第十二章
大型程式的發展與常用的類別

本章學習重點

- 學習如何分割檔案
- 認識類別庫，以及如何取用在不同類別庫裡的類別
- 建構 package 的階層關係
- 學習 Java 裡常用的類別庫

12.1 檔案的分割

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  }
```

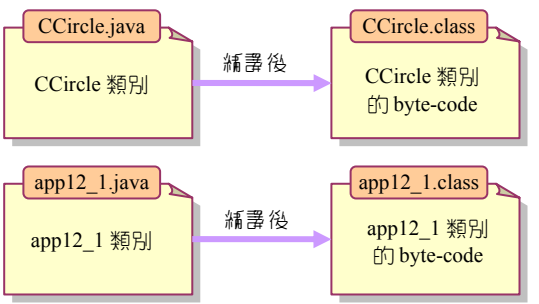
app12_1.java

```
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  }
```

2. 分別以下列的指令編譯 CCircle.java 與 app12_1.java 兩個檔案：

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

編譯後會分別產生 CCircle.class 與 app12_1.class 兩個檔案，如下圖：



3. 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
----------------------*/
```

12.2 使用 package

12.2.1 package 的基本概念

package 使用格式如下：

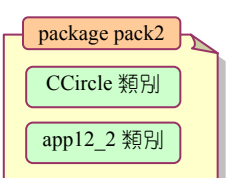
```
package package名稱;
```

格式 12.2.1
package 的宣告

app12_2 是 package 使用的範例。

app12_2.java

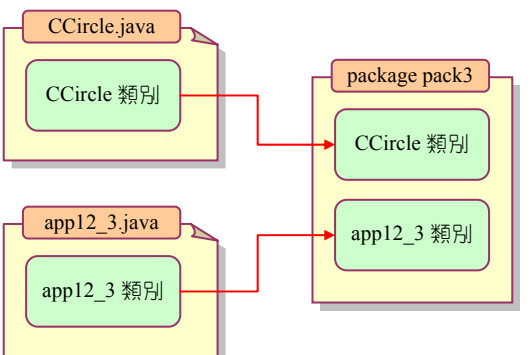
```
01  // app12_2, package 的使用 (- ),此檔案置於 pack2 資料夾內
02  package pack2;   // 宣告以下程式碼所定義的類別均納入 package pack2 中
03  class CCircle    // CCircle 類別被 納入 package pack2 中
04  {
05     public void show()
06     {
07        System.out.println("show() method called");
08     }
09  }
10  public class app12_2    // 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  }
```



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

12.2.2 將不同檔案中的類別組成不同一個 package 中

儲存於不同的檔案中的類別，也可以隸屬於同一個 package：



下面是將下列檔案中的類別納入同一個 package 中的範例：

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  }


/* app12_3 OUTPUT--

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

12.3 存取在不同 package 裡的類別

- ✓ 若某個類別需要被存取時，此類別必須宣告成 public。
- ✓ 若要存取不同 package 內某個 public 類別的成員時，在程式碼內必須明確的指明「被存取 package 的名稱.類別名稱」。

12.3.1 簡單的範例

下面的範例說明了如何存取在不同 package 裡的類別：

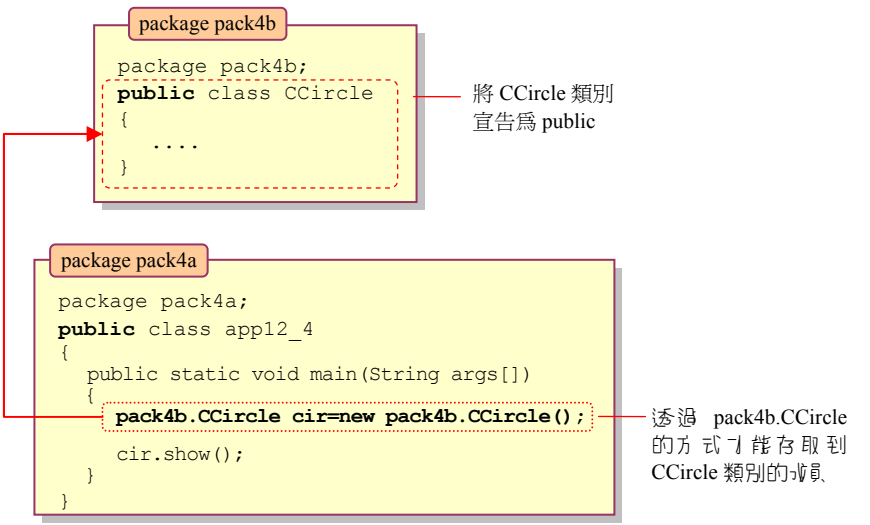
CCircle.java

```
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  }
```

