類別要宣告成public，否則無法讓存放在pack5a的程式讀取

- 請依下面的輸入來編譯與執行：

```
C:\Java>javac pack5a\app12_5.java
C:\Java>javac pack5b\CCircle.java
C:\Java>java pack5a.app12_5

/* app12_5 OUTPUT-----
show() method called
----------------------*/
```

package的階層關係 (1/2)

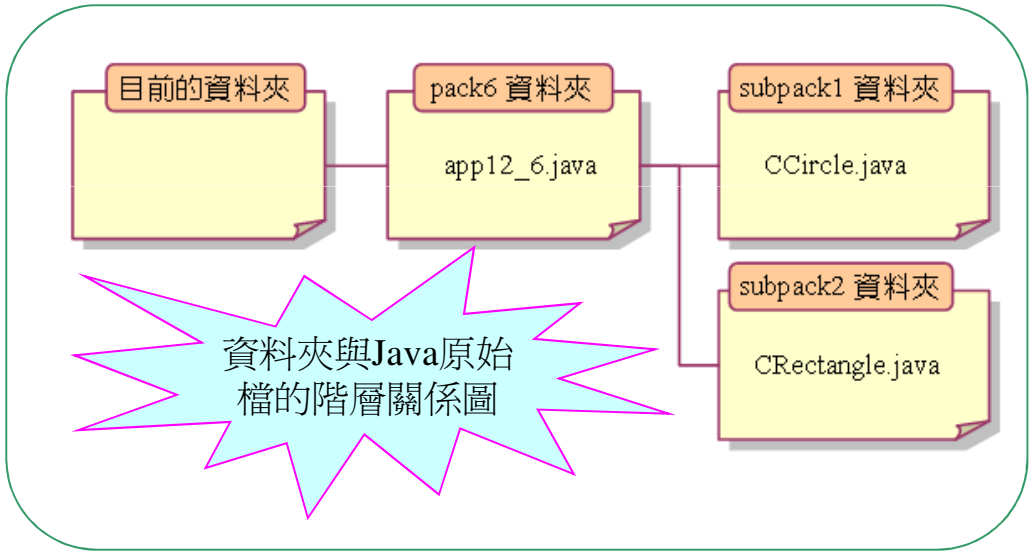
- 把packages劃分爲階層的關係，程式碼更容易維護
- package依功能劃分，可再分爲幾個「子package」
- 宣告sub-package的語法：

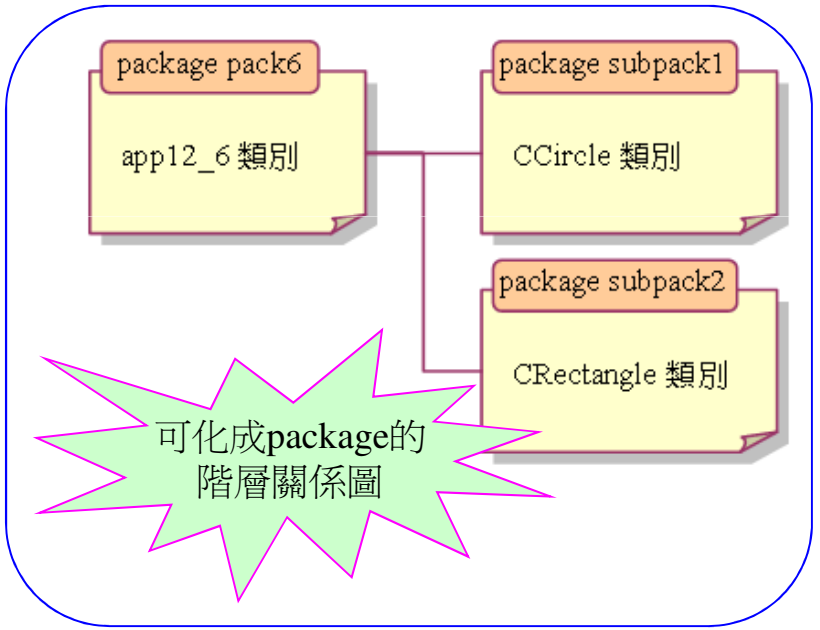
宣告sub-package

```
package package名稱.sub-package名稱;
```

package的階層關係 (2/2)

- package的階層關係示意圖：





建構package的階層 (1/2)

