

Java程式設計入門

運算子

鄭安翔

ansel_cheng@hotmail.com

課程大綱

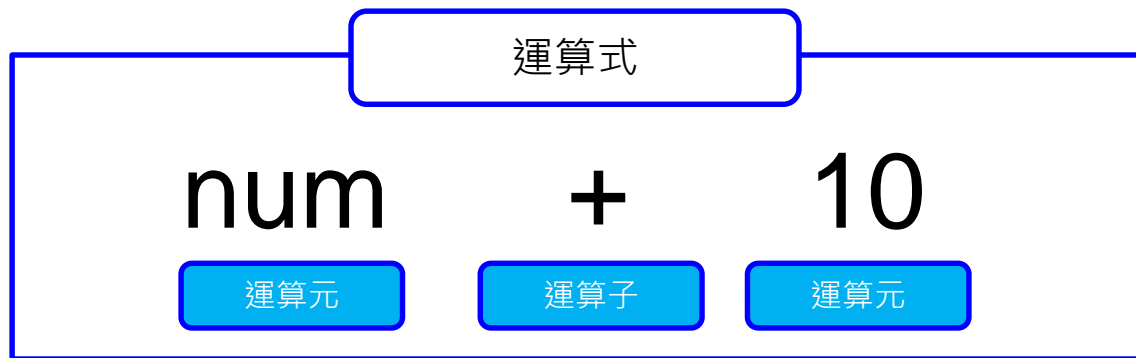
1. Java 的運算子

- 算數運算子(Arithmetic Operator)
- 關係運算子(Relational Operator)
- 邏輯運算子(Logic Operator)
- 位元運算子(Bitwise Operator)
- 指定運算子(Assignment Operator)
- 條件(三元)運算子(Conditional (Ternary) Operator)

2. Java 運算子的優先順序

運算式

- 運算式是由運算元與運算子所組成
- 運算元可以是變數或字面值
- 運算子是程式裡用於執行計算動作的符號
- 例如



運算子分類

- 運算子依運算元數目可分成

- 一元運算子(一般運算子的前後皆會有運算元，但一元運算子只有一個運算元)

- `-5`、`i++` 或 `!booA`

- 二元運算子

- `x + y`

- 三元運算子

- `max = a > b ? a : b`

//判斷 `a` 是否大於 `b`，若成立，將 `a` 指派給 `max`，若不成立，將 `b` 指派給 `max`

算術運算

運算子	說明	範例	類別
+	正號	+2 ➔ 取得正數 2	一元
	加法	2 + 1 = 3	二元
-	負號	-2 ➔ 取得負數 2	一元
	減法	2 - 1 = 1	二元
*	乘法	2 * 3 = 6	二元
/	除法	4 / 2 = 2	二元
%	餘數	3 % 2 = 1，求得 3 除以 2 的餘數	二元
++	遞增	前置遞增：int A = 0；A = ++A；表示 A 先加上 1 之後再指派給A所以A=1。	一元
		後置遞增：int A = 0；A = A++；表示A（此時的A=0）會先指派給 A 之後才會執行 ++ 的動作但 A 仍然是 0。	
--	遞減	前置遞減：int A = 0；A = --A；表示 A 先減 1 之後再指派給 A 所以 A = - 1	一元
		後置遞減：int A = 0；A = A--；表示 A（此時的A=0）會先指派給 A 之後才會執行 -- 的動作但 A 仍然是 0。	

```

01 public class ArithmeticDemo {
02     public static void main(String[] args) {
03         //宣告變數與值
04         int x = 10;
05         int y = 5;
06         System.out.println("變數值...");
07         System.out.println(" x = " + x);
08         System.out.println(" y = " + y);
09         //加法示範
10         System.out.println("加法...");
11         System.out.println(" x+y= " + (x+y));
12         //減法示範
13         System.out.println("減法...");
14         System.out.println(" x-y= " + (x-y));
15         //乘法示範
16         System.out.println("乘法...");
17         System.out.println(" x*y= " + (x*y));
18         //除法示範
19         System.out.println("除法...");
20         System.out.println(" x/y= " + (x/y));
21         //模數運算
22         System.out.println("計算餘數...");
23         System.out.println(" x%y= " + (x%y));
24     }
25 }

