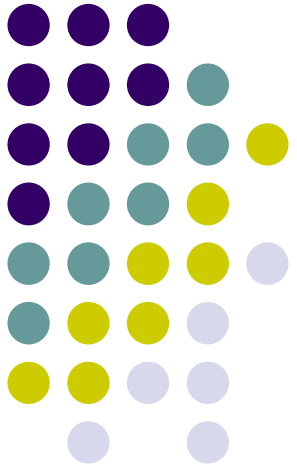
第四章
運算子、運算式與敘述



認識運算式與運算子

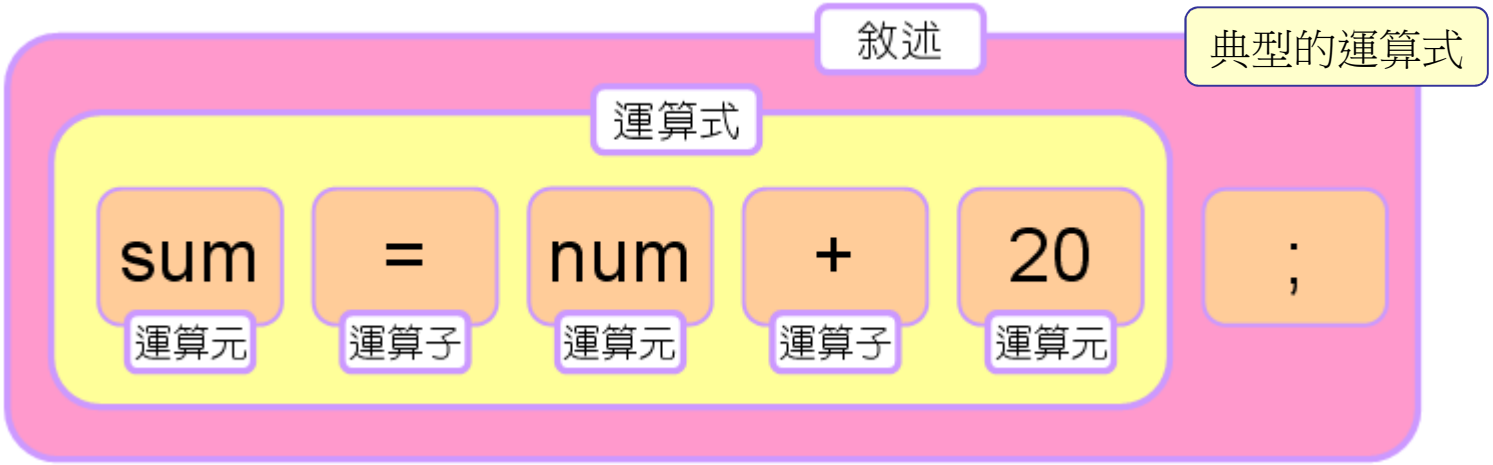
學習各種常用的運算子

認識運算子的優先順序

學習如何進行運算式之資料型態的轉換

認識運算式

- 運算式由運算元（operand）與運算子（operator）組成
- 運算元可以是變數或是常數
- 運算子就是數學上的運算符號
  - 如「+」、「-」、「*」、「/」等

