

資料型態、變數與運算式

人工智慧與無線感應設備開發專班

湜憶電腦知訊顧問股份有限公司

馬傳義

前言

■ 電腦所要處理的東西稱為資料（Data）。

例如：

貨品的名稱、單價、數量、...等。

■ 這些資料必須先儲存於電腦記憶體中，然後電腦中央處理器（CPU）才能進行各種運算。

■ 不同的資料有其不同意義，有些是文字符號，有些是數值，數值可以直接用於算術運算，因此必須有所區別，稱之為資料型別（Data Type）。

■ Java提供許多不同資料型別，應用於不同領域，佔用不同記憶體空間。

■ Java的資料在被使用之前必須先宣告資料型別。

整數

基本資料型態	名稱	位元組數 (byte)	使用說明	範圍	預設值
byte	位元組	1	最小的整數型態，適用時機：處理網路或檔案傳遞時的資料流（ stream ）。	-127~128	0
short	短整數	2	不常用的整數型態，適用時機： 16 位元電腦，但現在已經慢慢減少。	-32768~32767	0
int	整數	4	最常使用的整數型態，適用時機：一般變數的宣告、迴圈的控制單位量、陣列的索引值（ index ）。	-2147483648 ~2147483647	0
long	長整數	8	範圍較大的整數型態，適用時機：當 int （整數）不敷使用時，可以將變數晉升（ promote ）至 long （長整數）。	-922337203685 4775808L ~9223372036854 775807L	0L

整數

注意：

- ◆ 在設計程式時，要視整數能表達的有效範圍來宣告整數型別。
- ◆ 若整數資料沒有特別指定型別，Java會依出現的整數資料範圍，自動歸類出「int」及「long」兩種型別。
 - 範圍在-2,147,483,648 ~ 2,147,483,647 之間的整數資料，會被視為「int」型別，超過這個範圍者會被視為「long」型別。


整數

- ◆ 在 -2,147,483,648 ~ 2,147,483,647 之間的整數資料，若被指定為long型別，在數值後面要加上「L」或「l」。

例如：

- ◆ 數值資料的「300」預設為「int」型別。
- ◆ 若要將數值資料的「300」指定為「long」型別，則該數值資料表示方式為「300L」或「300l」。

整數

 程式：

```
public class CH02_01
{
    public static void main(String[] args)
    {
        byte a = 123;
        short b = 1234;
        int c = 345678;
        long d = 3456789123L;

        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
    }
}
```

整數

```
1 public class CH02_01
2 {
3     public static void main(String[] args)
4     {
5         byte a = 123;
6         short b = 1234;
7         int c = 345678;
8         long d = 3456789123L;
9
10        System.out.println("a = " + a);
11        System.out.println("b = " + b);
12        System.out.println("c = " + c);
13        System.out.println("d = " + d);
14    }
15 }
```

 執行結果：

```
a = 123
b = 1234
c = 345678
d = 3456789123
```

整數

說明：

◆ 行05~08：

- 宣告分別為byte、short、int、long資料型態的a、b、c、d四個變數，並直接指定初值。

◆ 行10~13：

- 螢幕輸出四個變數的內容。
 - ◆ 在括號中：
 - ◆ 雙引號的內容會直接顯示。
 - ◆ 此處的「+」當作「字串合併」運算子，它會將字串合併運算子前後的字串頭尾合併成一字串（不是「算數運算子」）。

浮點數

基本資料型態	名稱	位元組數 (byte)	使用說明	範圍	預設值
float	浮點數	4	單一精準的數值，適用時機：當需要小數計算但精準度要求不高，則float（浮點數）應該就夠使用。	1.40239846E-45 ~3.40282347E+38	0.0f
double	倍精度浮點數	8	雙重精準的數值，適用時機：小數計算精準度要求高，譬如說；「高速數學運算」、「複雜的數學函數」或「精密的數值分析」。	4.9406564584124 6544E-324~ 1.7976931348623 1570E308	0.0d

浮點數

 注意：

- ◆ 浮點數的表示方法有兩種，一種是小數點方式，另一種是科學記號方式。

例如：

3.14、-100.521、 $6e-2$ 、 $3.2E-18$ 、...等。

下表為小數點表示法與科學符號表示法的互換表：

小數點表示法	科學符號表示法
0.007	$7e-3$
-376.236	$-3.76236e+02$
89.768	$8.9768e+01$
3450000	$3.45E6$
0.000543	$5.43E-4$

浮點數

- ◆ 所有的浮點數資料Java皆預設為double型別。如果要指定浮點數為float型別，在數值後面要加上「F」或「f」。

例如：

- ◆ 數值資料「6.253」預設為double型別。
- ◆ 若要將數值資料「6.253」指定為float型別，則該數值資料表示方式為「6.253F」或「6.253f」。

