第三章
變數與資料型態

本章學習目標

- 認識變數與常數
- 認識 Java 的基本資料型態
- 學習如何進行資料型態轉換
- 學習如何由鍵盤輸入資料

Java 的資料型態分為：

原始資料型態（primitive type）

與

非原始資料型態（non-primitive type）。

原始資料型態包括了整數與浮點數等型態。

非原始資料型態如字串與陣列。

3.1 變數的使用

下面的程式裡宣告了經常使用的變數：

```
01  // app3_1,簡單的實例
02  public class app3_1
03  {
04     public static void main(String args[])
05     {
06        int num=6;       // 宣告 num 為整數，並設值為 6
07        char ch='C';     // 宣告 ch 為字元變數，並設值為'C'
08        System.out.println(num+" is an integer");
09        System.out.println(ch+" is a character");
10     }
11  }
```

```
/* app3_1 OUTPUT---

6 is an integer
C is a character
--------------------*/
```

若想讓變數的值不會再被更改，可以利用 final 將該變數宣告成 final：

```
final 資料型態 變數名稱 = 常數值;
```

格式 3.1.1
變數宣告成 final 的格式

利用 final 宣告的變數，一經設定初值後，其值不能再被更改，如下面的宣告敘述：

```
final double PI=3.1415926;
```

3.2 基本資料型態

下表列出了各種基本資料型態，以及所佔的記憶體空間及範圍：

表 3.2.1 Java 的基本資料型態

資料型態	位元組	表示範圍
long(長整數)	8	-9223372036854775808～9223372036854775807
int(整數)	4	-2147483648～2147483647
short(短整數)	2	-32768～32767
byte(位元)	1	-128～127
char(字元)	1	0～255
boolean(布林)	1	布林值只能使用 true 或 false
float(浮點數)	4	-3.4E38(-3.4×10^{38})～3.4E38(3.4×10^{38})
double(倍精數)	8	-1.7E308(-1.7×10^{308})～1.7E308(1.7×10^{308})

3.2.1 整數型態

整數資料型態可分為 long、int、short 及 byte 四種：

(1) long 佔了 64 個位元（bits），也就是 8 個位元組（bytes）；

(2) int 佔了 32 個位元，也就是 4 個位元組；

(3) 資料值的範圍較小（介於-32768 到 32767 之間）時，可宣告為 short（短整數）；

(4) 若是資料值更小，在-128 到 127 之間時，可以宣告為 byte 以節省記憶體空間。

常數的資料型態

整數常數的型態為 int，但用了超過 2147483647 的常數，編譯時將發生錯誤：

```
01  // app3_2, 整數常數的使用--錯誤的範例
02  public class app3_2
03  {
04     public static void main(String args[])
05     {
06        long num=32967359818;
07        System.out.println("num= "+num);
08     }
09  }
```

編譯後得到下列的錯誤訊息：

```
C:\java\app3_2.java:6: integer number too large:
32967359818
     long num=32967359818;
```

只要把第 6 行的敘述改成：

```
06     long num=32967359818L;
```

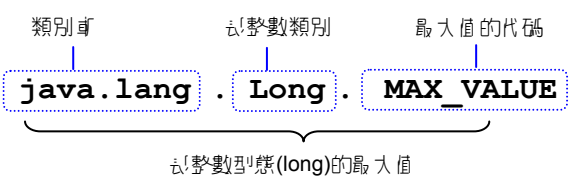
即可成功的編譯與執行。

簡單易記的代碼

Java 提供了 long、int、short 及 byte 四種整數型態的最大值、最小值的代碼。

最大值的代碼是 MAX_VALUE，最小值是 MIN_VALUE。

取整數的最大值，可用下面的語法來表示：



下表為整數之最大值與最小值的識別字及常數值：

表 3.2.2 整數常數的特殊值代碼

	long	int
所屬類別	java.lang.Long	java.lang.Integer
最大值代碼	MAX_VALUE	MAX_VALUE
最大值常數	9223372036854775807	2147483647
最小值代碼	MIN_VALUE	MIN_VALUE
最小值常數	-9223372036854775808	-2147483648

	short	byte
所屬類別	java.lang.Short	java.lang.Byte
最大值代碼	MAX_VALUE	MAX_VALUE
最大值常數	32767	127
最小值代碼	MIN_VALUE	MIN_VALUE
最小值常數	-32768	-128