app12_4.java

```
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  }


/* app12_4 OUTPUT---

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

了解說明了如何在 package pack4a 裡存取 package pack4b 裡的物件：



12.3.2 public、private 與 protected 修飾字的角色

下面列出了修飾字相對於類別成員的存取所扮演的角色：

表 12.3.1 類別與介面所使用的修飾字

修飾字	說　　明
沒有修飾字	只能讓同一個 package 裡的類別來存取
public	對它 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字

修飾字	說　　明
沒有修飾字	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

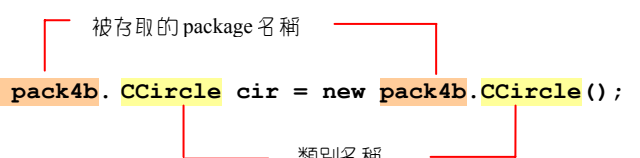
12.3.3 使用 packages

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。

app12_4.java 的第 7 行即是：



透過 import 指令，即可將某個 package 內的特定類別匯入。

```
import package名稱.類別名稱;
```

格式 12.3.1

匯入 package 裡的某個類別

下面是 import 指令的範例：

CCircle.java

```
01  // CCircle.java, package 的使用 (二),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  }
```

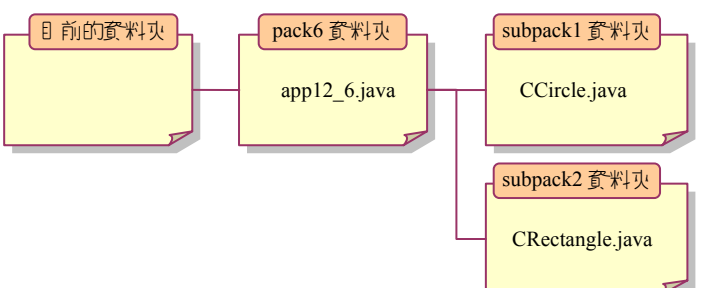
app12_5.java

```
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 的名稱了
10        cir.show();
11     }
12  }


/* app12_5 OUTPUT---

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

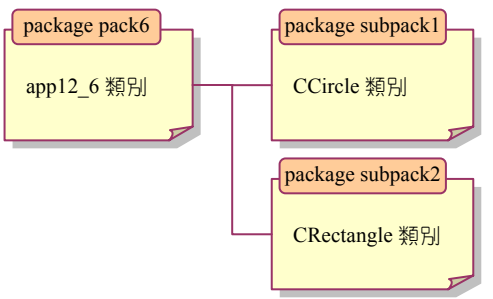
12.4 建構 package 的階層關係

把 packages 劃分為階層的關係，程式碼可以更容易維護。

下面的範例為資料夾與所存放之 Java 原始檔的階層關係圖：



上圖的資料夾階層關係圖可化成 package 的階層關係圖：



要宣告某個類別是屬於某個 sub-package，可用下面的語法來宣告：

package package名稱.sub-package名稱;

格式 12.4.1
sub-package 的宣告格式

下列分別為 CCircle.java、CRectangle.java 與 app12_6.java 的程式碼:

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

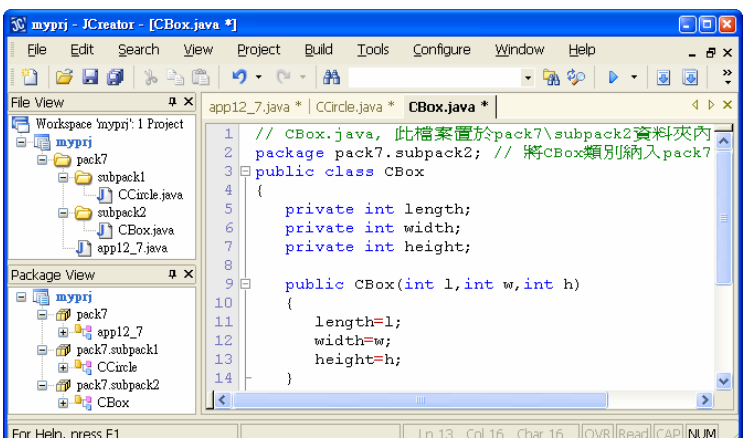
```
01  // CRectangle.java, 此檔案置於 pack6\subpack2 資料夾內
02  package pack6.subpack2; // 將 CRectangle 類別納入 pack6.subpack2 中
03  public class CRectangle
04  {
05     public void show()
06     {
07        System.out.println("show() method of class CRectangle called");
08     }
09  }
```

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
----------------------------------------*/
```

12.5 JCreator 的 Project 管理

JCreator 的 Project，可以協助我們管理類別、介面及程式檔案，發展大型程式。

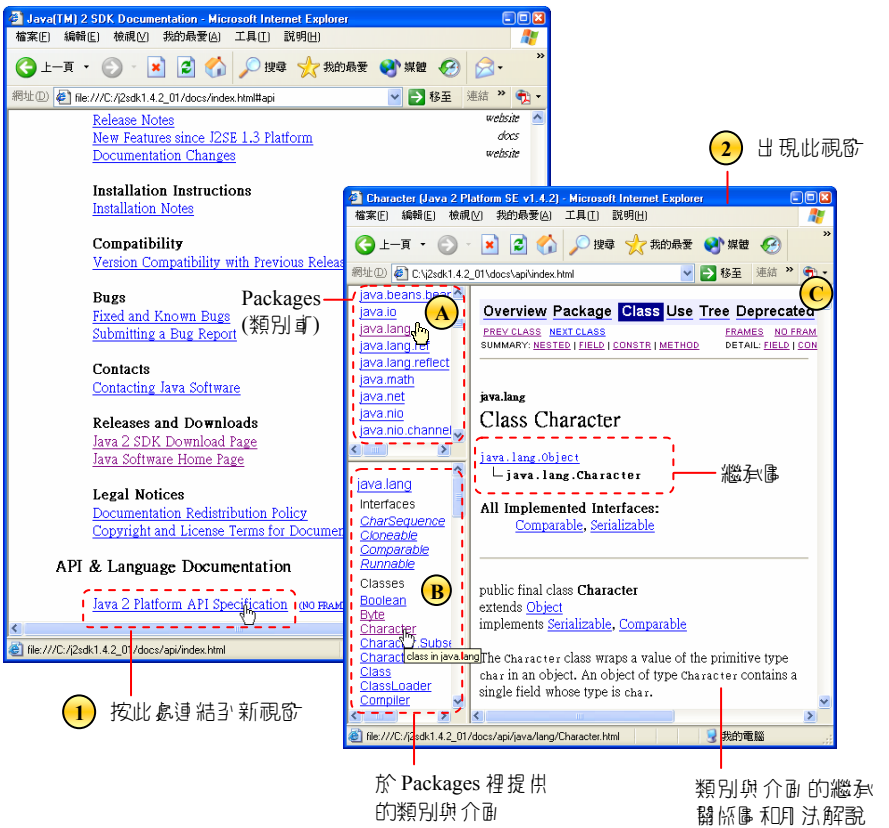


實作的部分請參閱書本內容。

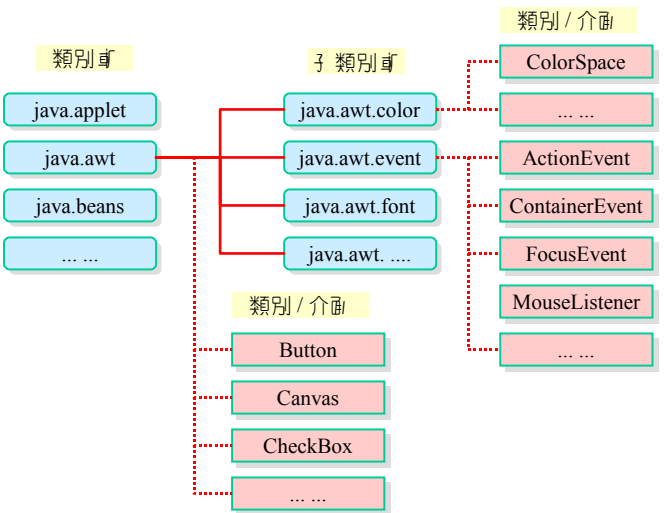
12.6 Java 常用的類別庫

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

Java 提供了描述類別庫的 html 檔，如下一頁所示。


Java(TM) 2 SDK Documentation - Microsoft Internet Explorer
Release Notes
New Features since J2SE 1.3 Platform
Documentation Changes
Installation Instructions
Installation Notes
Compatibility
Version Compatibility with Previous Releas
Bugs
Fixed and Known Bugs
Submitting a Bug Report
Contacts
Contacting Java Software
Releases and Downloads
Java 2 SDK Download Page
Java Software Home Page
Legal Notices
Documentation Redistribution Policy
Copyright and License Terms for Documer
API & Language Documentation
Java 2 Platform API Specification
Packages
(類別庫)
A
B
C
2 出現此視窗
Character (Java 2 Platform SE v1.4.2) - Microsoft Internet Explorer
java.beans
java.io
java.lang
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
java.nio.channel
Overview Package Class Use Tree Deprecated
java.lang
Class Character
java.lang.Object
java.lang.Character
繼承圖
All Implemented Interfaces:
Comparable, Serializable
public final class Character
extends Object
implements Serializable, Comparable
The Character class wraps a value of the primitive type char in an object. An object of type Character contains a single field whose type is char.
java.lang
Interfaces
CharSequence
Cloneable
Comparable
Runnable
Classes
Boolean
Byte
Character
Class
ClassLoader
Compiler
1 按此處連結新視窗
於 Packages 裡提供的類別與介面
類別與介面的繼承關係圖和用法解說

下面的圖為是類別庫、子類別庫、類別與介面之間的層級關係。



下表列出了 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

要匯入 java.awt 類別庫裡的 Button 類別，可用下列的語法：