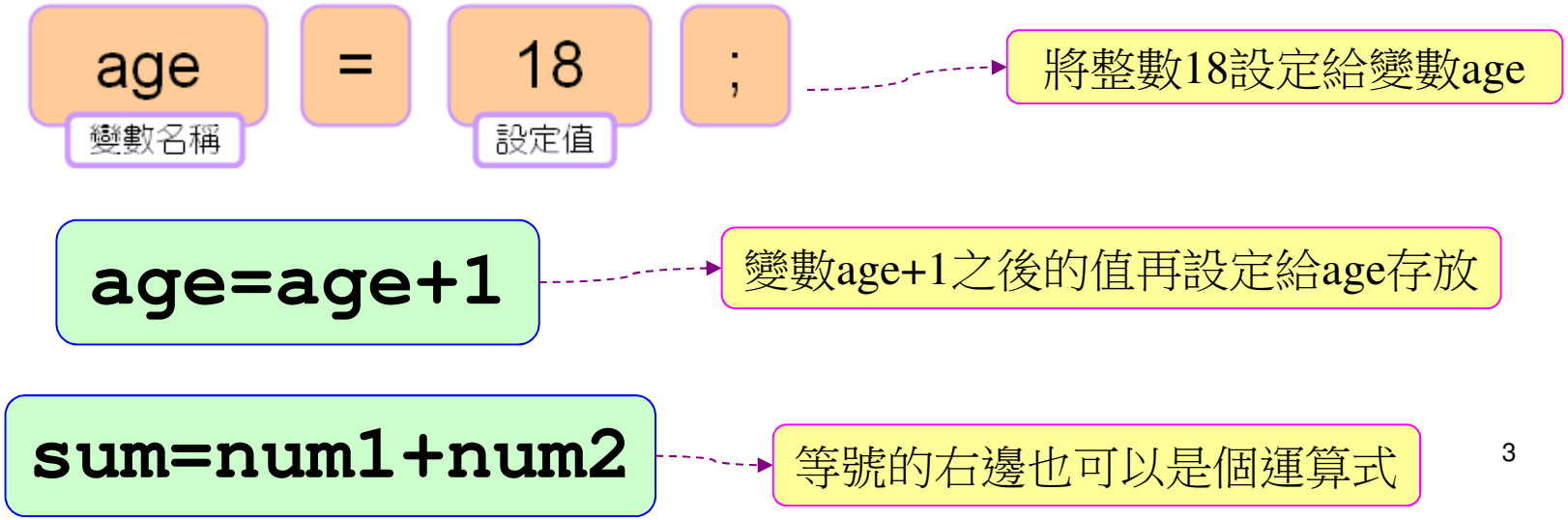


設定運算子(1/2)

- 為各種不同資料型態的變數設值，可使用設定運算子

設定運算子	意義
=	設定

- 等號（=）並不是「等於」，而是「設定」



設定運算子 (2/2)

- 下面的範例示範設定運算子的使用：

```
01  // app4_1, 設定運算子「=」
02  public class app4_1
03  {
04     public static void main(String args[])
05     {
06        int age=18;      // 宣告整數變數 age,並設值為 18
07
08        System.out.println("before compute, age="+age);  // 印出 age 的值
09        age=age+1;       // 將 age 加 1 後再設定給 age 存放
10        System.out.println("after compute, age="+age); // 印出計算後 age 的值
11     }
12  }
```

```
/* app4_1 OUTPUT-------

before compute, age=18
after compute, age=19
----------------------*/
```

一元運算子 (1/2)

- 下面的敘述是由一元運算子與單一個運算元組成

```
+6;          // 表示正6
~a;          // 表示取a的1補數
x=-y;        // 表示負y的值設定給變數x存放
!a;          // a的NOT運算，若a為0，則!a為1，若a不為0，則!a為0
```

- 一元運算子的成員

一元運算子	意義
+	正號
-	負號
!	NOT，否定
~	取 1 的補數

一元運算子 (2/2)

- 「~」與「!」運算子的範例：

```
01  // app4_2, 一元運算子「~」與「!」
02  public class app4_2
03  {
04     public static void main(String args[])
05     {
06        byte a=Byte.MIN_VALUE;     // 宣告變數 a,並設為該型態之最小值
07        boolean b=true;            // 宣告 boolean 變數 b,並設為 true
08
09        System.out.println("a="+a+",~a="+(~a));  // 印出 a 與~a 的值
10        System.out.println("b="+b+",!b="+(!b));  // 印出 b 與!b 的值
11     }
12  }
```

```
/* app4_2 OUTPUT----

a=-128,~a=127
b=true,!b=false
--------------------*/
```

算術運算子 (1/3)

- 算術運算子在數學上經常會使用到
- 算術運算子的成員：

算術運算子	意義
+	加法
-	減法
*	乘法
/	除法
%	取餘數

- 加法運算子「+」可將前後兩個運算元做加法運算

```
6+2;                    // 計算6+2
b=a+15                  // 將a的值加15之後，再設定給變數b存放
sum=a+b+c               // 將a,b與c的值相加後，設定給變數sum存放
```

