




型態變數及運算子

學習目標

- 認識型態與變數
 - 學習運算子基本使用
 - 瞭解型態轉換細節
- 

型態

Java語言一共有八種資料型態(type)

- 位元組整數(byte)
- 短整數(short)
- 字元(char)
- 整數(int)
- 長整數(long)
- 浮點數(float)
- 倍精準度浮點數(double)
- 布林邏輯值(Boolean)

型態

Java可區分為兩大型態系統

- 基本型態（**Primitive type**）直接存取整數、浮點數、字元、或布林值
- 類別型態（**Class type**）間接存取由物件所組成的型態

型態

- 整數
可細分為**short**整數（佔2個位元組）、**int**整數（佔4個位元組）與**long**整數（佔8個位元組）
- 位元組
byte型態，長度就是一個位元組
- 浮點數
可分為**float**浮點數（佔4個位元組）與**double**浮點數（佔8個位元組）

型態

- 字元
char型態用來儲存 ‘A’ 、 ‘B’ 、 ‘林’ 等字元
符號
Java的字元採Unicode 6.2.0編碼
JVM實作採UTF-16 Big Endian，所以每個字元型態
佔兩個位元組，中文字元與英文字元在Java中同
樣都是用兩個位元組儲存
- 布林
boolean型態可表示true與false

型態

類別型態 (**Class type**)

Class

內含各種屬性，方法，事件。也就是說，可以是任何基本型態的組合。

Interface

類似 **Class**，但是所有屬性，方法，事件都沒有實作的程式碼，只是一個空殼子。

Array

可以是一群相同型態的集合。如：整數陣列就是一群整數的集合。

輸出

System.out.printf

System.out.printf的功能完全類似C語言中的printf函數。

printf的一般格式：

printf(格式控制部分，運算式1，運算式2，...運算式n);

print指令

- print將它的引數顯示在命令視窗，並將輸出游標定位在所顯示的最後一個字元之後。
- println 將它的引數顯示在命令視窗，並在結尾加上換行符，將輸出游標定位在下一行的開始。
- printf是格式化輸出。

```
package other;
public class TestPrint {
    public static void main(String[] args) {
        int i = 4;
        double j = 5;
        System.out.print("用print輸出i:" + i);
        System.out.println( "用println輸出i:" + i);
        System.out.printf("i的值為%d,j的值為%f", i,j);

    }
}
```

結果:
用print輸出i:4用println輸出i:4
i的值為4,j的值為5.000000

型態

- 各種型態可儲存的數值範圍，可以透過API來得知：

```
package ch3_1_1_1;
public class Range {
public static void main(String[] args) {
    // byte、short、int、long 範圍
    System.out.printf("%d ~ %d\n", Byte.MIN_VALUE, Byte.MAX_VALUE);
    System.out.printf("%d ~ %d\n", Short.MIN_VALUE, Short.MAX_VALUE);
    System.out.printf("%d ~ %d\n", Integer.MIN_VALUE, Integer.MAX_VALUE);
    System.out.printf("%d ~ %d\n", Long.MIN_VALUE, Long.MAX_VALUE);
    // float、double 精度範圍
    System.out.printf("%d ~ %d\n", Float.MIN_EXPONENT, Float.MAX_EXPONENT);
    System.out.printf("%d ~ %d\n", Double.MIN_EXPONENT, Double.MAX_EXPONENT);
    // char 可儲存的碼元範圍
    System.out.printf("%h ~ %h\n", Character.MIN_VALUE, Character.MAX_VALUE);
    // boolean 的兩個值
    System.out.printf("%b ~ %b\n", Boolean.TRUE, Boolean.FALSE);
}
}
```

型態

- 各種型態可儲存的數值範圍，可以透過**API**來得知：

結果：

```
-128 ~ 127  
-32768 ~ 32767  
-2147483648 ~ 2147483647  
-9223372036854775808 ~ 9223372036854775807  
-126 ~ 127  
-1022 ~ 1023  
0 ~ ffff  
true ~ false
```

註解

- 單行註解 //
- 多行註冊 /* */

```
/* 作者：良葛格  
   功能：示範 printf() 方法  
   日期：2011/7/23  
*/  
public class Demo {  
    ...
```

註解

- 以下使用多行註解的方式是不對的：

```
/* 註解文字 1.....bla...bla
/*
    註解文字 2.....bla...bla
*/
*/
```