- 程式實作的部份 ：

CCircle.java

```
01  // CCircle.java, 此檔案置於 pack6\subpack1 資料夾內
02  package pack6.subpack1;   // 將 CCircle 類別納入 pack6.subpack1 中
03  public class CCircle
04  {
05     public void show()
06     {
07        System.out.println("show() method of class CCircle called");
08     }
09  }
```

CRectangle.java

```
01  // CRectangle.java, 此檔案置於 pack6\subpack2 資料夾內
02  package pack6.subpack2;   // 將 CRectangle 類別納入 pack6.subpack2 中
03  public class CRectangle
04  {
05     public void show()
06     {
07        System.out.println("show() method of class CRectangle called");
08     }
09  }
```

建構package的階層 (2/2)

app12_6.java

```
01  // app12_6.java,此檔案置於 pack6 資料夾內
02  package pack6;  // 將 app12_6 類別納入 package pack6 當中
03  import pack6.subpack1.CCircle;    // 載入 pack6.subpack1 裡的 CCircle 類別
04  import pack6.subpack2.CRectangle; // 載入 pack6.subpack2 裡的 CRectangle 類別
05
06  public class app12_6
07  {
08     public static void main(String args[])
09     {
10        CCircle cir=new CCircle();
11        CRectangle rect=new CRectangle();
12        cir.show();
13        rect.show();
14     }
15  }
```

```
/* app12_6 OUTPUT-----------------------
show() method in class CCircle called
show() method in class CRectangle called
----------------------------------------*/
```

- 分別編譯檔案後，再執行app12_6.class：

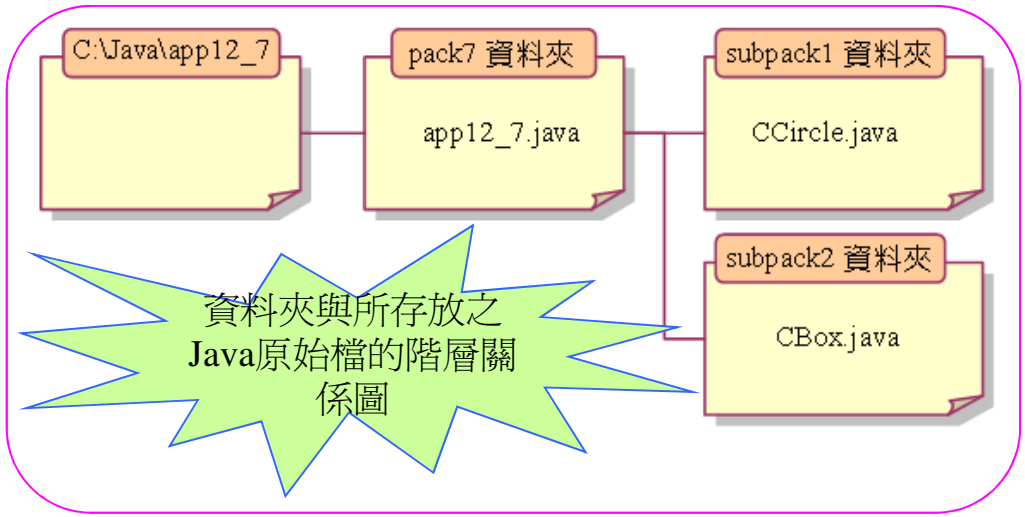
```
C:\Java>javac pack6\subpack1\CCircle.java
C:\Java>javac pack6\subpack2\CRectangle.java
C:\Java>javac pack6\app12_6.java
C:\Java>java pack6.app12_6
```