算術運算子 (2/3)

- 減法運算子「-」可將前後兩個運算元做減法運算

```
age=age-1;          // 計算age-1之後，再將其結果設定給age存放
c=a-b;              // 計算a-b之後，再設定給c存放
54-12;              // 計算54-12的值
```

- 乘法運算子「*」可將前後兩個運算元相乘

```
b=c*3;              // 計算c*3之後，再將其結果設定給b存放
a=a*a;              // 計算a*a之後，再設定給a存放
17*5;               // 計算17*5的值
```

算數運算子 (3/3)

- 除法運算子「/」可將前面的運算元後除以後面的運算元

```
b=a/6;          // 計算a/6之後，再將其結果設定給b存放
d=c/d;          // 計算c/d之後，再設定給d存放
3/8;            // 計算3/8的值
```

- 餘數運算子「%」：取出二數相除後的餘數

```
age=age%5;      // 計算age/5的餘數，再把計算結果給age存放
c=a%b;          // 計算a/b的餘數，然後把計算結果給c存放
48%7;           // 運算48%7的值
```

關係運算子與 if 敘述 (1/2)

- if 敘述的格式如下：

if 敘述的格式

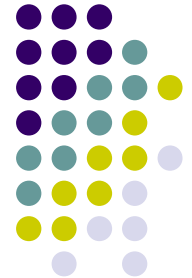
```
if(條件判斷)
    敘述;
```

- 如下面的程式片段：

```
if (x<0)
    System.out.println("x的值小於0");
```

關係運算子的成員

關係運算子	意義
>	大於
<	小於
>=	大於等於
<=	小於等於
==	等於
!=	不等於



關係運算子與 if 敘述 (2/2)

- 關係運算子的使用範例：

```
01  // app4_3, 關係運算子
02  public class app4_3
03  {
04     public static void main(String args[])
05     {
06        if (9>4)          // 判斷 9>4 是否成立
07           System.out.println("9>4 成立");                    // 印出字串
08
09        if (true)         // 判斷 true 是否成立
10           System.out.println("此行會被執行--true");          // 印出字串
11
12        if (false)        // 判斷 false 是否成立
13           System.out.println("此行會被執行--false");         // 印出字串
14
15        if (5==7)         // 判斷 5 是否等於 7
16           System.out.println("5==7 成立");                   // 印出字串
17     }
18  }
```

```
/* app4_3 OUTPUT----
9>4 成立
此行會被執行--true
--------------------*/
```

遞增與遞減運算子

- 遞增與遞減運算子的成員：

遞增與遞減運算子	意義
++	遞增，變數值加 1
--	遞減，變數值減 1

- 想讓變數a加上1，其敘述如下

```
a=a+1;
```

a加1後再設定給a存放，簡潔的寫法為a++;

本程式比較a++與++b的不同

```
01  // app4_4, 遞增運算子「++」
02  public class app4_4
03  {
04     public static void main(String args[])
05     {
06        int a=5,b=5;
07
08        System.out.print("a="+a);     // 印出 a
09        System.out.println(",a++="+(a++)+",a="+a);  // 印出 a++
10        System.out.print("b="+b);     // 印出 b
11        System.out.println(",++b="+(++b)+",b="+b);  // 印出++b
12     }
13  }
```

```
/* app4_4 OUTPUT----
a=5,a++=5,a=6
b=5,++b=6,b=6
--------------------*/
```

邏輯運算子 (1/2)