浮點數

 程式：

```
public class CH02_02
{
    public static void main(String[] args)
    {
        float a = 12.5f;
        double b = 123456.654d;

        System.out.println("a=" + a);
        System.out.println("b=" + b);
    }
}
```

浮點數

```
1 public class CH02_02
2 {
3     public static void main(String[] args)
4     {
5         float a = 12.5f;
6         double b = 123456.654d;
7
8         System.out.println("a=" + a);
9         System.out.println("b=" + b);
10    }
11 }
```

 執行結果：

```
a=12.5
b=123456.654
```

浮點數

 說明：

◆ 行05~06：

- 宣告分別為float、double資料型態的a、b二個變數，並直接指定初值。
 - ◆ 數字的後面多加了一個字母「f」（「F」）、「d」（「D」）的目的是做為float（浮點數）及double（倍精數）的標記，但是通常撰寫程式時，有沒有標記並無太大的關係。

◆ 行08~09：

- 螢幕輸出二個變數的內容。

資料型別轉換

由小變大

- ◆ 「由小變大」的轉換機制會「自動轉換」，不至於損失精確度。
- ◆ 「目的變數」的資料型態必須大於「來源變數或資料」的資料型態，也就是以範圍較大的為主。

例如：

- ◆ short（短整數）可以自動轉換為int（整數）。
- ◆ int（整數）可以自動轉換為long（長整數）。
- ◆ float（浮點數）可以自動轉換為double（倍精度浮點數）。

資料型別轉換

由大轉小

- ◆ 「由大變小」的轉換機制需「指定轉換」，當「目的變數」的資料型態小於「來源變數或資料」的資料型態，其語法如下：

(指定型態) 資料 | 變數;

- 所謂「指定型態」是指目的型態。
- 「資料 | 變數」是指來源變數或資料。
- 大範圍的資料型態轉換成小範圍的資料型態時，部份資料可能會被切割。

注意：

括號，不可省略。

資料型別轉換

例如：

- ◆ 宣告2個整數變數，分別為X和Y，並各指定預設值，X=19、Y=4。
- ◆ 如果除法運算「X/Y」，則運算的結果「Z」為4。
- ◆ 如果需要結果的精確度能夠到小數點，那結果的類型就不能使用「整數int」，正確的做法應該是採用「強制轉換」的方式，重新定義結果的類型：

Z=(float)X / (float)Y;

- ◆ 說明：

先將X和Y的原本所宣告的整數類型，強制轉變成浮點數，然後才計算，此時「Z」便會為「浮點數」型態。

字元

基本資料型態	名稱	位元組數 (byte)	位元數 (bits)	範圍	預設值
char	字元	2	16	\u0000~ \uFFFF	\u0000

注意：

- ◆ 電腦無法直接處理字元，因此每一個字元都分配有一個數字編碼，稱為字元編碼。
- ◆ 每一個字元編碼都分別代表一個Unicode字元，其中前面128個為ASCII字元。
 - Unicode是一套可用來表示多國文字的字元編碼。

例如：

英文字、中文字、日本字、簡體字、... 等。

字元

- ◆ 宣告單一字元是以單引號括起來。

例如：

- ◆ 'A'表示「A」字元。
- ◆ 'n'表示「n」字元。
- ◆ '3'表示「3」這個字元（此為「字元」資料，又稱為「文數字」，它不是數值資料，故不能計算）。

- ◆ 字元也可以用Unicode表示。

例如：

- ◆ 'A'的ASCII碼為65（十進制），轉成十六進制時為0041，故「A」字元的Unicode碼為「\u0041」。
- ◆ 'n'的ASCII碼為110（十進制），轉成十六進制時為006E，故「n」字元的Unicode碼為「\u006E」。
- ◆ '3'的ASCII碼為51（十進制），轉成十六進制時為0033，故「3」字元的Unicode碼為「\u0033」。


字元

- ◆ 有些字元無法由鍵盤鍵入，又稱為「逃逸字元（Escape Character）」，可以用特殊字元組合表示，也可以Unicode用表示。

- 常用的「逃逸字元」有：

字元組合	意義	Unicode碼
\n	New Line（游標移到下一行）	\u000A
\t	Tab（游標移下一個水平定位，8格為一單位）	\u0009
\'	顯示單引號「'」	\u0027
\"	顯示雙引號「"」	\u0022
\\	顯示反斜線「\」	\u005C

字元


 程式：

```
public class CH02_03
{
    public static void main(String[] args)
    {
        char ch1 = 'X';
        char ch2 = '\u0058';

        System.out.println("ch1=" + ch1 + "\n" + "ch2=" + ch2);
    }
}
```