下面的程式印出 Java 定義的常數之最大值：

```
01  // app3_3,印出 Java 定義的常數之最大值
02  public class app3_3
03  {
04     public static void main(String args[])
05     {
06        long lmax=java.lang.Long.MAX_VALUE;
07        int imax=java.lang.Integer.MAX_VALUE;
08        short smax=Short.MAX_VALUE;   // 省略類別可 java.lang
09        byte bmax=Byte.MAX_VALUE;     // 省略類別可 java.lang
10
11        System.out.println("Max value of long  : "+lmax);
12        System.out.println("Max value of int   : "+imax);
13        System.out.println("Max value of short : "+smax);
14        System.out.println("Max value of byte  : "+bmax);
15     }
16  }
```

```
/* app3_3 OUTPUT----------------------
Max value of long  : 9223372036854775807
Max value of int   : 2147483647
Max value of short : 32767
Max value of byte  : 127
----------------------------------------*/
```

溢位的發生

下面的程式碼示範了整數資料型態的溢位：

```
01  // app3_4,整數資料型態的溢位
02  public class app3_4
03  {
04     public static void main(String args[])
05     {
06        int i=java.lang.Integer.MAX_VALUE;   // 將 i 設為整數的最大值
07
08        System.out.println("i="+i);
09        System.out.println("i+1="+(i+1));
10        System.out.println("i+2="+(i+2));
11     }
12  }
```

```
/* app3_4 OUTPUT---
i=2147483647
i+1=-2147483648
i+2=-2147483647
--------------------*
```

下面的例子說明了資料型態的溢位問題：

在下面的程式中加上防止溢位的處理：

```
01  // app3_5,int 型態的溢位處理
02  public class app3_5
03  {
04     public static void main(String args[])
05     {
06        int i=java.lang.Integer.MAX_VALUE;  // 將 i 設為整數的最大值
07
08        System.out.println("i="+i);
09        System.out.println("i+1="+(i+1));
10        System.out.println("i+2="+(i+2L));
11        System.out.println("i+3="+((long)i+3));
12     }
13  }
```

```
/* app3_5 OUTPUT-----
i=2147483647
i+1=-2147483648
i+2=2147483649
i+3=2147483650
----------------------*/
```

3.2.2 字元型態

字元型態佔有 1 個位元組，可以用來儲存英文字母等字元。

Java 採用了 Unicode（標準萬國碼）。

下面的程式嘗試利用下列的方法來列印出字元 'G'。

```
01  // app3_6,字元型態的列印
02  public class app3_6
03  {
04     public static void main(String args[])
05     {
06        char ch1=71;          // 設定字元變數 ch1 等於編碼為 71 的字元
07        char ch2='G';         // 設定字元變數 ch2 等於'G'
08        char ch3='\u0047';    // 以 16 進位值設定字元變數 ch3
09
10        System.out.println("ch1="+ch1);
11        System.out.println("ch2="+ch2);
12        System.out.println("ch3="+ch3);
13     }
14  }
```

```
/* app3_6 OUTPUT---

ch1=G
ch2=G
ch3=G
----------------------*/
```

下表為常用的跳脫字元：

表 3.2.3 常用的跳脫字元

跳脫字元	所代表的意義	跳脫字元	所代表的意義
\f	換頁(Form feed)	\\	反斜線(Backslash)
\b	倒退一格(Backspace)	\'	單引號(Single quote)
\n	換行(New line)	\"	雙引號(Double quote)
\r	歸位(Carriage return)	\uxxxx	十六進位的unicode字元
\t	跳格(Tab)	\ddd	八進位Unicode字元，範圍在八進位的000~377之間

下面的程式示範了列印跳脫字元：

```
01  // app3_7,列印跳脫字元
02  public class app3_7
03  {
04     public static void main(String args[])
05     {
06        char ch1='\"';       // 將 ch1 設值為 \"
07        char ch2='\74';      // 以八進位值設定字元變數 ch2
08        char ch3='\u003e';   // 以 16 進位值設定字元變數 ch3
09
10        System.out.println(ch1+"Time flies."+ch1);
11        System.out.println("\"Time is money!\"");
12        System.out.println(ch2+"Tomorrow never comes"+ch3);
13     }
14  }
```

```
/* app3_7 OUTPUT------
"Time flies."
"Time is money!"
<Tomorrow never comes>
----------------------*/
```