```

```

C:\JavaClass>java ArithmeticDemo
變數值...
x = 10
y = 5
加法...
x+y= 15
減法...
x-y= 5
乘法...
x*y= 50
除法...
x/y= 2
計算餘數...
x%y= 0
C:\JavaClass>

```

算術運算與自動轉型規則

■ 算術運算與自動轉型

- Java 會將運算中所有的operands,自動promote成最大operand的型別來運算

- 若所有operands都小於int,自動promote成int來運算

```
short a = 1,
```

```
short b = 2
```

```
short c = (short) (a+b);
```

■ String Concatenation

- + 中只要有一個operand 是String,另一個operand也會被轉成String,結果為字串串接

```
System.out.println(3+4+"str"); //7str
```

```
System.out.println ("str"+3+4); //str34
```

/ 除法

■ 整數除整數

- $10 / 3 = 3 \dots 1$

餘數捨棄

- $10 / 0 = \text{throws } \text{ArithmeticException}$

分母不可為0

■ 除數或被除數至少有一為浮點數

- $10.0 / 3 = 3.3333333333333335$ //近似值，不丟棄餘數

- $10.0 / 0.0 = \text{infinity}$

- $-10.0 / 0.0 = \text{-infinity}$

- $0.0 / 0.0 = \text{NaN}$ //分母可為0

■ 浮點數中額外定義:

$+\infty$:一個大於該型態其所能表示的最大值的正浮點數

$-\infty$:一個小於該型態所能表示的最大負值的負浮點數

$+0.0$:一個接近0的正浮點數,超過該型態可表示之精確度

-0.0 :一個接近0的負浮點數,超過該型態可表示之精確度

NaN :浮點數做不適當運算 **ex:** $0.0/0.0$

% 餘數

■ 整數 % 整數

□ $10 \% 0 = \text{throws } \text{ArithmeticException}$

//分母不可為0

□ $11 \% 3 = 2$ $11 - 3 - 3 - 3 \rightarrow |2| < |3|$

□ $11 \% (-3) = 2$ $11 - 3 - 3 - 3 \rightarrow |2| < |-3|$

□ $(-11) \% 3 = -2$ $-11 + 3 + 3 + 3 \rightarrow |-2| < |3|$

□ $(-11) \% (-3) = -2$ $-11 + 3 + 3 + 3 \rightarrow |-2| < |-3|$

■ 有一個以上的operand為浮點數

□ $10.0 \% 0.0 = \text{NaN}$

□ $0.0 \% 0.0 = \text{NaN}$

□ $(5/0.0) \% 2.0 = \text{NaN}$ ($\infty \% 2.0$)

□ $8.0 \% 3 = 2.0$

□ $6.5 \% 3.2 = 0.1$ (商須為整數)

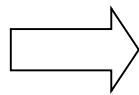
□ $2.0 \% (5/0.0) = 2.0$ ($2.0 \% \infty$)

遞增遞減運算

- $X = X + 1;$ \Rightarrow 遞增運算 $X++, ++X$
- $X = X - 1;$ \Rightarrow 遞減運算 $X--, --X$

- 後置遞增, 後置遞減

`int var = age++;`



`int var = age;`
`age = age + 1;`

```
01 public class UnaryOperatorDemo {  
02     public static void main(String[] args) {  
03         int age = 10;  
04         int var = age++;  
05         System.out.println("age="+age);  
06         System.out.println("var="+var);  
07     }  
08 }
```