System.out.printf()輸出格式

| 符號 | 說明 |
|--------|---|
| %% | 因為%符號已經被用來作為控制符號前置，所以規定使用%%才能在字串中表示%。 |
| %d | 以 10 進位整數格式輸出，可用於 byte、short、int、long、Byte、Short、Integer、Long、BigInteger。 |
| %f | 以 10 進位浮點數格式輸出，可用於 float、double、Float、Double 或 BigDecimal。 |
| %e, %E | 以科學記號浮點數格式輸出，提供的數必須是 float、double、Float、Double 或 BigDecimal。%e 表示輸出格式遇到字母以小寫表示，如 2.13e+12 ，%E 表示遇到字母以大寫表示。 |
| %o | 以 8 進位整數格式輸出，可用於 byte、short、int、long、Byte、Short、Integer、Long、或 BigInteger。 |
| %x, %X | 以 16 進位整數格式輸出，可用於 byte、short、int、long、Byte、Short、Integer、Long、或 BigInteger。%x 表示字母輸出以小寫表示，%X 則以大寫表示。 |

System.out.printf()輸出格式

```
package ch3_1_1_2;  
  
public class float1 {  
  
    public static void main(String[] args) {  
        System.out.printf("example:%.2f%n",19.234);  
        System.out.printf("example:%6.2f%n",19.234);  
    }  
  
}
```