- 邏輯運算子與真值表：

邏輯運算子的成員

邏輯運算子	意義
&&	AND，且
||	OR，或

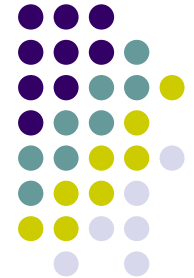
AND與OR真值表

AND	T	F
T	T	F
F	F	F

OR	T	F
T	T	T
F	T	F

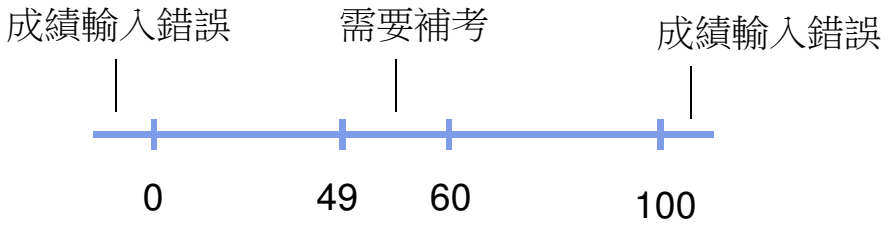
- 邏輯運算子的使用範例：

```
(1) a>0 && b>0     // 兩個運算元皆為真，運算結果才為真
(2) a>0 || b>0     // 兩個運算元只要一個為真，運算結果就為真
```



邏輯運算子(2/2)

- 邏輯運算子應用在 if 敘述中 ：




```
01  // app4_5, 邏輯運算子
02  public class app4_5
03  {
04     public static void main(String args[])
05     {
06        int a=53;
07
08        if ((a<0) || (a>100))
09           System.out.println("Input error!!");    // 成績輸入錯誤
10        if ((a<60) && (a>49))
11           System.out.println("Make up exam!!");   // 需要補考
12     }
13  }
```

```
/* app4_5 OUTPUT----

Make up exam!!
--------------------*/
```

括號運算子

- 括號運算子() 用來處理運算式的優先順序：

括號運算子	意義
()	提高括號中運算式的優先順序

- 以一個加減乘除式子為例：

```
12-2*6/4+1                    // 未加括號的運算式
```

運算結果為10

- 想分別計算12-2*6及4+1之後再將兩數相除則成為

```
(12-2*6)/(4+1)                // 加上括號的運算式
```

運算結果為0

運算子的優先順序列表

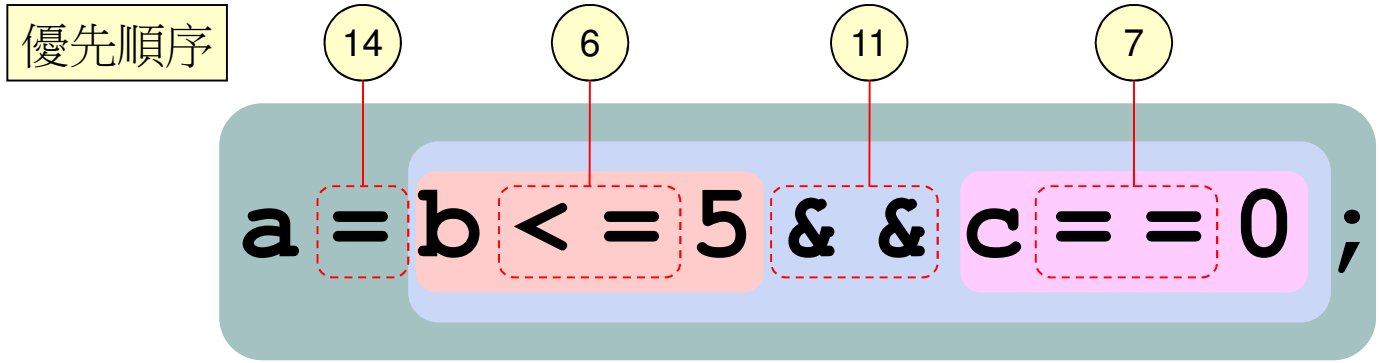
表 4.2.1　運算子的優先順序

優先順序	運算子	類別	結合性
1	()	括號運算子	由左至右
1	[]	方括號運算子	由左至右
2	!、+（正號）、-（負號）	一元運算子	由右至左
2	~	位元邏輯運算子	由右至左
2	++、--	遞增與遞減運算子	由右至左
3	*、/、%	算術運算子	由左至右
4	+、-	算術運算子	由左至右
5	<<、>>	位元左移、右移運算子	由左至右
6	>、>=、<、<=	關係運算子	由左至右
7	==、!=	關係運算子	由左至右
8	&（位元運算的 AND）	位元邏輯運算子	由左至右
9	^（位元運算的 XOR）	位元邏輯運算子	由左至右
10	|（位元運算的 OR）	位元邏輯運算子	由左至右
11	&&	邏輯運算子	由左至右
12	||	邏輯運算子	由左至右
13	?:	條件運算子	由右至左
14	=	設定運算子	由右至左