3.2.3 浮點數型態與倍精度浮點數型態

浮點數型態為 4 個位元組，有效範圍為 $-3.4\times10^{38}\sim3.4\times10^{38}$ 。

倍精度浮點數型態為 8 個位元組，有效範圍為 -1.7×10^{308} 到 1.7×10^{308} 。

下面為 double 與 float 型態的變數之宣告及設值：

```
double num;        // 宣告 num 為倍精度浮點數變數
float sum=2.0f;    // 宣告 sum 為浮點數變數，並設其初值為 2.0
```

下列為 float 與 double 型態之變數在宣告與設值時，應注意的事項：

```
double num1=-5.6e64;      // num1 為 double，其值為 -5.6×10^64
double num2=-6.32E16;     // e 也可以用大寫的 E 來取代
float num3=2.478f;        // sum3 為 float，並設初值為 2.478
float num4=2.63e64;       // 錯，2.63×10^64 超過 float 可表示範圍
```

下面的範例為浮點數的使用：

```
01  // app3_8,浮點數的使用
02  public class app3_8
03  {
04     public static void main(String args[])
05     {
06        float num=5.0f;
07        System.out.println(num+"*"+num+"="+(num*num)); // 印出 num*num 的結果
08     }
09  }

/* app3_8 OUTPUT---
5.0*5.0=25.0
--------------------*/
```

下表為浮點數型態的最大值與最小值的代碼，與所屬類別與所代表之值的範圍：

表 3.2.4 浮點數常數的特殊值

	float	double
所屬類別	java.lang.Float	java.lang.Double
最大值	MAX_VALUE	MAX_VALUE
最大值常數	3.4028235E38	1.7976931348623157E308
最小值	MIN_VALUE	MIN_VALUE
最小值常數	1.4E-45	4.9E-324

下面的程式是印出 float 與 double 兩種浮點數型態的最大與最小值：

```
01  // app3_9,印出 Java 定義的浮點數常數值
02  public class app3_9
03  {
04     public static void main(String args[])
05     {
06        System.out.println("f_max="+Float.MAX_VALUE);
07        System.out.println("f_min="+Float.MIN_VALUE);
08        System.out.println("d_max="+Double.MAX_VALUE);
09        System.out.println("d_min="+Double.MIN_VALUE);
10     }
11  }
```

```
/* app3_9 OUTPUT-----------

f_max=3.4028235E38
f_min=1.4E-45
d_max=1.7976931348623157E308
d_min=4.9E-324
-----------------------------*/
```

3.2.4 布林型態

布林型態的變數，只有 true（真）和 false（假）兩種。

下面是布林型態的變數之使用範例：

```
01  // app3_10,印出布林值
02  public class app3_10
03  {
04     public static void main(String args[])
05     {
06        boolean status=false;   // 設定 status 布林變數的值為 false
07        System.out.println("status="+status);
08     }
09  }
```

```
/* app3_10 OUTPUT---

status=false
--------------------*/
```

3.2.5 基本資料型態的預設值

下表列出了各種型態的預設值：

表 3.2.5 基本資料型態的預設值

資料型態	預設值
byte	(byte)0
short	(short)0
int	0
long	0L
float	0.0f
double	0.0d
char	\u0000
boolean	false

3.3 資料型態的轉換

資料型態的轉換可分為「自動型態轉換」及「強制型態轉換」兩種。

3.3.1 自動型態轉換

下列的條件皆成立時，即會自動做資料型態的轉換：

(1) 轉換前的資料型態與轉換後的型態相容。

(2) 轉換後的資料型態之表示範圍比轉換前的型態大。

下面的程式碼為型態自動轉換的範例：

```
01  // app3_11,型態自動轉換
02  public class app3_11
03  {
04     public static void main(String args[])
05     {
06        int a=45;              // 宣告 a 為整數
07        float b=2.3f;          // 宣告 b 為浮點數
08
09        System.out.println("a="+a+",b="+b);    // 印出 a、b 的值
10        System.out.println("a/b="+(a/b));      // 印出 a/b 的值
11     }
12  }

/* app3_11 OUTPUT---

a=45,b=2.3
a/b=19.565218
----------------------*/
```