結果:

```
example:19.23  
example: 19.23
```

變數 Variables

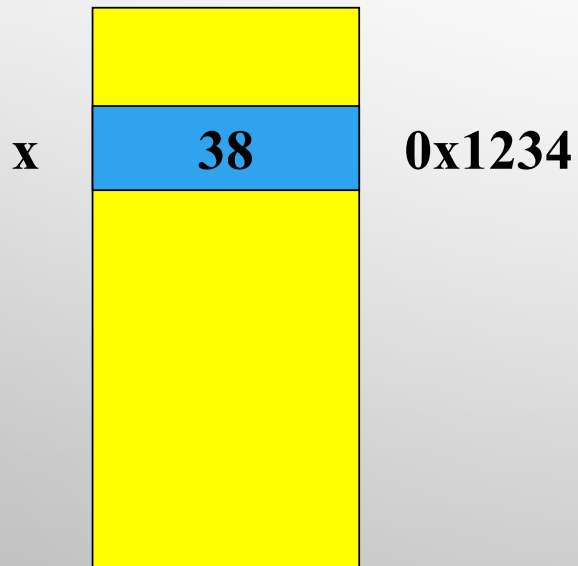
- 變數名稱用來識別變數。
- 型態則是用以指出變數儲存資料的方式。

變數 Variables

基本型態與參考型態在記憶體中的不同

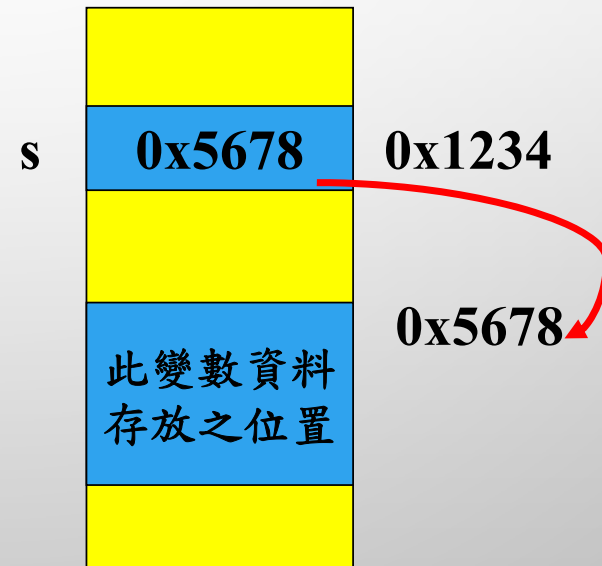
基本型態

`int x = 38;`



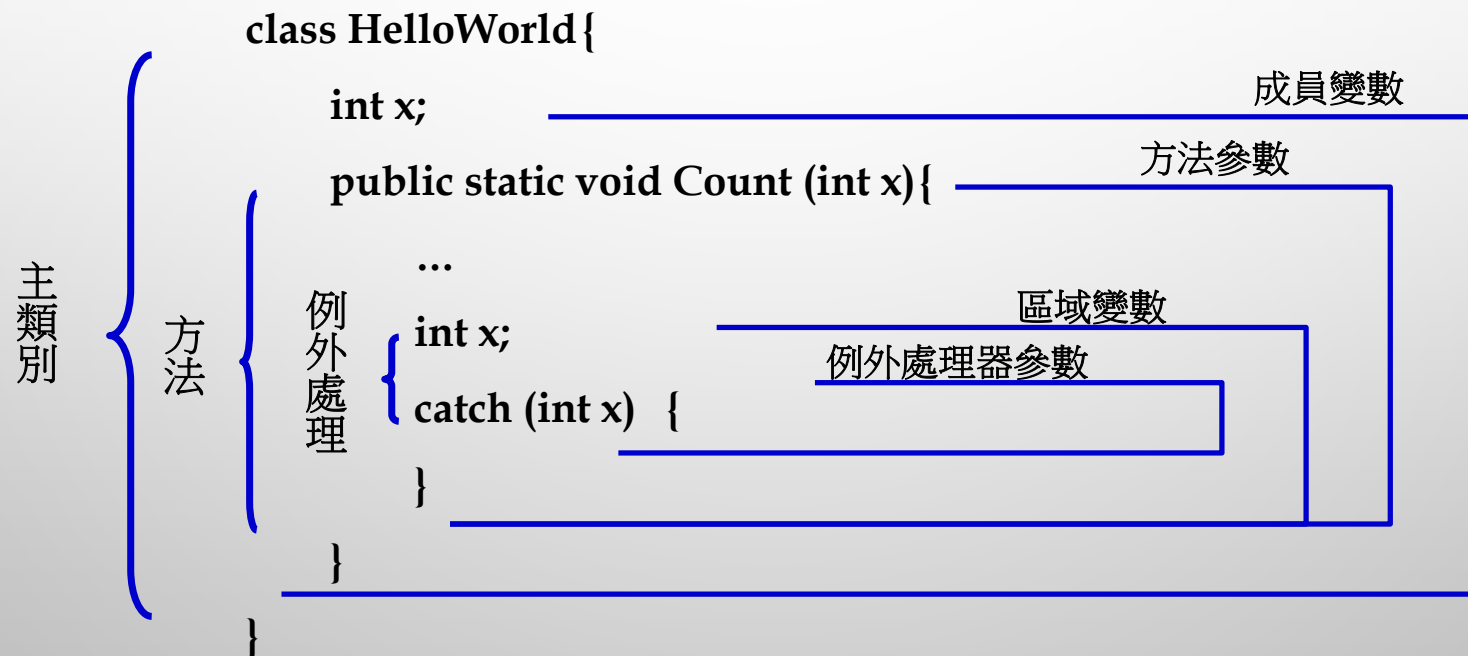
參考型態

`String s;`



變數 Variables

- 變數有效範圍 (**Scope**)
- 變數有效範圍 = 宣告處 ~ 所屬下大括號為止



變數 Variables

變數名稱

- 大小寫不同，可用 **A-Z, a-z, 0-9, _ (底線), \$**，長度不限制
第一個字不可以是數字。
- 不能是關鍵字（**keyword**）或稱保留字 (**null**)、布林字
（**true**、**false**）命名慣例
- 變數名稱以小寫開頭，類別名稱由大寫開頭
- 如果名稱由不同字組成，必須將字組相連，建議後面字的首字元大寫，例：**myParameter**
- 常數（**constant**）通常全部大寫，且使用底線（**_**）分開字組，例：**CURRENT_USER_NUM**

變數 Variables

🔑 關鍵字

🔑 **Java** 保留給本身所使用的文字，也有人稱為 “保留字” (Reserved Words)

Keyword:

abstract default if private this boolean do implements
protected throw break double import public throws
byte else instanceof return transient case extends int short
try catch final interface static void char finally
long strictfp volatile class float native super while const
for new switch continue goto package synchronized

變數 Variables

- 程式中輸出10的部份很多，如果想要一次把它改為20 ...

```
package ch3_1_2_2;

public class var2 {

    public static void main(String[] args) {
        int number = 10;
        double PI = 3.14;
        System.out.println(10);
        System.out.println(3.14);
        System.out.println(10);

        System.out.println(number);
        System.out.println(PI);
        System.out.println(number);
    }
}
```

```
double score=20.55;
System.out.println(score);
}

}
```