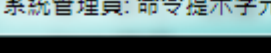
```
C:\JavaClass>java UnaryOperatorDemo  
age=11  
var=10  
C:\JavaClass>
```

遞増遞減運算

■ 前置遞增,前置遞減

```
int var = ++age;    ⇒    age = age + 1  
                     int var = age;
```

```
01 public class UnaryOperatorDemo2 {
02     public static void main(String[] args) {
03         int age = 10;
04         int var = ++age;
05         System.out.println("age="+age);
06         System.out.println("var="+var);
07     }
08 }
```

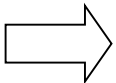


系統管理員: 命令提示字元

```
c:\JavaClass>java UnaryOperatorDemo2
age=11
var=11
c:\JavaClass>
```

遞增遞減運算

```
int a=0, b=0;
```

```
b=a++ + ++a;        a=2; b=2
```

```
int a=0, b=0;
```

```
b=a-- - --a;        a=-2; b=2
```

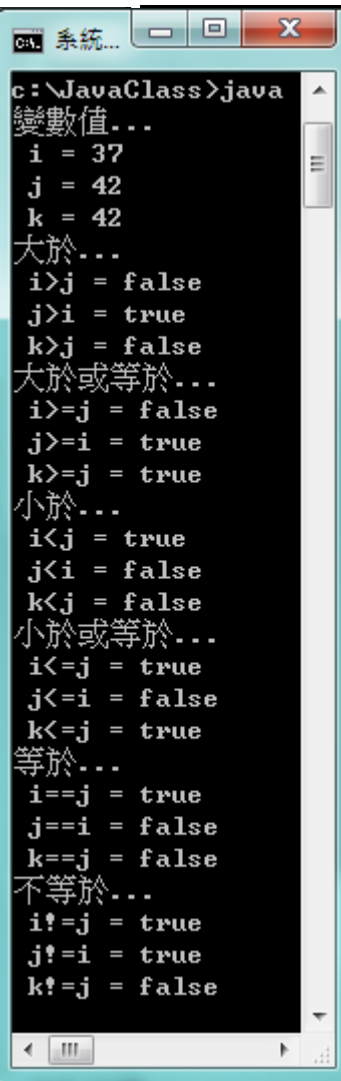
關係運算

運算子	說明	運算元	範例	運算子類別
==	等於	布林,數字, 字元,物件	a == b	雙元
!=	不等於		a != b	雙元
>	大於	數字,字元	a > b	雙元
<	小於	數字,字元	a < b	雙元
>=	大於等於	數字,字元	a >= b	雙元
<=	小於等於	數字,字元	a <= b	雙元
instanceof		物件,類別	S instanceof String	雙元

傳回值為布林值(true或false)

關係運算

```
01 public class RelationalDemo {
02     public static void main(String[] args) {
03         //宣告變數
04         int i = 37;
05         int j = 42;
06         int k = 42;
07         System.out.println("變數值...");
08         System.out.println("i="+i);
09         System.out.println("j="+j);
10         System.out.println("k="+k);
11         //大於
12         System.out.println("大於...");
13         System.out.println("i>j="+ (i>j));
14         System.out.println("j>i="+ (j>i));
15         System.out.println("k>j="+ (k>j));
16         //大於或等於
17         System.out.println("大於或等於...");
18         System.out.println("i>=j="+ (i>=j));
19         System.out.println("j>=i="+ (j>=i));
20         System.out.println("k>=j="+ (k>=j));
```



```
c:\JavaClass>java
變數值...
i = 37
j = 42
k = 42
大於...
i>j = false
j>i = true
k>j = false
大於或等於...
i>=j = false
j>=i = true
k>=j = true
小於...
i<j = true
j<i = false
k<j = false
小於或等於...
i<=j = true
j<=i = false
k<=j = true
等於...
i==j = false
j==i = false
k==j = true
不等於...
i!=j = true
j!=i = true
k!=j = false
```

```
//小於
System.out.println("小於...");
System.out.println("i<j="+ (i<j));
System.out.println("j<i="+ (j<i));
System.out.println("k<j="+ (k<j));
//小於或等於
System.out.println("小於或等於...");
System.out.println("i<=j="+ (i<=j));
System.out.println("j<=i="+ (j<=i));
System.out.println("k<=j="+ (k<=j));
//等於
System.out.println("等於...");
System.out.println("i==j="+ (i==j));
System.out.println("j==i="+ (j==i));
System.out.println("k==j="+ (k==j));
//不等於
System.out.println("不等於...");
System.out.println("i!=j="+ (i!=j));
System.out.println("j!=i="+ (j!=i));
System.out.println("k!=j="+ (k!=j));
```