字元

```
1 public class CH02_03
2 {
3     public static void main(String[] args)
4     {
5         char ch1 = 'X';
6         char ch2 = '\u0058';
7
8         System.out.println("ch1=" + ch1 + "\n" + "ch2=" + ch2);
9     }
10 }
```

 執行結果：

```
ch1=X
ch2=X
```

字元

說明：

◆ 行05~06：

- 宣告char資料型態的ch1、ch2二個變數，並直接指定初值。
 - ◆ 行06是以Unicode方式來值定初值。
 - ◆ 「\u0058」即為大寫的「X」。

◆ 行08：

- 螢幕輸出二個變數的內容。
 - ◆ 括號中有使用「逃逸字元」的「\n」來換行。

字串

■ 多個字元的集合便組成了字串，字串的資料型別為 String，而字串資料的前後要用雙引號「"」括起來。

例如：

- "Hello"
- "I am a student."
- "12345"（此為「字串」資料，又稱為「文數字」，它不是數值資料，故不能計算）。

■ 其實字串屬於String類別的物件，但現在先把它當作是資料型別。

布林值

基本資料型態	名稱	位元組數 (byte)	位元數 (bits)	有效範圍
boolean	字元	2	16	true 或 false


注意：

- ◆ 布林值就只有兩個，分別是true（真）與false（假）。
- ◆ 當只有兩種選擇時，就可以用布林值。

例如：

對、錯；是、否；男、女；...等。

布林值

 程式：

```
public class CH02_04
{
    public static void main(String[] args)
    {
        boolean logic = true;

        System.out.println("宣告的布林值=" + logic);
    }
}
```


布林值

```
1 public class CH02_04
2 {
3     public static void main(String[] args)
4     {
5         boolean logic = true;
6
7         System.out.println("宣告的布林值=" + logic);
8     }
9 }
```

 執行結果：

宣告的布林值=true

布林值

 說明：

◆ 行05：

- 宣告boolean資料型態的logic變數，並直接指定初值。

◆ 行07：

- 螢幕輸出logic變數的內容。

變數

- 當程式要執行時，須先將程式和資料載入電腦的主記憶體中。
- 但程式執行時所要處理的資料如何取得呢？也就是在撰寫程式敘述時，先將每一個要處理的資料指派一個變數來存放。
- 每一個變數要分別取不同的名稱以便電腦識別，且每一個有名稱的變數皆會分配到主記憶體空間。
- 這就是資料載入主記憶體的方式。

變數

- 使用變數時，必須先宣告此變數的資料型態。
- 在Java中，變數的宣告不必在程式最前面，可以在需要使用時再宣告。
- 它的優點是可以彈性使用記憶空間，缺點是萬一宣告時，記憶空間不足時，則程式無法正常執行，尤其宣告陣列時最明顯。
- Java變數宣告的語法：

資料型別 變數名稱 [= 初值]；

同一資料型別的變數，可以一起宣告：

資料型別 變數名稱1 [= 初值1], 變數名稱2 [= 初值2]；

變數

- 假設宣告兩個整變數num1、num2，其中int為Java中整數宣告的關鍵字（keyword）：

```
int num1=30;
```

```
int num2=77;
```

- 這時Java系統會分別自動分配記憶體給變數num1，儲存值為30，及變數num2，儲存值為77，當程式需要存取這塊記憶體時，就可直接利用變數名稱num1與num2來進行存取。

記憶體位置

變數名稱

1024

30

num1

1028

77

num2

變數

變數的命名規則：

◆ 變數命名有一定的要求與規則性：

- 識別字名稱可以由英文字母大小寫（A~Z、a~z），數字（0~9），底線（_），或貨幣符號（\$）等字元組成，且第一個字元不可以使用數字。
- 在Java中，識別字名稱中的英文字母大小寫是有分別的。

例如：

「PRICE」、「Price」、「price」三者分別代表不同的名稱。

- 在Java程式中有一個不成文的規則，通常變數名稱是以小寫英文字母作為開頭，並接上一個大寫開頭有意義的單字。

變數

例如：

存放「使用者密碼」的變數可命名為userPassword。

- 識別名稱最好取有意義名稱，方便程式的撰寫與維護。

例如：

score（成績）、name（姓名）、tel_no（電話號碼）。

- 同一範圍內，變數名稱必須是獨一無二；但在不同範圍下，變數的名稱可以允許相同。
- 變數名稱不可為關鍵字（Keyword）、保留字、運算子及其它符號。

例如：

int、class、+、-、*、/、@、#、...等。

下表將關鍵字以功能使用性分類如下：

變數

程式流程控制	do	while	if	else	for	goto
	switch	case	break	continue	return	throw
	throws	try	catch	finally		
資料型態設定	double	float	int	long	short	boolean
	byte	char				
物件特性宣告	synchronized	native	import	public	class	static
	abstract	private	void	extend	protected	default
	implements	interface	package			
其它功能	this	new	super	instanceof	assert	null
	const	strictfp	volatile	transient	true	false
	final					