結果:

```
10
3.14
10
10
3.14
10
20.55
```

變數 Variables

- 想要宣告何種型態的變數，就使用**byte**、**short**、**int**、**long**、**float**、**double**、**char**、**boolean**等關鍵字來宣告
- 變數在命名時有一些規則，它不可以使用數字作為開頭，也不可以使用一些特殊字元，像是*、&、^、%之類的字元
- 變數名稱不可以與Java 的關鍵字（**Keyword**）同名，例如**int**、**float**、**class**等就不能用來作為變數
- 變數名稱也不可以與Java保留字（**Reversed word**）同名，例如**goto**就不能用來作為變數名稱

變數 Variables

- 在Java領域中的命名慣例（**Naming convention**），通常會以小寫字母開始，並在每個單字開始時第一個字母使用大寫，稱為駝峰式（**Camel case**）命名法

```
int ageOfStudent;  
int ageOfTeacher;
```

變數 Variables

常數

語法：**final** *DataType* *VarName* = *InitialValue*;
一旦變數被宣告成 **final**，其值將不可再更改
若程式欲更改 **final** 變數的值將造成編譯時期錯誤
(**compile-time error**)

範例：

```
final double PI = 3.1416;  
PI = 2.345; // 會發生錯誤，因為 PI 是常數
```

變數 Variables

- 字面常數（ LITERAL CONSTANT ）

```
int number1 = 12;    // 10 進位表示  
int number2 = 0xC;   // 16 進位表示，以 0x 開頭  
int number3 = 014;  // 8 進位表示，以 0 開頭
```

```
double number1 = 0.00123;  
double number2 = 1.23e-3;
```

```
char size = 'S';  
char lastName = '林';
```

```
char symbol = '\\';
```

```
boolean flag = true;  
boolean condition = false;
```


變數 Variables

| 忽略符號 | 說明 |
|--------|---------------------------------|
| \\ | 反斜線\。 |
| \' | 單引號'。 |
| \" | 雙引號"。 |
| \uxxxx | 以 16 進位數指定 Unicode 字元輸出，x 表示數字。 |
| \xxx | 以 8 進位數指定 Unicode 字元輸出，x 表示數字。 |
| \b | 倒退一個字元。 |
| \f | 換頁。 |
| \n | 換行。 |
| \r | 游標移至行首。 |
| \t | 跳格(按下 Tab 鍵的字元)。 |

```
System.out.println("\u0048\u0065\u006C\u006C\u006F");
```

變數 Variables

- 常量表示法

```
int number1 = 1234_5678;  
double number2 = 3.141_592_653;
```

```
int mask = 0b101010101010;    // 用二進位表示 10 進位整數 2730
```

```
int mask = 0b1010_1010_1010;   // 用二進位表示 10 進位整數 2730
```

變數 Variables

| 型態名稱 | 大小 / 格式 | 範圍 | 說明 |
|--|----------------------|--|----------------------|
| 整數型態 負數: 二進位數型態首位為1時(11110111 → 先減1 → 11110110 → 取補數 → 00001001 → 9 → 真正的值 → -9) | | | |
| byte | 8 bit / 二補數方式 | -128 ~ 127 | 位元組整數 (Byte Integer) |
| short | 16 bit / 二補數方式 | -32768 ~ 32767 | 短整數 (Short Integer) |
| int | 32 bit / 二補數方式 | -2147483648 ~ 2147483647 | 整數 (Integer) |
| long | 64 bit / 二補數方式 | -9223372036854775808 ~ 9223372036854775807 | 長整數 (Long Integer) |
| 實數型態 (以近似表示任意某個實數； $a = m * b^e$) | | | |
| float | 32 bit / IEEE 754 格式 | 七位小數 | 單精度浮點數 |
| double | 64 bit / IEEE 754 格式 | 十五位小數 | 雙精度浮點數 |
| 其它型態 | | | |
| char | 16 bit / Unicode 格式 | \u0000 ~ \uffff (0 ~ 65535) | 字元 |
| boolean | true / false | true, false | 布林值 |