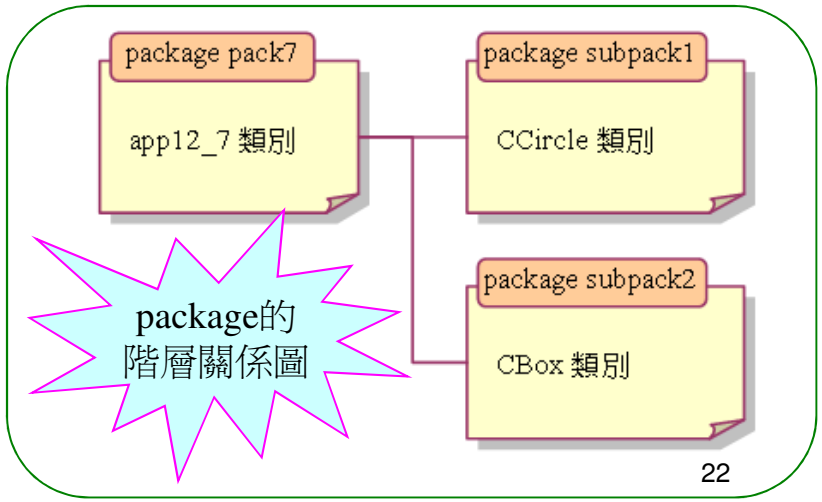
分別編譯每一個 Java 原始檔案

執行 app12_6

JCreator的Project

- JCreator的Project，可以協助我們
  - 管理類別
  - 介面
  - 程式檔案
  - 發展大型程式
- package的階層關係圖示意圖：



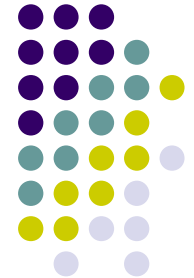


使用JCreator的Project (1/4)

- 本例的程式碼：

app12_7.java

```
01  // app12_7.java, 此檔案置於 pack7 資料夾內
02  package pack7;                    // 將 app12_7 類別納入 package pack7 當中
03  import pack7.subpack1.CCircle;    // 載入 pack7.subpack1 裡的 CCircle 類別
04  import pack7.subpack2.CBox;       // 載入 pack7.subpack2 裡的 CBox 類別
05
06  public class app12_7
07  {
08     public static void main(String args[])
09     {
10        CCircle cir=new CCircle(2);
11        CBox box=new CBox(2,3,4);
12        cir.show();
13        box.show();
14     }
15  }
```



使用JCreator的Project (2/4)

CCircle.java

```
01  // CCircle.java, 此檔案置於 pack7\subpack1 資料夾內
02  package pack7.subpack1;   // 將 CCircle 類別納入 pack7.subpack1 中
03  public class CCircle
04  {
05     final static double PI=3.14;
06     private double radius;
07
08     public CCircle(double r)
09     {
10        radius=r;
11     }
12     public void show()
13     {
14        System.out.print("radius="+radius);
15        System.out.println(", area="+PI*radius*radius);
16     }
17  }
```

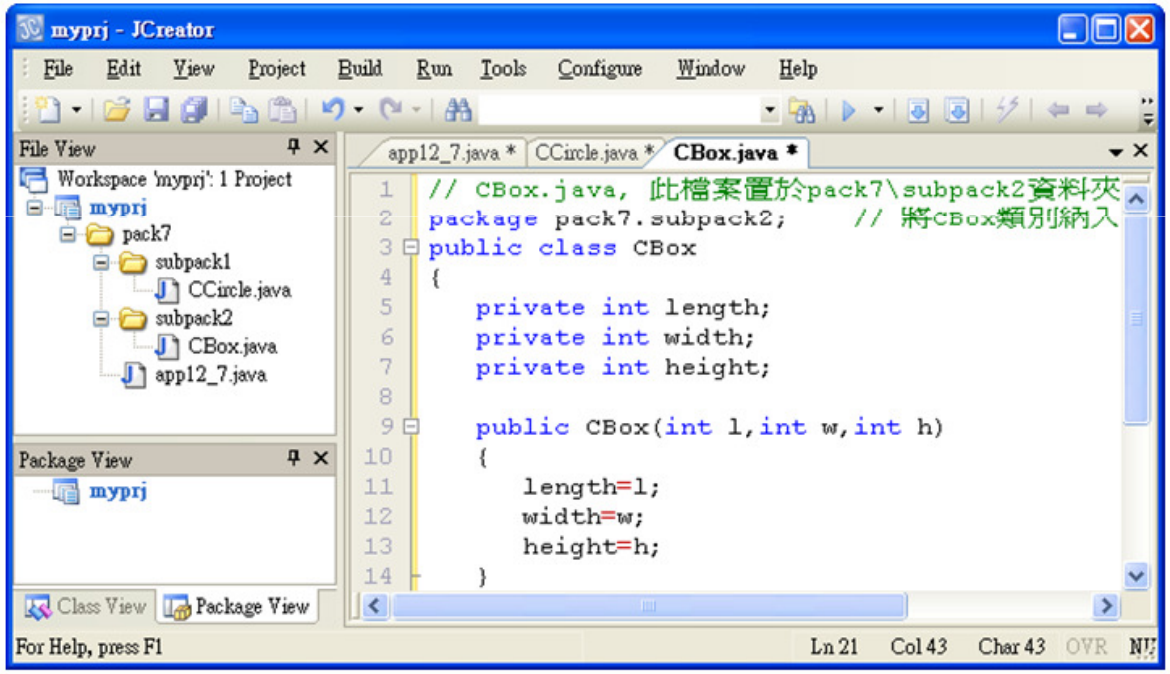
使用JCreator的Project (3/4)