數字愈小表示
優先順序愈高

運算子的優先順序

- 運算子優先順序的範例：



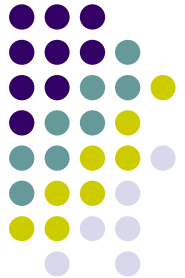
1. 先計算 b<=5 (<=的優先順序為6)
2. 再計算 c==0 (==的優先順序為7)
3. 然後進行&&運算 (&&的優先順序為11)
4. 最後再把運算結果設給變數 a 存放 (= 的優先順序為14)

結合性

- 結合性是指相同優先順序之運算子的執行順序
- 算術運算子的結合性為「由左至右」

```
a=b+d/3*6;      //  結合性可以決定運算子的處理順序
```

d會先除以3再乘以6
得到的結果加上b後，
將整個值給a存放



運算式的組成

- 運算式是由常數、變數或是其它運算元與運算子所組合而成
- 下面的例子，均是屬於運算式

```
-18                 // 由一元運算子「-」與常數18組成
sum+6               // 由變數sum、算術運算子與常數6組成
a+b-c/(d*3-9)       // 由變數、常數與算術運算子組成的運算式
```

簡潔的運算式 (1/2)

- 下表為簡潔的運算式與使用說明：

運算子	範例用法	說明	意義
+=	a+=b	a+b 的值存放到 a 中	a=a+b
-=	a-=b	a-b 的值存放到 a 中	a=a-b
*=	a*=b	a*b 的值存放到 a 中	a=a*b
/=	a/=b	a/b 的值存放到 a 中	a=a/b
%=	a%=b	a%b 的值存放到 a 中	a=a%b

- 簡潔寫法的運算式範例：

```
i++          // 相當於 i=i+1
a-=3         // 相當於 a=a-3
b%=c         // 相當於 b=b%c
a/=b--       // 相當於計算 a=a/b 之後，再計算 b--
```

簡潔的運算式 (2/2)

- 使用簡潔運算式的範例：

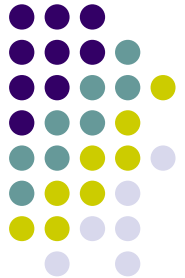
```
01  // app4_6, 簡潔運算式
02  public class app4_6
03  {
04     public static void main(String args[])
05     {
06        int a=2,b=5;
07
08        System.out.println("計算 a+=b 前, a="+a+" ,b="+b);
09        a+=b;       // 計算 a+=b 的值，此式相同於 a=a+b
10        System.out.println("計算 a+=b 後, a="+a+" ,b="+b);
11     }
12  }
                /* app4_6 OUTPUT---------

                計算 a+=b 前, a=2 ,b=5
                計算 a+=b 後, a=7 ,b=5
                -------------------------*/
```

運算式的資料型態轉換 (1/4)

- Java處理型態轉換的規則：

1. 佔用位元組較少的轉換成位元組較多的型態。
2. 字元型態會轉換成 short 型態（字元會取其 unicode 碼）。
3. short 型態（2 bytes）遇上 int 型態（4 bytes），會轉換成 int 型態。
4. int 型態會轉換成 float 型態。
5. 運算式中的若某個運算元的型態為 double，則另一個運算元也會轉換成 double 型態。
6. 布林型態不能轉換成其它的型態。



運算式的資料型態轉換 (2/4)

