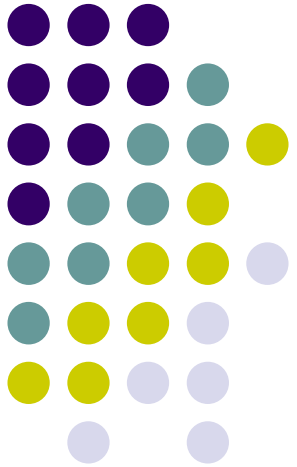
第三章
變數與資料型態



認識變數與常數

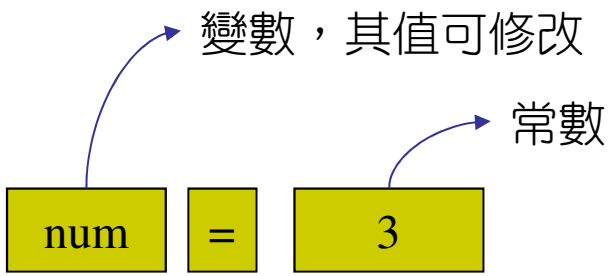
認識Java的基本資料型態

學習如何進行資料型態轉換

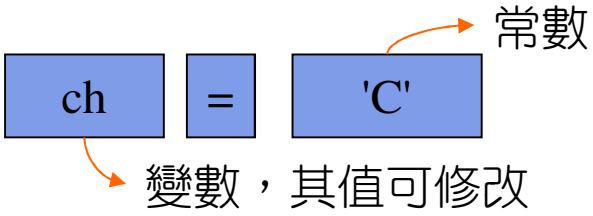
學習如何由鍵盤輸入資料

變數的使用

- 下面是變數使用的範例：




```
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=3;        // 宣告 num 為整數，並設值為 3
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----

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

變數宣告成final的格式

- 若是變數值不會變動，則可將該變數宣告成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，其值不能再被更改，如

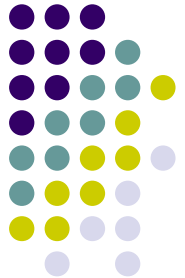
  final double PI=3.1415926;

基本資料型態

- 各種基本資料型態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)	2	0～65535
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(倍精度)	8	-1.7E308(-1.7×10^{308})～1.7E308(1.7×10^{308})



整數型態 int

- 整數型態可分為
  - 長整數 (long int)
  - 整數 (int)
  - 短整數 (short int)
  - 位元(byte)
- 下面為短整數型態宣告的範例：

```
short sum;                    // 宣告sum為短整數
```

常數的資料型態

- Java把整數常數的型態視為int，超過範圍時會發生錯誤

```
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  }
```

敘述改成`long num=32967359818L;`

即可編譯與執行

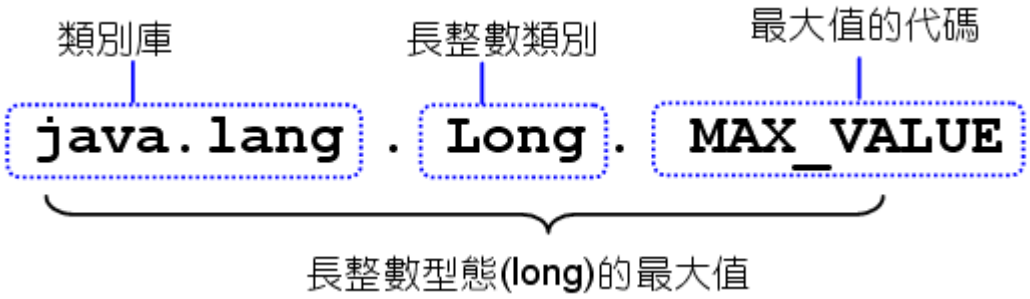
- 編譯上面的程式碼，將會得到下列的錯誤訊息：

```
C:\java\app3_2.java:6: integer number too large: 32967359818

    long num=32967359818;
```

簡單易記的代碼 (1/3)

- 需要用到長整數的最大值時，可用下面的語法來表示：



簡單易記的代碼 (2/3)

- 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

簡單易記的代碼 (3/3)

- 利用整數常數的特殊值代碼列印資料

```
01  // app3_3, 印出 Java 定義的整數常數之最大值
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;   // 省略類別庫 java.lang
09        byte bmax=Byte.MAX_VALUE;     // 省略類別庫 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 (overflow) 的發生 (1/3)