CBox.java

```
01  // CBox.java, 此檔案置於 pack7\subpack2 資料夾內
02  package pack7.subpack2;        // 將 CBox 類別納入 pack7.subpack2 中
03  public class CBox
04  {
05     private int length;
06     private int width;
07     private int height;
08
09     public CBox(int l,int w,int h)
10     {
11        length=l;
12        width=w;
13        height=h;
14     }
15     public void show()
16     {
17        int vol=length*width*height;
18        System.out.print("length="+length);
19        System.out.print(", width="+width);
20        System.out.print(", height="+height);
21        System.out.println(", volume="+vol);
22     }
23  }
```

使用JCreator的Project (4/4)

- 將所有的程式加入myprj之後的視窗如下圖：




```
/* app12_7 OUTPUT------------------------

radius=2.0, area=12.56
length=2, width=3, height=4, volume=24
----------------------------------------*/
```

加入現有的程式到Project (1/3)

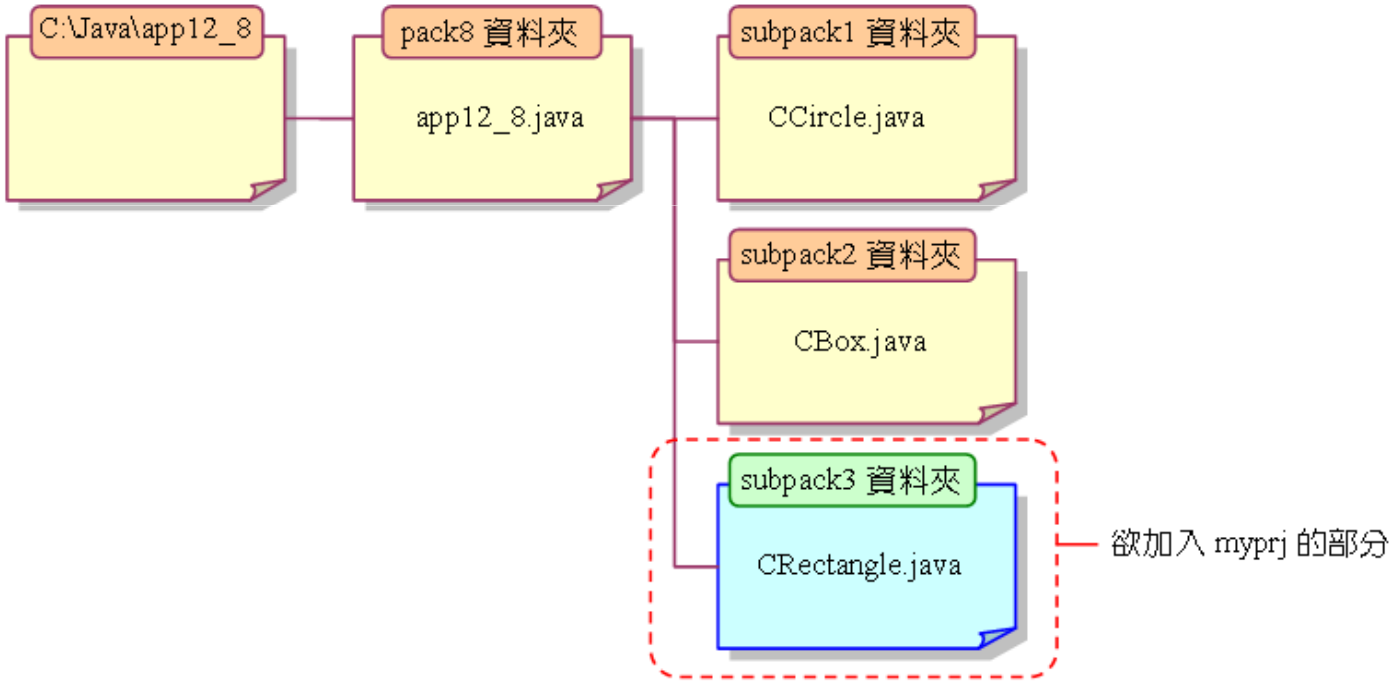
- 以app12_7爲延伸，將CRectangle.java加入myprj中：

CRectangle.java

```
01  // CRectangle.java, 此檔案置於 pack8\subpack3 資料夾內
02  package pack8.subpack3; // 將 CRectangle 類別納入 pack8.subpack3 中
03  public class CRectangle
04  {
05     private int length;
06     private int width;
07
08     public CRectangle(int l,int w)
09     {
10        length=l;
11        width=w;
12     }
13     public void show()
14     {
15        System.out.print("length="+length);
16        System.out.print(", width="+width);
17        System.out.println(", area="+length*width);
18     }
19  }
```

加入現有的程式到Project (2/3)

- 下圖爲app12_8資料夾及其階層關係圖：



加入現有的程式到Project (3/3)

app12_8.java

```
01 // app12_8.java,此檔案置於 pack8 資料夾內
02 package pack8; // 將 app12_8 類別納入 package pack8 當中
03 import pack8.subpack1.CCircle;    // 載入 pack8.subpack1 裡的 CCircle 類別
04 import pack8.subpack2.CBox;       // 載入 pack8.subpack2 裡的 CBox 類別
05 import pack8.subpack3.CRectangle; // 載入 pack8.subpack3 裡的 CRectangle 類別
06
07  public class app12_8
08  {
09     public static void main(String args[])
10     {
11        CCircle cir=new CCircle(2);
12        CBox box=new CBox(2,3,4);
13        CRectangle rec=new CRectangle(5,7);
14        cir.show();
15        box.show();
16        rec.show();
17     }
18  }
```

```
/* app12_8 OUTPUT-----------------------

radius=2.0, area=12.56
length=2, width=3, height=4, volume=24
length=5, width=7, area=35
----------------------------------------*/
```