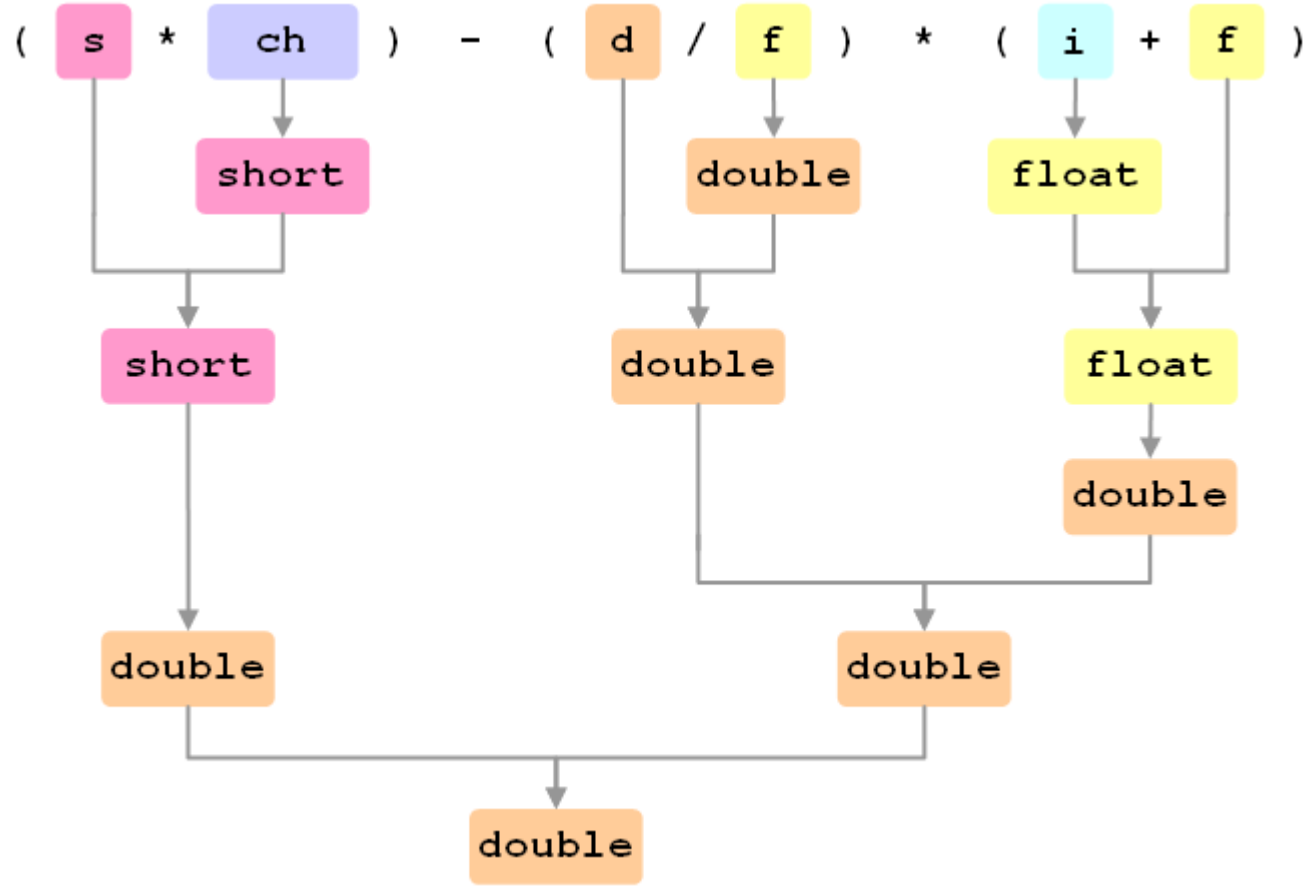
- 型態轉換的範例：

```
01  // app4_7,運算式的型態轉換
02  public class app4_7
03  {
04     public static void main(String args[])
05     {
06        char ch='X';
07        short s=-5;
08        int i=6;
09        float f=9.7f;
10        double d=1.76;
11
12        System.out.print("(s*ch)-(d/f)*(i+f)=");  // 印出結果
13        System.out.println((s*ch)-(d/f)*(i+f));
14     }
15  }
```

```
/* app4_7 OUTPUT--------------------
(s*ch)-(d/f)*(i+f)=-442.8486598152212
-------------------------------------*/
```

運算式的資料型態轉換 (3/4)

- 下圖為app4_7內變數的資料型態轉換過程解說：



運算式的資料型態轉換 (4/4)

- 下圖為app4_7運算式的運算過程：