邏輯運算

運算子	說明	範例	運算子類別
&	AND	a & b	雙元
 	OR	a b	雙元
^	XOR	a ^ b	雙元
!	NOT	! a	單元
(捷徑) Short-circuit Operator			
&&	AND	a && b	雙元
 	OR	a b	雙元

運算元為布林值(true或false)
傳回值為布林值

邏輯運算

■ 一般邏輯運算子 &, |, ^, !

AND	true	false
true	true	false
false	false	false

OR	true	false
true	true	true
false	true	false

XOR	true	false
true	false	true
false	true	false

NOT	
true	false
false	true

■ 捷徑邏輯運算子(short-circuit) &&, ||

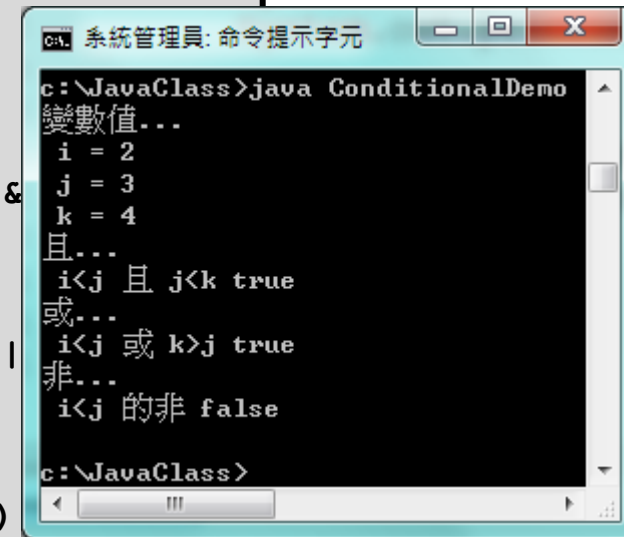
- 捷徑:當左式已經足以判斷整個運算的結果時,右式就不必做了,若左式不足以判斷整個運算的結果時,右式仍然要做
- 邏輯判斷會改變變數的值時,易產生非預期的結果

&&	X
true	X
false	false

	X
true	true
false	X

邏輯運算

```
01 public class ConditionalDemo {
02     public static void main(String[] args) {
03         int i = 2;
04         int j = 3;
05         int k = 4;
06         System.out.println("變數值...");
07         System.out.println(" i = " + i);
08         System.out.println(" j = " + j);
09         System.out.println(" k = " + k);
10         //&&運算
11         System.out.println("且...");
12         System.out.println(" i<j 且 j<k "+((i<j)&j<k));
13         //||運算
14         System.out.println("或...");
15         System.out.println(" i<j 或 k>j "+((i<j)||k>j));
16         //&&運算
17         System.out.println("非...");
18         System.out.println(" i<j 的非 "+(! (i<j)));
19     }
20 }
```



```
系統管理員: 命令提示字元
c:\JavaClass>java ConditionalDemo
變數值...
 i = 2
 j = 3
 k = 4
且...
 i<j 且 j<k true
或...
 i<j 或 k>j true
非...
 i<j 的非 false
c:\JavaClass>
```