運算子

- 算數運算

%運算子計算的結果是除法後的餘數

```
int count = 0;  
....  
count = (count + 1) % 360;
```

運算子

- 以下程式碼片段會在文字模式下顯示7

```
System.out.println(1 + 2 * 3);
```

- 以下程式碼會顯示的是6：

```
System.out.println(2 + 2 + 8 / 4);
```

- 以下程式碼顯示的是3：

```
System.out.println((2 + 2 + 8) / 4);
```

運算子

- 比較運算子（COMPARISON OPERATOR）

| 運算子 | 含義 | 用法舉例 |
|-----|-------|---|
| > | 大於 | <code>scoreMath>scoreEnglish</code> |
| >= | 大於或等於 | <code>scoreMath>=scoreEnglish</code> |
| < | 小於 | <code>scoreMath<scoreEnglish</code> |
| <= | 小於或等於 | <code>scoreMath<=scoreEnglish</code> |
| == | 等於 | <code>scoreMath==scoreEnglish</code> |
| != | 不等於 | <code>scoreMath!=scoreEnglish</code> |

運算子

- 比較運算子 (COMPARISON OPERATOR)

```
package ch3_1_3_1;  
public class Comparison {  
    public static void main(String[] args) {  
        System.out.printf("10 > 5 結果 %b%n", 10 > 5);  
        System.out.printf("10 >= 5 結果 %b%n", 10 >= 5);  
        System.out.printf("10 < 5 結果 %b%n", 10 < 5);  
        System.out.printf("10 <= 5 結果 %b%n", 10 <= 5);  
        System.out.printf("10 == 5 結果 %b%n", 10 == 5);  
        System.out.printf("10 != 5 結果 %b%n", 10 != 5);  
    }  
}
```

結果:

```
10 > 5 結果 true  
10 >= 5 結果 true  
10 < 5 結果 false  
10 <= 5 結果 false  
10 == 5 結果 false  
10 != 5 結果 true
```

運算子

- 條件運算子（**CONDITIONAL OPERATOR**）

```
System.out.printf("該生是否及格?%c%n", score >= 60 ? '是' : '否');
```

```
System.out.printf("是否為偶數?%c%n", (number % 2 == 0) ? '是' : '否');
```

```
if(number % 2 == 0) {  
    System.out.println("是否為偶數?是");  
}  
else {  
    System.out.println("是否為偶數?否");  
}
```


運算子

- 邏輯運算

| 運算子 | 含義 | 用法舉例 | 備註 |
|-----|-------|-----------------------------------|---------------------------|
| & | And | (scoreMath>=60)&(scoreMath<70) | 運算元為邏輯型 |
| | Or | (scoreMath>=90) (totalScore<570) | 運算元為邏輯型 |
| ! | Not | !(scoreMath>=60) | 運算元為邏輯型 |
| && | 簡潔and | (scoreMath>=60)&&(scoreMath<70) | 前一個運算元為false時後一個運算元不參與計算 |
| | 簡潔or | (scoreMath>=60) &&(scoreMath<70) | 前一個運算元為true時後一個運算元不參與計算 |
| ^ | XOR | (scoreMath>=60)^(scoreMath<70) | 兩個運算元不同時，結果為true,否則為false |

運算子

- 邏輯運算

```
int number = 75;  
System.out.println(number > 70 && number < 80);  
System.out.println(number > 80 || number < 75);  
System.out.println(!(number > 80 || number < 75));
```

運算子

- 位元運算

```
package ch3_1_3_3;
public class Bitwise {
public static void main(String[] args) {
    System.out.println("AND 運算 :");
    System.out.printf("0 AND 0 %5d\n", 0 & 0);
    System.out.printf("0 AND 1 %5d\n", 0 & 1);
    System.out.printf("1 AND 0 %5d\n", 1 & 0);
    System.out.printf("1 AND 1 %5d\n", 1 & 1);

    System.out.println("\nOR 運算 :");
    System.out.printf("0 OR 0 %6d\n", 0 | 0);
    System.out.printf("0 OR 1 %6d\n", 0 | 1);
    System.out.printf("1 OR 0 %6d\n", 1 | 0);
    System.out.printf("1 OR 1 %6d\n", 1 | 1);

    System.out.println("\nXOR 運算 :");
    System.out.printf("0 XOR 0 %5d\n", 0 ^ 0);
    System.out.printf("0 XOR 1 %5d\n", 0 ^ 1);
    System.out.printf("1 XOR 0 %5d\n", 1 ^ 0);
    System.out.printf("1 XOR 1 %5d\n", 1 ^ 1);
}
```