- 溢位：當儲存的數值超出容許範圍時

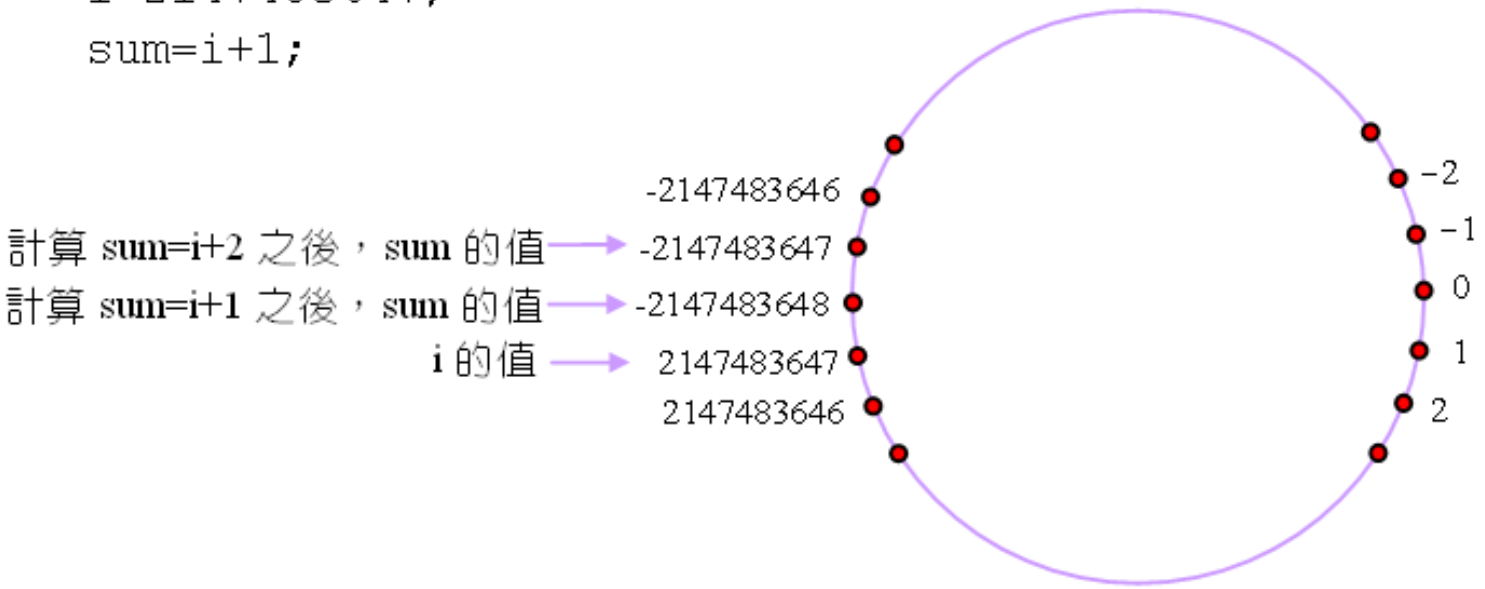
```
01  // app3_4, 整數資料型態的溢位
02  public class app3_4
03  {
04     public static void main(String args[])
05     {
06        int i=java.lang.Integer.MAX_VALUE;     // 將 i 設為整數的最大值
07        int sum;
08
09        System.out.println("i="+i);            // 印出 i 的值
10
11        sum=i+1;
12        System.out.println("i+1="+sum);        // 印出 i+1 的值
13
14        sum=i+2;
15        System.out.println("i+2="+sum);        // 印出 i+2 的值
16     }
17  }
```

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

溢位 (overflow) 的發生 (2/3)

- 下圖說明溢位的發生：

```
i=2147483647;
sum=i+1;
```



溢位 (overflow) 的發生 (3/3)

- int型態的溢位處理範例

```
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));      // 會發生溢位
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
------------------*/
```

字元型態 (1/2)

- 字元型態佔 2 個位元組，用來儲存字元
- Java使用的編碼系統為Unicode(標準萬國碼)
- 宣告字元變數，並設值給它：

```
char ch;            // 宣告字元變數ch
ch='A';             // 將字元常數'A'設值給字元變數ch
```

- 在宣告的同時便設定初值

```
char ch1='A';       // 宣告字元變數ch1，並將字元常數'A'設值給它
char ch2=97;        // 將ch2設值為ASCII碼為97的字元
char ch3='7';       // 將ch3設值為字元常數'7'
```

字元型態 (1/2)

- 下面的程式以不同的格式列印字元變數：

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

跳脫字元 (1/2)

