

Operator

運算子

Java Fundamental



Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

Operators

- ◆ 運算子可對一個、兩個或三個運算元執行動作
- ◆ 依照所需的運算元數目，可分為：
 - 一元運算子(unary operator)，例：`a++`、`--a`
 - 二元運算子(binary operator)，例：`a = 2`
 - 三元運算子(ternary operator)，例：`(x>0) ? x-1 : (x+1)`
- ◆ 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
3 * 4
運算元

Java 運算子 – 說明

- ◆ Java「運算式」是由「運算元」和「運算子」組成。

Expression = Operands + Operators

運算式 = 運算元 + 運算子

- ◆ Java運算式的範例，如下所示：

➤ $a + b - 1$

➤ $a \geq b$

➤ $a > b \ \&\& \ a > 1$

運算子
3 * 4
運算元

- ◆ 變數a、b和數值1都是運算元，「+」、「-」、「>=」、「>」和「&&」為運算子。

Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

算數運算子 (Arithmetic Operator)

運算子	說明	範例	類別
+	正號	+2 ➔ 取得正數 2	單元
	加法	2 + 1 = 3	雙元
-	負號	-2 ➔ 取得負數 2	單元
	減法	2 - 1 = 1	雙元
*	乘法	2 * 3 = 6	雙元
/	除法	4 / 2 = 2	雙元
%	餘數	3 % 2 = 1，求得 3 除以 2 的餘數	雙元
++	遞增	前序遞增：int A = 0；A = ++A ；表示 A 先加上 1 之後再指派給A所以A=1。	單元
		後序遞增：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	前序遞減：int A = 0；A = --A ；表示A先減1之後再指派給A所以 A = - 1	單元
		後序遞減：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。	

算數運算子(Arithmetic Operator)

```
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 }
```



系統管理員: 命令提示字元

```
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>
```

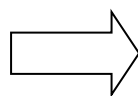

算數運算子 後置運算

◆ $X = X + 1;$ \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

- $\text{int var} = \text{age}++;$



$\text{int var} = \text{age};$
 $\text{age} = \text{age} + 1;$

```
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>
```

算數運算子 前置運算

◆ 前置遞增、前置遞減

- `int var = ++age;` \Rightarrow `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 }
```



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

算數運算子 遞增遞減運算

```
x = 17;
```

```
System.out.println("Before Prefix X = " + ++x);
```

```
System.out.println("After Prefix X = " + x);
```

```
x = 13;
```

```
System.out.println("Before Postfix X = " + x--);
```

```
System.out.println("After Postfix X = " + x);
```

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, 程式會自動promote成int來運算。
 - short a = 1
 - short b = 2
 - short c = (short) (a+b);

文字串接 (String Concatenation)

◆ String Concatenation

➤ + 中只要有一個運算元是String,另一個運算元也會被轉成String,結果為字串串接。

- System.out.println(3+4+"str"); //7str
- System.out.println("str"+3+4); //str34

```
public class Concatenate{  
    public static void main(String[] args){  
        String str1 = "Java ";  
        String str2 = "Operators";  
        int i = 1;  
        double d = 1.2;  
        String str3 = str1 + i + d;  
        System.out.println(str1.concat(str2));  
        System.out.println(str3);  
    }  
}
```

-----輸出-----
Java Operators
Java 11.2

會將 i 與 d 數字轉成字串後再串接

將 str1 與 str2 兩個字串串接起來

算數運算子 (Arithmetic Operator)

◆ 整數 / 整數(取商數)

➤ $10 / 3 = 3 \cdots 1$ 餘數捨棄

➤ $10 / 0 = \text{throws ArithmeticException}$ 分母不可為0

◆ 除數或被除數至少有一為浮點數

➤ $10.0 / 3 = 3.33333334$ 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

算數運算子 (Arithmetic Operator)

◆ 整數 % 整數(取餘數)

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

- $10.0 \% 0.0 = \text{NaN}$
- $0.0 \% 0.0 = \text{NaN}$
- $(5/0.0) \% 2.0 = \text{NaN}$ ($\infty \% 2.0$)
- $8.0 \% 3 = 2.0$
- $6.5 \% 3.2 = 0.1$ (商數須為整數)
- $2.0 \% (5/0.0) = 2.0$ ($2.0 \% \infty$)

算數運算子 (Arithmetic Operator)

◆ 浮點數中額外定義

- $+\infty$: 一個正的floating point, 大於所能表示的最大值。
- $-\infty$: 一個負的floating point, 小於所能表示的最大負值。
- 0 : 一個正的floating point, 微細於所能表示的最細值。
- -0 : 一個負的floating point, 微細於所能表示的最細值。
- NaN : 浮點數做不適當運算, ex: 0.0/0.0 。

Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

關係運算子 (Relational Operator)

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

傳回值為布林值(true或false)

關係運算子 (Relational Operator)

- ◆ 關係運算子會得到true或false的答案。
- ◆ 浮點數不是精準數，建議不要將浮點數用在「==」上。