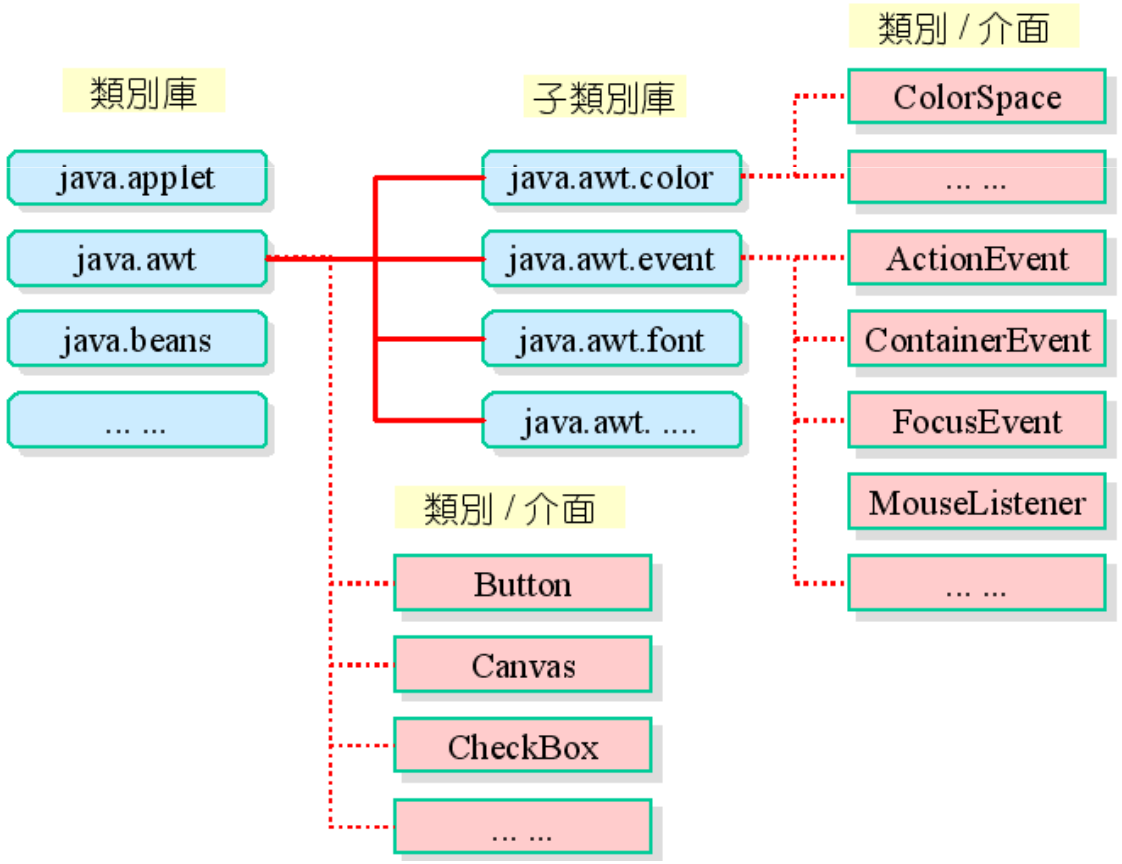
類別庫

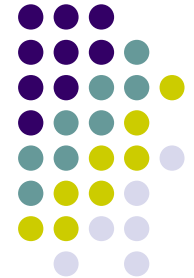
- package 可譯爲「類別庫」
- Java 的 package 是用來放置類別與介面的地方

查看Java所提供的類別庫

類別庫的層級關係圖

- 下面的簡圖顯示類別庫、子類別庫、類別與介面之間的層級關係



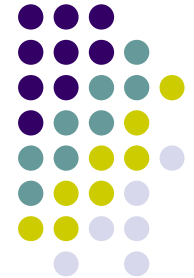


常用的類別庫

- 下表列出Java常用的類別庫：

表 12.6.1　Java 常用的類別庫

類別庫名稱	所包含類別之主要功能
java.applet	與 applet 相關的類別，applet 是嵌在網頁裡的小程式，用來執行特定的功能
java.awt	與 Java 早期視窗元件設計有關的類別
java.awt.event	與事件（event）觸發相關的類別
java.lang	Java 最基本的類別，此類別會自動載入
java.io	與輸入/輸出相關的類別
java.net	與網路元件與連線相關的類別
java.util	Java utility 相關的類別，如 Array、Vector 等



類別庫的使用

- 匯入java.awt類別庫裡的Button類別之語法：

```
import java.awt.Button;          // 匯入 java.awt 類別庫裡的 Button 類別
```

- 匯入多個類別，可用下面的語法：

```
import java.awt.Button;          // 匯入 java.awt 類別庫裡的 Button 類別
import java.awt.Canvas;          // 匯入 java.awt 類別庫裡的 Canvas 類別
```

- 匯入某個類別庫裡的所有類別時，可透過萬用字元「*」：

```
import java.awt.*;               // 匯入 java.awt 類別庫裡的所有類別
```

- 匯入java.awt.event下的所有類別，可用下列語法：

```
import java.awt.event.*;    // 匯入 java.awt.event 子類別庫裡的所有類別
```

建立字串物件

- String類別是放置在java.lang類別庫內
- java.lang類別庫裡所有的類別會自動載入
- 建立String物件的範例：

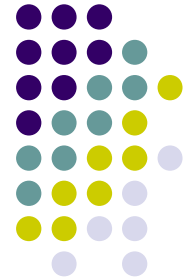
```
String str = "abc";                    // 宣告 str 變數，並設值為 abc
```

  - 也可利用字元陣列來產生字串：

```
char data[] = {'a', 'b', 'c'};         // 設定 data 為字元 a,b,c 所組成的陣列
String str = new String(data);         // 利用 String()建構元來產生字串
```

  - 另一種是直接利用String建構元來建立字串：

```
String str = new String("abc");        // 利用建構元來建立字串
```