結果:

```
AND 運算 :
0 AND 0      0
0 AND 1      0
1 AND 0      0
1 AND 1      1

OR 運算 :
0 OR 0       0
0 OR 1       1
1 OR 0       1
1 OR 1       1

XOR 運算 :
0 XOR 0      0
0 XOR 1      1
1 XOR 0      1
1 XOR 1      0
```

運算子

- 補數運算是將所有位元0變1，1變0。例如000000001經補數運算就會變為11111110

```
byte number = 0;  
System.out.println(~number);
```

運算子

- 左移（<<）與右移（>>）

| 運算子 | 含義 | 用法舉例 | 備註 |
|-----|-------|------------|------------------|
| >> | 按位右移 | $x \gg y$ | 左運算元為數值型，右運算元為整型 |
| << | 按位左移 | $x \ll y$ | 左運算元為數值型，右運算元為整型 |
| >>> | 無符號右移 | $x \ggg y$ | 左運算元為數值型，右運算元為整型 |

運算子

- 左移 (<<) 與右移 (>>)

```
package ch3_1_3_4;

public class Shift {

    public static void main(String[] args) {
        int number = 1;
        System.out.printf( "2 的 0 次方: %d\n", number);
        System.out.printf( "2 的 1 次方: %d\n", number << 1);
        System.out.printf( "2 的 2 次方: %d\n", number << 2);
        System.out.printf( "2 的 3 次方: %d\n", number << 3);
    }
}
```

結果:

| | | | | |
|---|---|---|-----|---|
| 2 | 的 | 0 | 次方: | 1 |
| 2 | 的 | 1 | 次方: | 2 |
| 2 | 的 | 2 | 次方: | 4 |
| 2 | 的 | 3 | 次方: | 8 |

運算子

- 指定運算

依照運算子擺放位置，可分為

前置（**prefix**）運算子：運算子擺放在運算元之前，例： $++i$

後置（**postfix**）運算子：運算子擺放在運算元之後，例： $i++$

中置（**infix**）運算子：運算子擺放在中間，例： $a + b$

運算子

```
int i = 0;  
int number = 0;  
number = ++i;    // 結果相當於 i = i + 1; number = i;  
System.out.println(number);  
number = --i;    // 結果相當於 i = i - 1; number = i;  
System.out.println(number);
```

```
int i = 0;  
int number = 0;  
number = i++;    // 相當於 number = i; i = i + 1;  
System.out.println(number);  
number = i--;    // 相當於 number = i; i = i - 1;  
System.out.println(number);
```


運算子

- 遞增、遞減運算

```
int i = 0;  
i = i + 1;  
System.out.println(i);  
i = i - 1;  
System.out.println(i);
```

```
int i = 0;  
i++;  
System.out.println(i);  
i--;  
System.out.println(i);
```

```
int i = 0;  
System.out.println(++i);  
System.out.println(--i);
```

運算子

- 指定運算

| 指定運算子 | 範例 | 結果 |
|-------|---------|------------|
| += | a += b | a = a + b |
| -= | a -= b | a = a - b |
| *= | a *= b | a = a * b |
| /= | a /= b | a = a / b |
| %= | a %= b | a = a % b |
| &= | a &= b | a = a & b |
| = | a = b | a = a b |
| ^= | a ^= b | a = a ^ b |
| <<= | a <<= b | a = a << b |
| >>= | a >>= b | a = a >> b |

基底資料型別的轉換

- 基底資料型別的轉換就是把一種基底資料型別變數轉變成另一種基本類型變數。
- 下列基本類型會涉及資料轉換，不包括邏輯類型和字元類型。我們將這些類型按精度從“低”到“高”排列了順序：

byte short int long float double

- 當把級別低的變數的值賦給級別高的變數時，系統自動完成資料類型的轉換，如int型轉換成long型。
- 當把級別高的變數的值賦給級別低的變數時，必須使用顯示類型轉換運算。
- 顯示轉換的格式：（類型名）要轉換的值；