```
1. public class Compare{
2.     public static void main(String[] args){
3.         int x = 50, y = 50;
4.         boolean b1 = x > y;
5.         boolean b2 = x == y;
6.         boolean b3 = x != y;
7.         System.out.println(b1);
8.         System.out.println(b2);
9.         System.out.println(b3);
10.    }
11. }
```

比較 **x** 與 **y** 是否相等，是的話
回傳 **true**；否則回傳 **false**

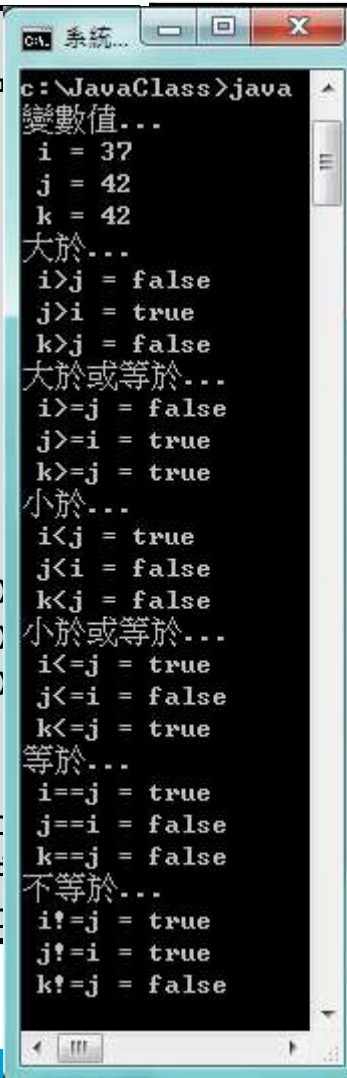
x 與 **y** 的值都是 **50**，所以回
傳 **false**

-----輸出-----

false
true
false

關係運算子 (Relational Operator)

```
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));
```



```
CA. 系統...
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 = true
j==i = false
k==j = false
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));
```

Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

邏輯運算子 (Logical operator)

- ◆ 運算元為布林值(true或false)
- ◆ 傳回值為布林值

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 Logical Operator			
&&	AND	a && b	雙元
	OR	a b	雙元

邏輯運算子 (Logical operator)

```
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>
```

捷徑邏輯運算子 (Short-circuit Logical Operator)

◆ 一般邏輯運算子 &, |, ^, !

AND	true	false
true	true	Flase
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) &&, ||

- 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。

&&	X
true	X
false	false

	X
true	true
false	X

捷徑邏輯運算子 (Short-circuit Logical Operator)

◆ `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`

邏輯運算子 (Logical operator)

◆ XOR (^) :

- XOR與OR運算的結果大致相同。
- 只有當所有條件式都是True時，XOR結果是False。

◆ NOT (!) :

- True碰到NOT符號會得到False；False碰到NOT符號會變成True。

XOR_NOT.java

```
1. class XOR_NOT{
2.     public static void main(String[] args){
3.         boolean furniture=true, decoration=true;
4.
5.         System.out.println(furniture ^ decoration);
6.         System.out.println(!(furniture ^ decoration));
7.     }
8. }
```

true ^ true 得到結果為 false

furniture ^ decoration 結果為 false，
前面加上「!」運算符號後會得到 true

-----輸出-----

false
true

Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

位元運算子(Bit Operator)

◆ 將數字拿來作一個bit一個bit的運算。

位元運算子	用法	功能
&	$X \& Y$	X 與 Y 以 bit 為單位，做 AND 運算
	$X Y$	X 與 Y 以 bit 為單位，做 OR 運算
~	$\sim X$	X 做補數運算
^	$X \wedge Y$	X 與 Y 以 bit 為單位，做 XOR 運算
>>	$X \gg Y$	將 X 向右移動 Y 個位元
<<	$X \ll Y$	將 X 向左移動 Y 個位元
>>>	$X \ggg Y$	將 X 向右移動 Y 個位元，不含正負號

位元運算子(Bit Operator)

- ◆ 位元運算與邏輯運算的觀念一樣，只不過位元運算要先將整數轉成2進位數值後方能運算。

位元運算

$1 \& 1 \rightarrow 1$

$1 | 1 \rightarrow 1$

$1 \wedge 1 \rightarrow 0$

$\sim 1 \rightarrow 0$

邏輯運算(對照組)

$\text{true} \& \text{true} \rightarrow \text{true}$

$\text{true} | \text{true} \rightarrow \text{true}$

$\text{true} \wedge \text{true} \rightarrow \text{false}$

$!\text{true} \rightarrow \text{false}$

位元運算子(Bit Operator)

◆ 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

\sim	
1	0
0	1

運算元為數字或字元
傳回值為數字或字元

8 | 33 \longrightarrow 41

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

位元運算子(Bit Operator)