變數

下表舉例不同的命名結果，說明是否合乎命名規則：

範例	合法	不合法	說明
My_name_is_Tim	✓		符合命名規則。
My_name_is_TimChen_Boy	✓		Java 變數名稱的長度沒有限制，所以符合命名規則。
Java 2		✓	不可以有空白字元，正確應該是「 Java2 」。
Java_2	✓		符合命名規則。
_TimChen	✓		符合命名規則。
AaBbCc	✓		符合命名規則。
2_Java		✓	第一個字元不可以是數字，正確應該是「 Java2 」或「 _2Java 」。
@yahoo		✓	不可以使用特殊符號「@」，可以更改成「 yahoo 」。
A=1+1		✓	不可以使用運算符號「+、-、×、\」。

變數

 程式：

```
public class CH02_05
{
    public static void main(String args[])
    {
        int m = 10;
        double C = 2997924581.2, e;

        e = m * C * C;

        System.out.println("當質量為：" + m);
        System.out.println("所釋放出的能量為：" + e);
    }
}
```

變數

```
1 public class CH02_05
2 {
3     public static void main(String args[])
4     {
5         int m = 10;
6         double C = 2997924581.2, e;
7
8         e = m * C * C;
9
10        System.out.println("當質量為：" + m);
11        System.out.println("所釋放出的能量為：" + e);
12    }
13 }
```