3.3.2 強制型態轉換

強制型態轉換的語法如下：

```
(欲轉換的資料型態)變數名稱;
```

格式 3.3.1
資料型態的強制性轉換語法

強制型態轉換也稱為顯性轉換（explicit cast）。

下面的程式說明了整數與浮點數是如何做強制轉換的：

```
01  // app3_12,強制轉換
02  public class app3_12
03  {
04     public static void main(String args[])
05     {
06        int a=36;
07        int b=7;
08
09        System.out.println("a="+a+",b="+b);    // 印出 a、b 的值
10        System.out.println("a/b="+a/b);        // 印出 a/b 的值
11        System.out.println("(float)a/b="+(float)a/b);
12     }
13  }
```

將 a 轉換成浮點數之後，再除以 b

```
/* app3_12 OUTPUT---

a=36,b=7
a/b=5
(float)a/b=5.142857
----------------------*/
```

3.4 由鍵盤輸入資料

下面的格式為輸入資料時所需要撰寫的基本架構：

格式 3.4.1
輸入資料的基本格式

```
import java.io.*;
public class class_name   // 類別名稱
{
   public static void main(String args[]) throws IOException
   {
      BufferedReader buf;        // 宣告 buf 為 BufferedReader 類別的變數
      String str;                // 宣告 str 為 String 型態的變數
      … …
      buf=new BufferedReader(new InputStreamReader(System.in));
      str=buf.readLine();  // 讀入字串至 buf
      … …
   }
}
```

3.4.1 輸入字串

從鍵盤輸入的所有的文字與數字，Java 皆視為字串：

```
01  // app3_13,由鍵盤輸入字串
02  import java.io.*;   // 載入 java.io 類別套裡的所有類別
03  public class app3_13
04  {
05     public static void main(String args[]) throws IOException
06     {
07        BufferedReader buf;
08        String str;
09
10        buf=new BufferedReader(new InputStreamReader(System.in));
11
12        System.out.print("Input a string: ");
13        str=buf.readLine();  // 將輸入的文字指定給字串變數 str 存放
14
15        System.out.println("string="+str);   // 印出字串
16     }
17  }
```

```
/* app3_13 OUTPUT-------------
Input a string: Happy Holiday!!
string=Happy Holiday!!
-----------------------------*/
```

3.4.2 輸入數值

下面的程式是由鍵盤輸入一整數後，再印出這個整數的平方值：

```
01  // app3_14,由鍵盤輸入整數
02  import java.io.*;
03  public class app3_14
04  {
05     public static void main(String args[]) throws IOException
06     {
07        int num;
08        String str;
09        BufferedReader buf;
10
11        buf=new BufferedReader(new InputStreamReader(System.in));
12
13        System.out.print("請輸入一個整數: ");
14        str=buf.readLine();       // 將輸入的文字指定給字串變數 str 存放
15        num=Integer.parseInt(str); // 將 str 轉成 int 型態後指定給 num 存放
16
17        System.out.println(num+" 的平方為 "+num*num);
18     }
19  }
```

```
/* app3_14 OUTPUT---
請輸入一個整數: 8
8 的平方為 64
----------------------*/
```

下表是 Java 所提供的轉換 method：

表 3.4.1 字串轉換成數值型態的 method

資料型態	轉換的 method()
long	Long.parseLong()
int	Integer.parseInt()
short	Short.parseShort()
byte	Byte.parseByte()
double	Double.parseDouble()
float	Float.parseFloat()

3.4.3 輸入多個資料

下面的程式是由鍵盤輸入兩個整數，並把相減後的結果列印到螢幕上：

```
01  // app3_15,由鍵盤輸入多個資料
02  import java.io.*;
03  public class app3_15
04  {
05     public static void main(String args[]) throws IOException
06     {
07        int num1,num2;
08        String str1,str2;
09        BufferedReader buf;
10
11        buf=new BufferedReader(new InputStreamReader(System.in));
12
13        System.out.print("Input first number: ");
14        str1=buf.readLine();        // 將輸入的文字指定給字串變數 str1
15        num1=Integer.parseInt(str1);// 將 str1 轉成 int 型態後給 num1 存放
16
17        System.out.print("Input second number: ");
18        str2=buf.readLine();        // 將輸入的文字指定給字串變數 str2
19        num2=Integer.parseInt(str2);// 將 str2 轉成 int 型態後給 num2 存放
20
```

```
21          System.out.println(num1+"*"+num2+"="+(num1*num2));
22      }
23  }
```

```
/* app3_15 OUTPUT----
Input first number: 3
Input second number: 9
3-9=-6
----------------------*/
```

-The End-