```
import java.awt.Button;
```

如果要匯入類別庫裡的所有類別時，可以透過萬用字元「*」來匯入：

```
import java.awt.*;
```

12.6.1 字串類別的類別庫

String 類別是放置在 java.lang 類別庫內。

建立字串物件 （String object）

建立 String 物件的方式：

```
String str = "abc";

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

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

下表列出了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，設定為 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，設定為 count 來建立字串
String(String original)	利用原始字串（original string）產生字串物件

字串類別所定義的 method

下表列出了一些常用的 method：

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

下面的範例列舉了幾個 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.
----------------------------------*/
```

12.6.2 StringBuffer 類別庫

要修改字串，必須使用 StringBuffer 類別：

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

下面的程式碼為 String 使用的典型範例：

```
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
-------------------*/
```

12.6.3 wrapper class

下表列出了原始資料型態與相對應的 wrapper class：

表 12.6.5 原始資料型態與對應 wrapper class

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

下表列出了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

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
--------------------*/
```

12.6.4 使用 Math 類別

Math 類別提供的 method 可用來計算相關的數學函數。

下表列出了 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

常用的數學函數列表如下：

表 12.6.8 Math 類別所提供的 method

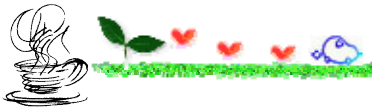
method	主要功能
public static double sin(double a)	正弦函數，計算 sin(a)
public static double cos(double a)	餘弦函數，計算 cos(a)
public static double tan(double a)	正切函數，計算 tan(a)
public static double asin(double a)	反正弦函數，計算 $\sin^{-1}(a)$
public static double acos(double a)	反餘弦函數，計算 $\cos^{-1}(a)$
public static double 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

method	主要功能
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

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
--------------------*/
```



-The End-