 執行結果：

當質量為：10
所釋放出的能量為：8.987551794563195E19

變數

說明：

◆ 行05：

- 宣告int資料型態的變數m，並指定初值為10。

◆ 行06：

- 宣告double資料型態的變數C、e，並指定C的初值為2997924581.2（光速）。

◆ 行08：

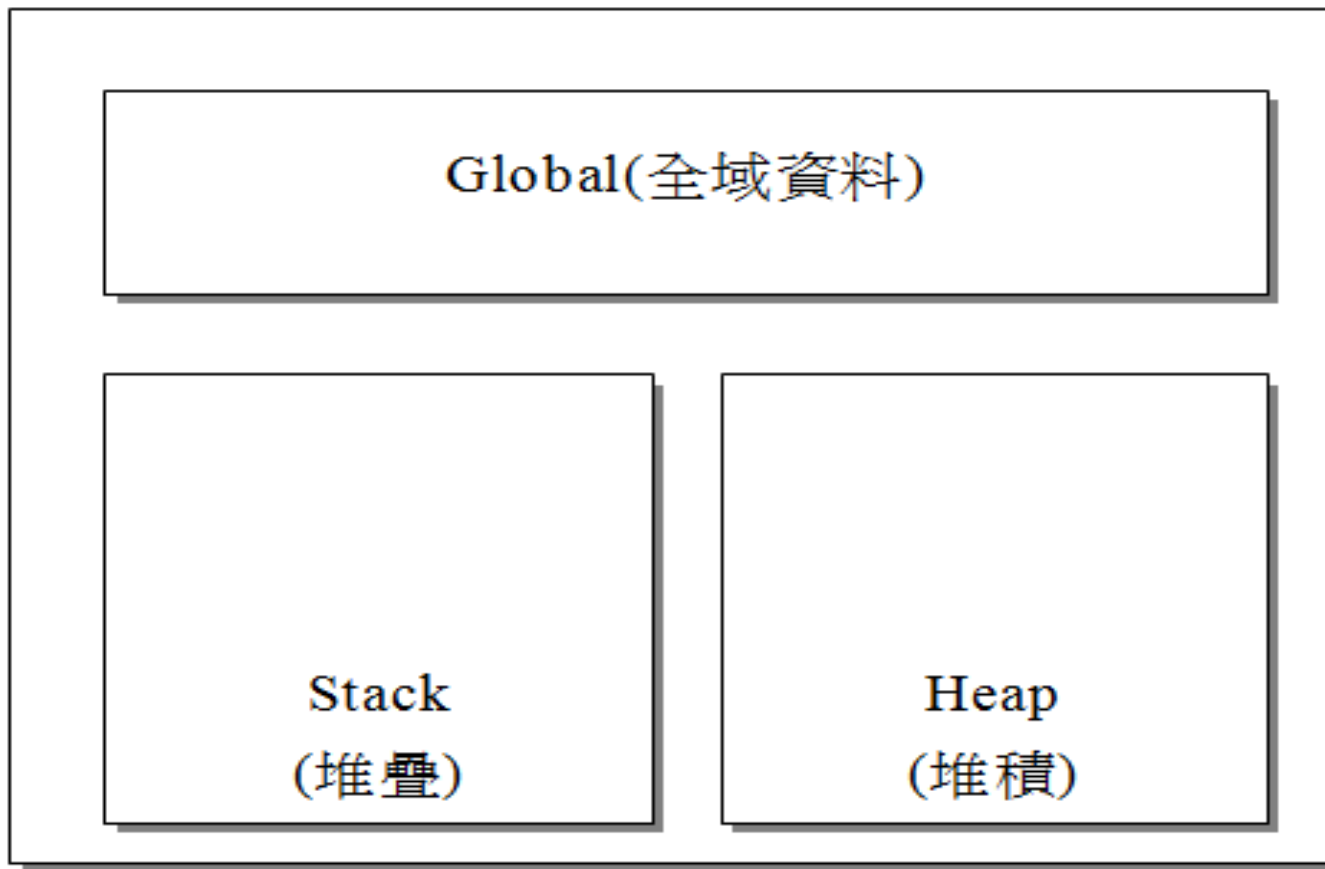
- 計算能量，並將計算結果存於變數e。

◆ 行10~11：

- 螢幕輸出m、e二個變數的內容。

基本資料型別與參考資料型別

- Java 物件變數 在記憶體內 Global、Stack、Heap 各儲存空間的配置方式：



基本資料型別與參考資料型別

Global（全域資料）

- ◆ 使用 static 保留字宣告的資料成員變數都存放在 Global 儲存空間稱為「靜態成員」或稱「類別成員」。
- ◆ 「靜態成員」是類別中共用成員，並不是指特定物件，不會因建立新物件實體再產生一個新靜態成員。
- ◆ 某一類別所產生的物件皆可共用「靜態成員」的資料。