捷徑邏輯運算

a=10; b=5

(a++ > 10) & (b-- < 5) \Rightarrow a=11; b=4

(a++ > 10) && (b-- < 5) \Rightarrow a=11; b=5

if((10 / i) > 2) {...} int i = 0; \Rightarrow throws [ArithmeticException](#)

if(i == 0 || (10/i) > 2) {...} \Rightarrow true

if(i != 0 && (10/i) > 2) {...} \Rightarrow false

捷徑邏輯運算

```
if((10 / friend) >=1) {  friend== 0;  $\Rightarrow$  throws ArithmeticException  
    System.out.println(" Happy! ");  
} else {  
    System.out.println(" Sad! ");  
}
```

```
if( friend != 0 && (10/friend) >=1) {  
    System.out.println(" Happy! ");  
} else {  
    System.out.println(" Sad! ");  
}
```

```
if( friend == 0 || (10/friend) >=1) {  
    System.out.println(" Happy! ");  
} else {  
    System.out.println(" Sad! ");  
}
```

位元邏輯運算

■ 位元運算子 $\&$, $|$, \wedge , \sim

&	1	0
1	1	0
0	0	0

	1	0
1	1	1
0	1	0

\wedge	1	0
1	0	1
0	1	0

~	
1	0
0	1

運算元為數字或字元
傳回值為數字或字元

10 & 35 \rightarrow 2

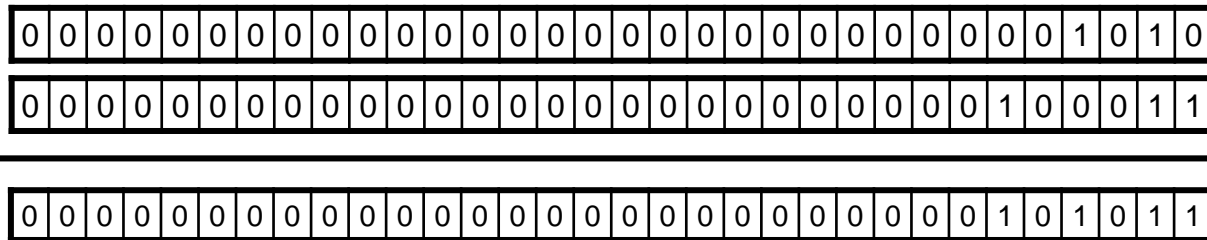
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

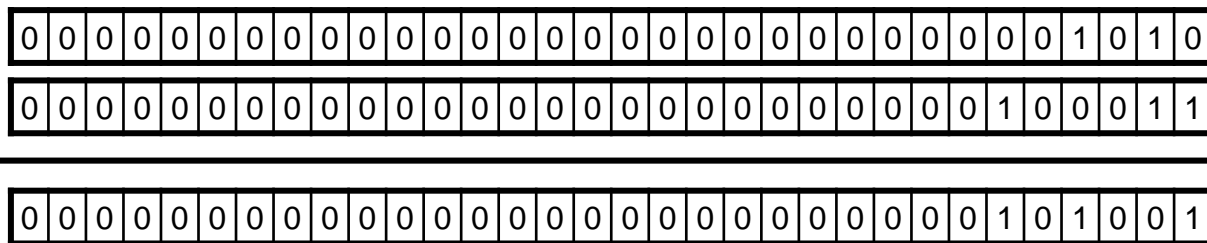
[illegible]