- 反斜線「\」稱為跳脫字元
- 反斜線「\」加上控制碼，稱為跳脫序列

表 3.2.3 常用的跳脫序列

跳脫序列	所代表的意義	跳脫序列	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

跳脫字元 (2/2)

- 利用跳脫序列列印字串：

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

```
01  // app3_7, 列印跳脫序列
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  }
```

可改成char ch2=074;

可改成char ch3=0x3e;

浮點數與倍精度浮點數型態（1/3）

- 浮點數（float）長度為4個位元組

  有效範圍為 $-3.4\times10^{38}\sim3.4\times10^{38}$

- 倍精度（double）浮點數的長度為8個位元組

  有效範圍為 $-1.7\times10^{308}\sim1.7\times10^{308}$

- float與double型態的變數宣告範例如下：

```
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
```

浮點數與倍精度浮點數型態 (2/3)

- float與double宣告與設值時注意事項
  - double num1=-5.6e64; // 宣告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; // 宣告 num3 為 float，並設初值為 2.478
  - float num4=2.63e64; // 錯誤，2.63×10^{64} 超過float可表示的範圍

- 浮點數使用的範例

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

浮點數與倍精度浮點數型態 (3/3)

- 浮點數型態的最大值與最小值的代碼

表 3.2.4 浮點數常數的特殊值代碼

	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

- 下面的範例印出 float 與 double 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
----------------------------*/
```

布林型態

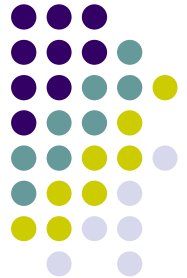
- 宣告布林變數的範例：

```
boolean status=true;      // 宣告布林變數status，並設值為true
```

- 在程式中印出布林值：

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



自動型態的轉換 (1/2)

- 型態轉換發生在運算子左右兩邊的運算元型態不同時
- Java會在下列條件皆成立時，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

  - 例如： int 和 float 相加，int 會被轉成 float

    char 和 int 相加，char 會被轉成 int

- 自動資料型態的轉換只限該行敘述
- 透過自動型態的轉換，可以保證資料的精確度
- 這種轉換也稱為擴大轉換（augmented conversion）

自動型態的轉換 (2/2)

- 浮點數與整數作運算的結果：

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

強制型態轉換 (1/3)

- 將資料型態強制轉換成另一種型態的語法：

資料型態的強制性轉換

```
（欲轉換的資料型態）變數名稱;
```

- 強制型態轉換也稱為顯性轉換（explicit cast）

強制型態轉換 (2/3)

- 整數與浮點數進行強制轉換的範例：

```
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  }
```

```
/* app3_12 OUTPUT-----
a=36,b=7
a/b=5
(float)a/b=5.142857
----------------------*/
```

將a轉換成浮點數後，再除以b

也可寫成

```
a/(float)b
```

```
(float)a/(float)b
```

強制型態轉換 (3/3)

- 強制型態轉換的注意事項
  - 變數強制轉換成另一種型態，原先的型態不會被改變
  - 縮小轉換（narrowing conversion）可能會漏失資料的精確度
  - Java不會主動做縮小轉換

輸入資料的基本架構 (1/4)

- 資料輸入的格式：

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));    // 產生buf物件
      str=buf.readLine();        // 讀入字串至 buf
      … …
   }
}
```

輸入資料的基本架構 (2/4)

- 由鍵盤輸入字串的範例

```
01  // app3_13, 由鍵盤輸入字串
02  import java.io.*;                  // 載入 java.io 類別庫裡的所有類別
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: Love makes the world go round
string=Love makes the world go round
-------------------------------------------*/
```

輸入資料的基本架構 (3/4)

- 由鍵盤輸入整數，再印出其平方值

```
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  }
```

將所輸入的字串轉換成int型態的數值

```
/* app3_14 OUTPUT---
請輸入一個整數: 5
5 的平方為 25
--------------------*/
```

輸入資料的基本架構 (4/4)

- 由鍵盤輸入文字時轉換成數值的method

表 3.4.1 由鍵盤輸入字串時轉換至數值型態的 method

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

輸入數值--不合型態的輸入

- 若是需要輸入數值，卻輸入字元 'U'，則會出現類似下列的錯誤訊息：

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "U"
 at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)
    at java.lang.Integer.parseInt(Integer.java:447)
    at java.lang.Integer.parseInt(Integer.java:497)
    at app3_14.main(app3_14.java:15)
```

輸入多個資料

- 由鍵盤輸入兩個整數的範例：

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

```
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  }
```