基本資料型別與參考資料型別

Stack（堆疊）

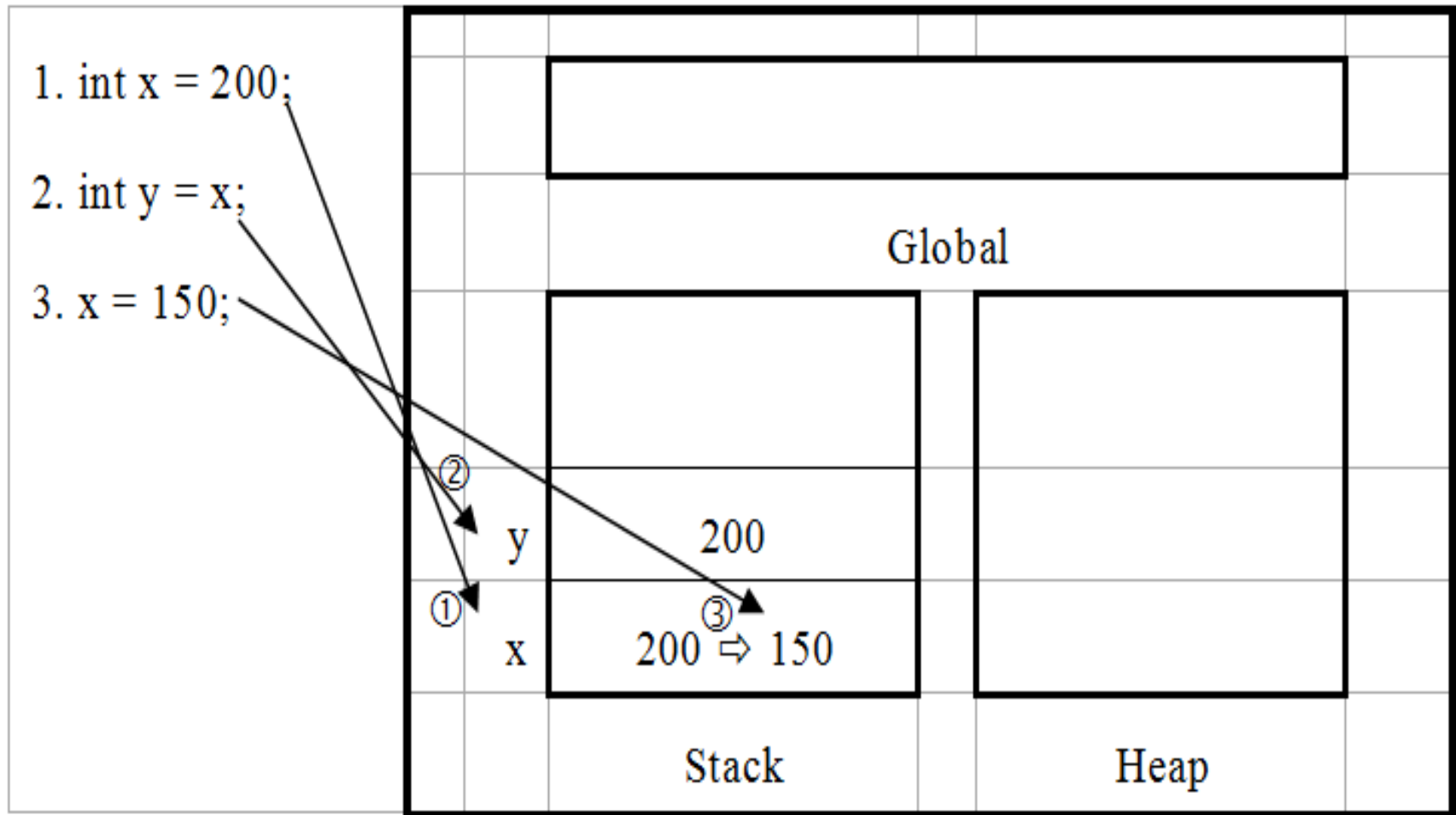
- ◆ 宣告屬於「基本資料」型別的變數，皆儲放在 Stack 記憶空間。

例如：

char、byte、short、int、long、float、double、boolean
。

- ◆ Stack 記憶空間是「直接存放」變數的「內容」。
- ◆ 使用 Stack 的好處是佔用記憶體空間小，存取速度快。
- ◆ 下面三行敘述說明 Stack 儲存變數與配置方式（如下圖）：