```
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  
11         System.out.println("位元運算...");  
12         System.out.println(" i&j = " + (i&j));  
13         System.out.println(" i|j = " + (i|j));  
14         System.out.println(" i^j = " + (i^j));  
15         System.out.println(" ~i = " + (~i));  
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
```

位元運算子(Bit Operator)

◆ Java整數資料的表示(以16位元為例): int a=3,b=-8

位元	15	14	13	12	11	10	9	8	7	6	6	4	3	2	1	0	值
a=	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3
b=	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	-8
代表值	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	

◆ 第16位元(Bit 15)若為0表該值為正，1表該值為負。

◆ a=0000 0000 0000 0011，以二進位表示為 $2+1=3$ 。

◆ b=1111 1111 1111 1000， $-32768 + 16384 + 8192 + 4096 + 1024 + 512 + 256 + 128 + 64 + 32 + 16 + 8 = -8$ ，另一算法為若0的位數較少，只加0的位元之代表值及加1後再加上負號，即 $-(4+2+1+1)=-8$ 。

◆ 整數最大值32767(0111 1111 1111 1111)，
整數最小值-32768(1000 0000 0000 0000)。

位元運算子(Bit Operator)

◆ ~補數運算子類似邏輯運算子之not，將0變為1，將1變為0 真值表如下：

a	b	~a	~b
0	0	1	1
1	0	0	1
0	1	1	0
1	1	0	0

➤ 例如: a=3 → 0000 0000 0000 0011

c=~a → 1111 1111 1111 1100 → c=-4

位元位移運算 (Bit Operator Shift)

- ◆ 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(正負號)右移運算($op1 \ggg op2$)

- $op1$ 的位元右移 $op2$ 個單位, 左邊補上0

位元位移運算 (Bit Operator Shift)

◆ 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	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

◆ $1025 \gg 3 = 1025 / 23 = 128$

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
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

◆ $1025 \ggg 3 = 1025 / 23 = 128$

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
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

◆ $1025 \ll 3 = 1025 * 23 = 8200$

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

位元位移運算 (Bit Operator Shift)

◆ -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	0	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

◆ $-1025 \ll 3 = -1025 * 2^3 = -8200$

[illegible]

位元位移運算 (Bit Operator Shift)

```
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 titled "系統管理員: 命令提示字元". The command executed is `c:\JavaClass>java ShiftDemo`. The output of the program is 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>
```

Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

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(Assignment Operator)

◆ 指定運算子

byte b = 20

➤ b >>= 3 **Compile Error** b = (byte) (b >>3)
➤ b = b>>3 b = 2

◆ 減次運算

➤ 8 >> (33%32) = 8 >> 1 = 4

Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

三元運算子 (Ternary Operator)

◆ 三元運算子(Ternary Operator)

➤ 概念類似if-else 條件敘述

➤ 使用語法

$X = (\text{布林運算式}) ? \text{true-value} : \text{false-value}$

➤ 當運算式回傳值為 **true** 的時候，會進行冒號 左邊的敘述 (**true-value**) 給 X。

➤ 當運算式回傳值為 **false** 的時候，會進行冒號 右邊的敘述 (**false-value**) 給 X。

```
String s = "";  
int i = 0, j = 1;  
s = (i < j) ? "正確" : "錯誤";  
↑  
true
```

System.out.println("s=" + s); 執行結果：s=正確

Content

- ◆ 運算式, 運算元, 運算子介紹
- ◆ 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)
- ◆ 三元(條件)運算子 (Ternary (Conditional) Operator)
- ◆ 運算子優先順序

運算子的優先權與結合性

- ◆ 下圖依照優先序高 → 低排列(上面優先序高於下面)。
- ◆ 當優先順序相同時就必須依照結合性來運算。

[] 陣列索引 () 方法呼叫 .成員取用

++ 遞增 -- 遞減 + 正號 - 負號 ~ 位元NOT ! 邏輯NOT (type) 轉型 new 建立實體物件

* 乘 / 除 % 取餘數

+ 加 - 減 + 文字串接

<< 有號左移 >> 有號右移 >>> 無號右移

> 大於 >= 大於等於 < 小於 <= 小於等於

== 相等於 != 不等於

& AND運算 | OR運算 ^ XOR運算

&& 邏輯AND運算 || 邏輯OR運算

?: 條件運算

= 指派運算

右結合

右結合

右結合

運算子的優先權與結合性

◆ 運算子優先權

高



低

運算子分類	運算子
分隔運算子	() [] {} , .
單一運算元	++ -- + (正號) - (負號) ~ !
建造 / 轉型運算子	new (type)expr
乘除	* / %
加減	+ -
位移運算子	<< >> >>>
關係運算子	< <= > >= instanceof
相等運算子	== !=
位元 AND	&
位元 XOR	^
位元 OR	
邏輯 AND	&&
邏輯 OR	
條件運算子	? :
指定運算子	= += -= *= /= %= &= ^= = >>= <<= >>>=

運算子的優先權與結合性

1. `x = 5 * 3 < 20 & 3 + 7 > 9 - 1 || 20 >= 20 - 30 && false`
2. 先算 `5*3`、`3+7`、`9-1`、`20-30` 的結果
3. `15 < 20 & 10 > 8 || 20 >= -10 && false`
4. 再算 `15 < 20`、`10 > 8`、`20 >= -10` 的結果
5. `true & true || true && false`
6. 再算 `true & true` 的結果
7. `true || true && false`
8. **`x = true`**

Q & A