位元邏輯運算

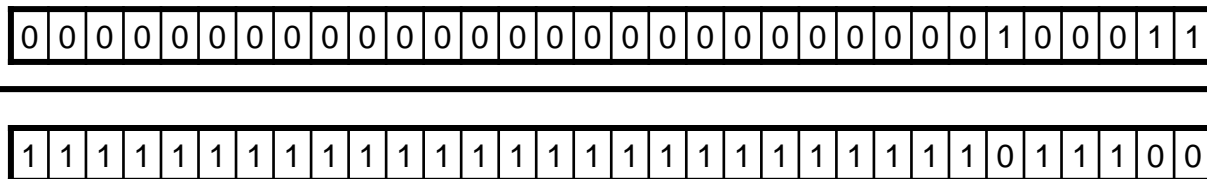
$10 \mid 35 \longrightarrow 43$



$10 \wedge 35 \longrightarrow 41$

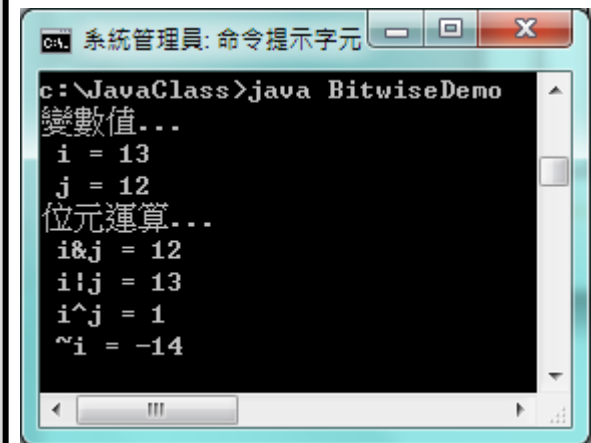


$\sim 35 \longrightarrow -36$



位元邏輯運算

```
01 public class BitwiseDemo {
02     public static void main(String[] args) {
03         int i = 13;
04         int j = 12;
05
06         System.out.println("變數值...");
07         System.out.println(" i = " + i);
08         System.out.println(" j = " + j);
09
10         System.out.println("位元運算...");
11         System.out.println(" i&j = " + (i&j));
12         System.out.println(" i|j = " + (i|j));
13         System.out.println(" i^j = " + (i^j));
14         System.out.println(" ~i = " + (~i));
15     }
16 }
```



```
C:\JavaClass>java BitwiseDemo
變數值...
 i = 13
 j = 12
位元運算...
 i&j = 12
 i|j = 13
 i^j = 1
 ~i = -14
```

位移運算

- 左移運算

$op1 \ll op2$ $op1$ 的位元左移 $op2$ 個單位, 右邊補上0

- 右移運算

$op1 \gg op2$ $op1$ 的位元右移 $op2$ 個單位, 左邊補上最左邊的位元值

- 無向右移運算

$op1 \ggg op2$ $op1$ 的位元右移 $op2$ 個單位, 左邊補上0

位移運算

1025

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

$1025 \gg 3 = 1025 / 2^3 = 128$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

$1025 \ggg 3 = 1025 / 2^3 = 128$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

$1025 \ll 3 = 1025 * 2^3 = 8200$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

位移運算

-1025

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

$$-1025 \gg 3 = -1025 / 2^3 = -129$$

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---


$-1025 \ggg 3 = 536870783$

0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

$$-1025 \ll 3 = -1025 * 2^3 = -8200$$
[illegible]

位移運算

```
01 public class ShiftDemo {
02     public static void main(String[] args) {
03         int i = 1;
04         System.out.println("變數值...");
05         System.out.println(" i = " + i);
06
07         System.out.println("位移運算...");
08         System.out.println(" i<<1 = " + (i<<1));
09         System.out.println(" i<<2 = " + (i<<2));
10         System.out.println(" i<<3 = " + (i<<3));
11     }
12 }
```



The screenshot shows a Windows command prompt window with the title '系統管理員: 命令提示字元'. The command 'c:\JavaClass>java ShiftDemo' has been executed. The output of the program is displayed as follows:

```
c:\JavaClass>java ShiftDemo
變數值...
i = 1
位移運算...
i<<1 = 2
i<<2 = 4
i<<3 = 8
c:\JavaClass>
```

指定運算子(Assignment Operator)