基本資料型別與參考資料型別



基本資料型別與參考資料型別

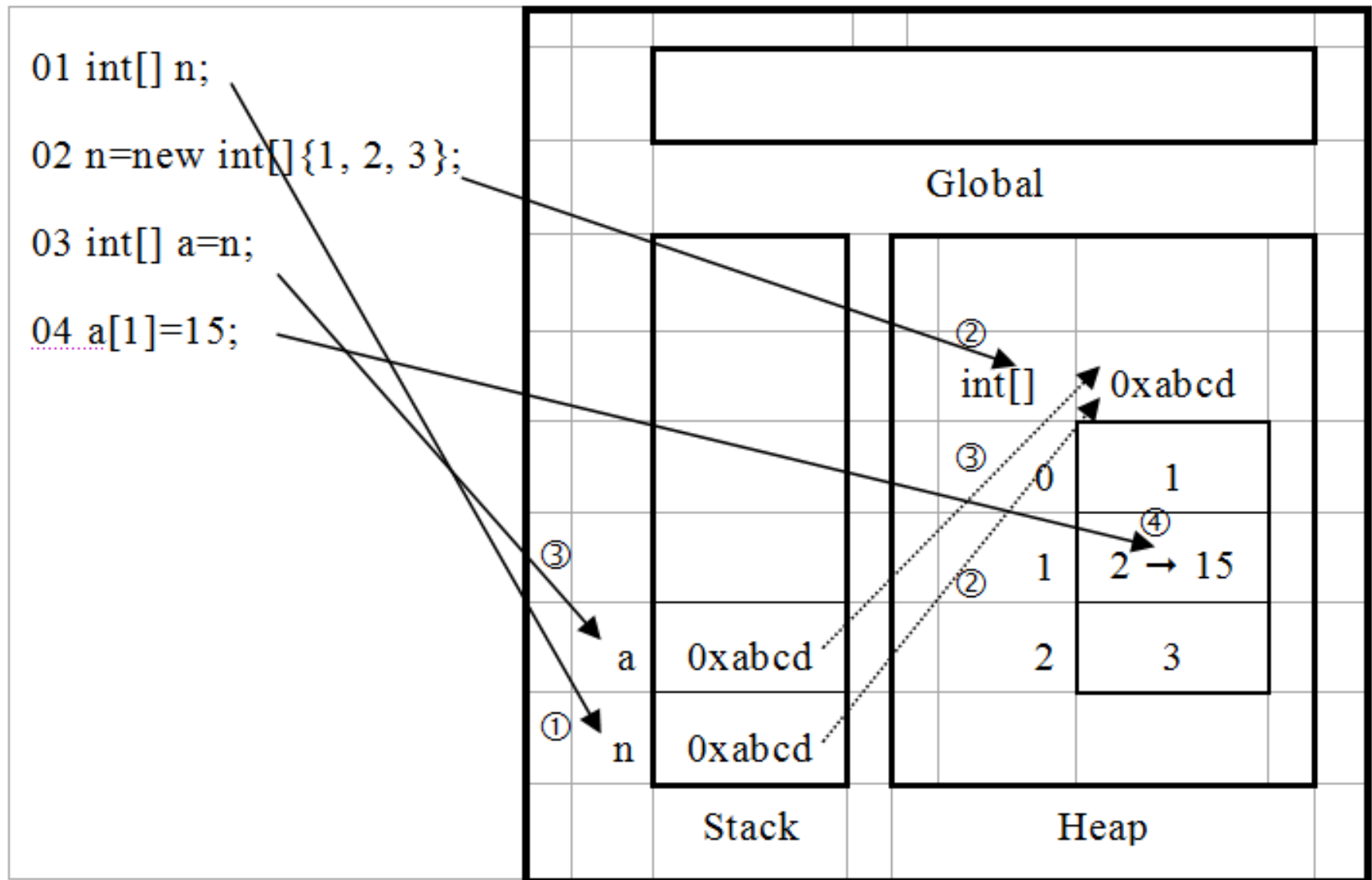
Heap（堆積）

- ◆ Heap中是存放「參考資料型別」的物件實體。
- ◆ 當宣告屬於「參考資料型別」物件變數時，會在Stack配置一塊記憶空間存放該物件變數的參考值（非資料「本身」，是資料「位址」）。
- ◆ 當用new建立新物件實體時，會在Heap中配置一塊記憶空間用來存放物件實體（指資料「本身」），此時Stack的物件變數參考值會指向Heap中該物件實體。
- ◆ 「參考型別」變數都以間接方式設定或取得資料。

例如：

陣列、類別、字串、...等。

基本資料型別與參考資料型別



運算式與運算子

■ 運算式（Expression）就像平常所用的數學公式一樣，是由運算子（Operator）與運算元（Operand）所組成。

例如：

d=a*b-123.4;

- 其中「d」、「a」、「b」、「123.4」等變數或常數稱為運算元，而「=」、「*」、「-」等運算符號稱為運算子。

■ Java的運算子除了基本的加、減、乘和除四則運算符號，還有很多運算子，譬如指定運算子（=）符號，也是屬於運算子的一種。

運算式與運算子

算術運算子

- ◆ 用法及功能和傳統的數學運算相同，但值得注意的是加法運算子。
 - 加法運算子除了可以執行數值計算，還具有「字串連結」的功能。
- ◆ 基本運算子的用法，整理如下表：

算術運算子	用途	範例	結果
+	加法	$X=2 + 3$	$X=5$
-	減法	$X=5 - 3$	$X=2$
*	乘法	$X=5 * 4$	$X=20$
/	除法	$X=100 / 50$	$X=2$
%	取餘數	$X=100 \% 33$	$X=1$

運算式與運算子

◆ 數值的正負數表示：

- 當數值分成正數和負數時，正數不用任何符號作區別，但負數則要使用減法（-）運算子的符號來表示。

例如：

```
int x=5;
```

```
int y=-5;
```

- 當負數進行減法運算時，為了避免運算子的分辨混淆，最好應以空白字或「括號（）」隔開。

例如：

```
x=x- -2;
```

```
x=x-(-2);
```

運算式與運算子

■ 程式：

```
public class CH02_06
{
    public static void main(String[] args)
    {
        int apple = 15, banana = 20;

        System.out.print("(1).小明買蘋果15顆，香蕉20條，水果總共買了");
        System.out.println((apple + banana) + "個");
        System.out.print("(2).蘋果每顆10元，香蕉每條3元，水果總共花費");
        System.out.println((apple * 10 + banana * 3) + "元");
        System.out.print("(3).將蘋果4個和香蕉3個裝成一盒，共可包裝");
        System.out.println((apple / 4) + "盒");
        System.out.println("(4).裝盒後蘋果剩下" + (apple % 4) + "個，" + "香蕉剩下" + (15 - 3 * 3) + "個");
    }
}
```


運算式與運算子

```
1 public class CH02_06
2 {
3     public static void main(String[] args)
4     {
5         int apple = 15, banana = 20;
6
7         System.out.print("(1).小明買蘋果15顆，香蕉20條，水果總共買了");
8         System.out.println((apple + banana) + "個");
9         System.out.print("(2).蘋果每顆10元，香蕉每條3元，水果總共花費");
10        System.out.println((apple * 10 + banana * 3) + "元");
11        System.out.print("(3).將蘋果4個和香蕉3個裝成一盒，共可包裝");
12        System.out.println((apple / 4) + "盒");
13        System.out.println("(4).裝盒後蘋果剩下" + (apple % 4) + "個，" + "香蕉剩下" + (15 - 3 * 3) + "個");
14    }
15 }
```

執行結果：

(1).小明買蘋果15顆，香蕉20條，水果總共買了35個
(2).蘋果每顆10元，香蕉每條3元，水果總共花費210元
(3).將蘋果4個和香蕉3個裝成一盒，共可包裝3盒
(4).裝盒後蘋果剩下3個，香蕉剩下6個