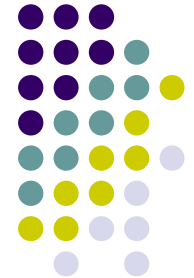


建立字串物件

- 下表列出第二種與第三種所使用的建構元之格式：

表 12.6.2　String 類別建構元的格式

建構元格式	主要功能
String()	沒有引數的 String() 建構元
String(byte[] bytes)	以 byte 陣列建立字串
String(byte[] bytes, int offset, int length)	取出 byte 陣列裡，從陣列的第 offset 位置開始，長度為 length 來建立字串
String(char[] value)	利用字元陣列來產生字串物件
String(char[] value, int offset, int count)	取出字元陣列裡，從陣列的第 offset 位置開始，長度為 count 來建立字串
String(String original)	利用原始字串（original string）產生字串物件



字串類別所提供的method

- 下表列出常用的mehtod：

表 12.6.3　String 類別常用的 method

method	主要功能
byte[] getBytes()	將字串轉換成 byte 型態的陣列
char charAt(int index)	取得 index 位置的字元
boolean equals(String str)	測試字串是否與 str 相同
int indexOf(char ch)	根據字元 ch 找出第一個在字串出現的位置
int length()	取得字串的長度
String substring(int index)	取出 index 之後的子字串
String substring(int ind1, int ind2)	取出位於 ind1 和 ind2 之間的字串
boolean startsWith(String prefix)	測試字串是否以 prefix 字串為開頭
String toLowerCase()	將字串轉換成小寫
String toUpperCase()	將字串轉換成大寫

String類別的使用範例

- 下面的範例舉出幾個method的用法：

```
01  // app12_9, String 類別使用的範例
02  public class app12_9
03  {
04     public static void main(String args[])
05     {
06        String str="Easier said than done.";
07        System.out.println("length="+str.length());
08        System.out.println("charAt(8)="+str.charAt(8));
09        System.out.println("sub string="+str.substring(7));
10        System.out.println("start with \"th\"="+str.startsWith("th"));
11        System.out.println("upper case="+str.toUpperCase());
12     }
13  }
```

```
/* app12_9 OUTPUT----------------

length=22
charAt(8)=a
sub string=said than done.
start with "th"=false
upper case=EASIER SAID THAN DONE.
---------------------------------*/
```

StringBuffer類別庫

- 要修改字串，必須使用StringBuffer類別
- 下表列出常用的mehtod：

表 12.6.4　StringBuffer 類別常用的 method

method	主要功能
StringBuffer append(char c)	將字元 c 附加到到字串之後
StringBuffer append(String str)	將字串 str 附加到字串之後
StringBuffer deleteCharAt(int index)	刪除字串第 index 位置的字元
StringBuffer insert(int k, char c)	將字串的第 k 個位置插入字元 c
StringBuffer insert(int k, String str)	將字串的第 k 個位置插入字串 str
int length()	取得字串的長度
StringBuffer replace(int m,int n,String str)	將字串第 m 到 n 之間以字串 str 取代
StringBuffer reverse()	將字串反向排列
String toString()	將 StringBuffer 型態的字串轉換成 String 型態

StringBuffer類別使用的範例

- 下面的程式碼爲StringBuffer類別的範例：

```
01  // app12_10, StringBuffer 類別使用的範例
02  public class app12_10
03  {
04     public static void main(String args[])
05     {
06        StringBuffer str=new StringBuffer("Black & White");
07
08        System.out.println(str);
09        System.out.println("length="+str.length());
10        System.out.println(str.replace(0,5,"cats"));
11        System.out.println(str.replace(7,12,"dogs"));
12        System.out.println(str.reverse());
13        System.out.println(str);
14     }
15  }
```

```
/* app12_10 OUTPUT---

Black & White
length=13
cats & White
cats & dogs
sgod & stac
sgod & stac
---------------------*/
```

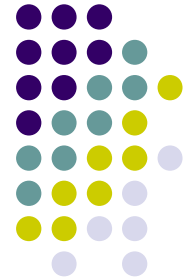
使用wrapper class

- 下表列出原始資料型態與相對應的wrapper class：

表 12.6.5 原始資料型態與其 wrapper class

原始資料型態	wrapper class	原始資料型態	wrapper class
boolean	Boolean	int	Integer
byte	Byte	long	Long
char	Character	float	Float
short	Short	double	Double