型態轉換

轉換變數型態

型態相容時

相容時，Java 會自動加以轉換

```
int a;
```

```
long b = a; (相容會自動轉換)
```

型態不相容時

兩種不相容的型態要轉換，必須使用強制轉換

語法： (target-type) value

```
int a;
```

```
b = (byte) a; (縮小轉換) - 將會被縮減成byte範圍的餘數
```

```
char x = (char) 65;
```

基底資料型別的轉換

```
package ch3_1_3_5;

public class conv {

    public static void main(String[] args) {
        int x=(int)34.89;
        long y=(long)56.98F;
        byte a=(byte)128;
        byte b=(byte)(-129);
        byte c=(byte)(-130);
        byte d=(byte)(129);
        byte e=(byte)(130);
        System.out.printf("x=%d\n",x);
        System.out.printf("y=%d\n",y );
        System.out.printf("(byte)128=%d\n",a );
        System.out.printf("(byte)(-129)=%d\n",b );
        System.out.printf("(byte)(-130)=%d\n",c );
        System.out.printf("(byte)(129)=%d\n",d );
        System.out.printf("(byte)(130)=%d\n",e );
    }
}
```

byte的取值範圍為： -128 ~ 127

1000 0000是負數（第一位為1），負數用補數表示

1000 0000即128

0000 0000 0000 0000 0000 0000 1000 0000 編碼是128

1111 1111 1111 1111 1111 1111 0111 1111編碼是-129

所以強轉 128 會發生“溢位”，就變成了 -128

129 就自然為 -127

結果：

```
x=34
y=56
(byte)128=-128
(byte)(-129)=127
(byte)(-130)=126
(byte)(129)=-127
(byte)(130)=-126
```

基底資料型別的轉換

```
package ch3_1_3_6;

public class conv2 {

    public static void main(String[] args) {
        int x=(int)34.89;
        long y=(long)56.98F;
        System.out.printf("x=%d%n",x);
        System.out.printf("y=%d%n",y );

        for(var j = -130; j < 131; j++) {

            byte a=(byte) j;

            System.out.printf("(byte)%d=%d%n",j,a );

        }

    }

}
```

結果:

```
x=34
y=56
(byte)-130=126
(byte)-129=127
(byte)-128=-128
(byte)-127=-127
(byte)-126=-126
(byte)-125=-125
(byte)-124=-124
(byte)-123=-123
(byte)-122=-122
```

•
•
•

```
(byte)122=122
(byte)123=123
(byte)124=124
(byte)125=125
(byte)126=126
(byte)127=127
(byte)128=-128
(byte)129=-127
(byte)130=-126
```

型態轉換

- 浮點數時，編譯器預設會使用**double**型態
- 編譯器會告知想將**double**長度的資料指定給**float**型態變數，會因為**8**個位元組資料要放到**4**個位元組空間，而遺失**4**個位元組的資料

型態轉換

- 兩種方式可以避免這個錯誤...

```
float PI = 3.14F;
```

```
float PI = (float) 3.14;
```

- 使用(float)語法告訴編譯器，你就是要將double型態的3.14指定給float變數

型態轉換

- 整數時，預設是使用不超過int型態長度2147483648超出了int型態的長度
- 直接告訴編譯器，用long來配置整數的長度，也就是在數字後加上個L

```
long number = 2147483648L;
```

字串

- 字串類別 **String**:注意**String**的第一個字是大寫的，這表示它是一個類別，而不是型別。
用法有一點類似型別
- **String**類別可以用等號(=)來設定初始值，並且可以使用加號(+)來執行連結(**concatenate**)運算等

例如：

String 字串1="歡迎來到Java世界";

字串

- 字串既然是一連串的字元所組成的，因此所有字元都可以出現在字串中
- 例如：
- **String** 字串= “今年是\t 公元\n \t 2001年\n \t !!!!” ; 變數字串的初始值中即含\t及\n兩個特別字元
- 若是使用**System.out.print(字串);**
結果：

今年是 公元
 2001年
 !!!!

字串

- `\t`代表定位字元，會將列印的位置移動到後方定位點上，在Java語言中定位點是位於8的倍數的位置。
- `\n`代表換行字元，此字元會使字串換行列印

Example:

```
System.out.print("歡迎來到\n Java\n世界");
```

這個敘述與下列三個敘述同義：

```
System.out.println("歡迎來到");
```

```
System.out.println("Java");
```

```
System.out.println("世界");
```