運算子	說明	範例	類別
=	基本指定	$a = b$	單元
+=	加法指定	$a += b \rightarrow a = a + b$	雙元
-=	減法指定	$a -= b \rightarrow a = a - b$	雙元
*=	乘法指定	$a *= b \rightarrow a = a * b$	雙元
/=	除法指定	$a /= b \rightarrow a = a / b$	雙元
%=	餘數指定	$a \% = b \rightarrow a = a \% b$	雙元
&=	AND 指定	$a \& = b \rightarrow a = a \& b$	雙元
=	OR 指定	$a = b \rightarrow a = a b$	雙元
^=	XOR 指定	$a \wedge = b \rightarrow a = a \wedge b$	雙元
>>=	算數右移指定	$a >> = b \rightarrow a = a >> b$	雙元
<<=	算數左移指定	$a << = b \rightarrow a = a << b$	雙元
>>>=	邏輯右移指定	$a >>> = b \rightarrow a = a >>> b$	雙元

指定運算子

■ 指定運算子

byte b = 20

❑ b = b>>3 **Compile Error**

❑ b >>= 3 b = (byte) (b >>3)
 b = 2

■ 減次運算

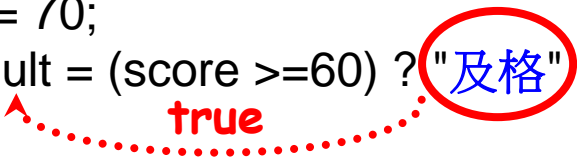
❑ $8 \gg 33 = 8 \gg (33\%32) = 8 \gg 1 = 4$

條件運算子(Conditional Operator)

■ 條件(三元)運算子

- 概念類似if-else 條件敘述
- **X = (布林運算式) ? true-value : false-value**
- 當(布林運算式)的回傳值為 **true** 的時候，會進行冒號“：”左邊的敘述(**true-value**)，反之則進行冒號右邊的敘述(**false-value**)給 **X**。

```
int score = 70;  
String result = (score >= 60) ? "及格" : "不及格";  
System.out.println(result);
```



執行結果：及格

運算子的優先權與結合性

優先權等級	運算子	說明	結合性
1	! ~ ++ -- - +	邏輯的「非」 位元的「非」 累加 漸減 負號 正號	由右至左
2	(type)	定型運算子	由右至左
3	* / %	乘號 除號 求餘數	由左至右
4	+ -	加號 減號	由左至右
5	<< >> >>>	位元左移 位元右移 位元無號右移	由左至右

運算子的優先權與結合性

優先權等級	運算子	說明	結合性
6	< <= > >=	是否小於 是否小於等於 是否大於 是否大於等於	由左至右
7	== !=	是否等於 是否不等於	由左至右
8	&	位元的「且」	由左至右
9	^	位元的「互斥」	由左至右
10		位元的「或」	由左至右
11	&&	邏輯的「且」	由左至右
12		邏輯的「或」	由左至右
13	?:	條件運算子	由右至左
14	= +=, -=, *=, /=, %= &=, =, ^=, <<=, >>=, >>>=	指定運算子 指定類運算子	由右至左

運算子的優先權範例

- $x = 5 * 3 < 20 \ \& \ 3 + 7 > 9 - 1 \parallel 20 \geq 20 - 30 \ \&\& \text{false}$
- 先算 $5*3$ 、 $3+7$ 、 $9-1$ 、 $20-30$ 的結果
- $15 < 20 \ \& \ 10 > 8 \parallel 20 \geq -10 \ \&\& \text{false}$
- 再算 $15 < 20$ 、 $10 > 8$ 、 $20 \geq -10$ 的結果
- $\text{true} \ \& \ \text{true} \parallel \text{true} \ \&\& \ \text{false}$
- 再算 $\text{true} \ \& \ \text{true}$ 的結果
- $\text{true} \parallel \text{true} \ \&\& \ \text{false}$
- **$x = \text{true}$**