- wrapper class提供的變數均屬「類別變數」
- wrapper class所提供的method均是「類別函數」

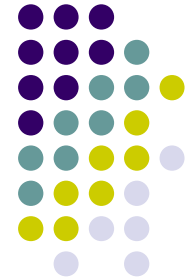


wrapper class提供的method

- 下表列出各種類別常用的轉換函數：

表 12.6.6　各種類別常用的轉換函數

類別	method	主要功能
Byte	static byte parseByte(String s)	將字串 s 轉換成 byte 型態的值
Byte	static String toString(byte b)	將 byte 型態的數值 b 轉換成字串
Character	static String toString(char c)	將字元 c 轉換成字串
Short	static short parseShort(String s)	將字串 s 轉換成短整數
Short	static String toString(short s)	將短整數 s 轉換成字串
Integer	static int parseInt(String s)	將字串 s 轉換成整數
Integer	static String toString(int i)	將整數 i 轉換成字串
Long	static long parseLong(String s)	將字串 s 轉換成長整數
Long	static String toString(Long i)	將長整數 i 轉換成字串
Float	static float parseFloat(String s)	將字串 s 轉換成浮點數
Float	static String toString(float f)	將浮點數 f 轉換成字串
Double	static double parseDouble(String s)	將字串 s 轉換成倍精度浮點數
Double	static String toString(double d)	將倍精度浮點數 d 轉換成字串

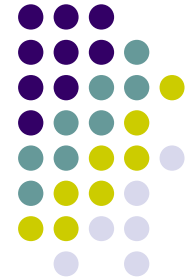


wrapper class的範例

- 下面的程式碼是Integer類別使用的範例：

```
01  // app12_11, Integer class method 的應用
02  public class app12_11
03  {
04     public static void main(String args[])
05     {
06        String str;
07        int inum;
08
09        inum=Integer.parseInt("654")+3;      // 將字串轉成整數後，再加 3
10        System.out.println(inum);
11        str=Integer.toString(inum)+"3";      // 將 "3" 附加在字串後面
12        System.out.println(str);
13     }
14  }              /* app12_11 OUTPUT---

                   657
                   6573
                   ---------------------*/
```

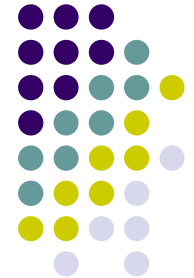


使用Math 類別

- Math類別提供的method可用來計算相關的數學函數
- 下表列出Math 類別所提供的「類別變數」：

表 12.6.7　Math 類別所提供的類別變數

method	主要功能
public static final double E	尤拉常數 (Euler's constant)
public static final double PI	圓周率，π



Math類別的method

表 12.6.8　Math 類別所提供的 method

method	主要功能
public static double sin(double a)	正弦函數，計算 sin(a)
public static double cos(double a)	餘弦函數，計算 cos(a)
public static double tan(double a)	正切函數，計算 tan(a)
public static double asin(double a)	反正弦函數，計算 $\sin^{-1}(a)$
public static double acos(double a)	反餘弦函數，計算 $\cos^{-1}(a)$
public static double atan(double a)	反正切函數，計算 $\tan^{-1}(a)$
public static double exp(double a)	自然指數函數，計算 exp(a)
public static double log(double a)	自然對數函數，計算 log(a)
public static double sqrt(double a)	開根號函數，計算 sqrt(a)
public static double ceil(double a)	傳回大於 a 的最小整數
public static double floor(double a)	傳回小於 a 的最大整數
public static double pow(double a, double b)	計算 a 的 b 次方
public static int round(float a)	傳回最接近 a 的整數
public static double random()	傳回 0.0~1.0 之間的亂數
public static type abs(type a)	計算 a 的絕對值，其中 type 可為 int、long、float 或是 double
public static int max(int a, int b)	找出 a 與 b 中較大者
public static int min(int a, int b)	找出 a 與 b 中較小者

常用到的
數學函數

Math類別的使用範例

- 下面的範例說明數學函數的使用：

```
01  // app12_12, 數學函數的使用
02  public class app12_12
03  {
04     public static void main(String args[])
05     {
06        System.out.println("ceil(3.9)= "+Math.ceil(3.9));
07        System.out.println("sin(PI/2)= "+Math.sin(Math.PI/2));
08        System.out.println("max(8,2)= "+Math.max(8,2));
09     }
10  }
```

```
/* app12_12 OUTPUT---

ceil(3.9)= 4.0
sin(PI/2)= 1.0
max(8,2)= 8
---------------------*/
```