運算式與運算子

 說明：

◆ 行05：

- 宣告變數，並指定初值。

◆ 行07~13：

- 螢幕輸出算術運算結果。

運算式與運算子

■ 遞增（Increment）與遞減（Decrement）運算子

◆ 遞增與遞減運算子還是可以細分成分成前序（Prefix）及後序（Postfix）兩種。

◆ 運算方式如下表：


運算子	使用方式	範例：X=5	運算結果	註解
++	前序	A=++X	A=6；X=6	先將X值加1後，再將X值儲存於A中
	後序	A=X++	A=5；X=6	先將X值儲存於A後，再將X值加1
--	前序	A=--X	A=4；X=4	先將X值減1後，再將X值儲存於A中
	後序	A=X--	A=5；X=4	先將X值儲存於A後，再將X值減1

運算式與運算子

關係運算子

關係運算子	用途	範例	運算執行結果
==	等於	10 == 10	true
		5 == 3	false
!=	不等於	10 != 10	false
		5 != 3	true
>	大於	10 > 10	false
		5 > 3	true
<	小於	10 < 10	false
		5 < 3	false
>=	大於或等於	10 >= 10	true
		5 >= 3	true
<=	小於或等於	10 <= 10	true
		5 <= 3	false

運算式與運算子

 程式：

```
public class CH02_07
{
    public static void main(String[] args)
    {
        System.out.println("15大於5 為" + (15 > 5) + "\n");
        System.out.println("15小於5 為" + (15 < 5) + "\n");
        System.out.println("15大於等於15 為" + (15 >= 15) + "\n");
        System.out.println("15小於等於5 為" + (15 <= 5) + "\n");
        System.out.println("15不等於5 為" + (15 != 5) + "\n");
        System.out.println("15等於5 為" + (15 == 5) + "\n");
    }
}
```

運算式與運算子

```
1 public class CH02_07
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("15大於5 為" + (15 > 5) + "\n");
6         System.out.println("15小於5 為" + (15 < 5) + "\n");
7         System.out.println("15大於等於15 為" + (15 >= 15) + "\n");
8         System.out.println("15小於等於5 為" + (15 <= 5) + "\n");
9         System.out.println("15不等於5 為" + (15 != 5) + "\n");
10        System.out.println("15等於5 為" + (15 == 5) + "\n");
11    }
12 }
```

 執行結果：

15大於5 為true

15小於5 為false

15大於等於15 為true

15小於等於5 為false

15不等於5 為true

15等於5 為false

運算式與運算子

 說明：

◆ 行05~10：

- 螢幕輸出運算2數值的關係。
 - ◆ 關係運算子計算的結果是得知運算元之間的關係，所以結果不是以數值的型式出現，而是以布林（boolean）的資料型態出現。
 - ◆ 程式碼中使用「"\n"」，是表示「換行」的意思。

運算式與運算子

邏輯運算子

邏輯運算子	用途	範例：boolean A, B	運算結果說明
!	NOT	!A	當A為true，傳回值為false
			當A為false，傳回值為true
&	AND	A & B	只有當A和B都為true，傳回值為true，否則全為false
&&	AND	A && B	只有當A和B都為true，傳回值為true，否則全為false
	OR	A B	只有當A和B都為false，傳回值為false，否則全為true
	OR	A B	只有當A和B都為false，傳回值為false，否則全為true
^	XOR	A ^ B	只有當A和B都為true或false，傳回值為false，否則全為true

運算式與運算子

- ◆ 經過與運算元結合成邏輯運算式，運算後會產生 true（真）或 false（假）兩個結果。
- ◆ 「&」與「&&」及「|」與「||」的運算結果是相同的，但「&&」及「||」可以加快執行效率。
 - 說明：
 - ◆ 當使用「&」與「|」時，必須做全部的邏輯運算，才能決定運算結果；當使用「&&」與「||」時，只要前面條件滿足，不必執行後面的條件，就可以得到運算結果。
 - ◆ 例如：
在做「&」運算時，必須所有的值均為「true」，結果才會是「true」；但用「&&」運算時，當有一運算結果為「false」，其結果就是「false」，不必在往後判斷其它的值。

運算式與運算子

■ 程式：

```
public class CH02_08
{
    public static void main(String[] args)
    {
        int a = 15, b = 3;

        System.out.println("(a>10)的傳回值為" + (a > 10));
        System.out.println("(!(a>10)的傳回值為" + !(a > 10));
        System.out.println("(a>10)&(b>5)的傳回值為 " + (a > 10 & b > 5));
        System.out.println("(a>10)&&(b>5)的傳回值為 " + (a > 10 && b > 5));
        System.out.println("(a>10)|(b>5)的傳回值為 " + (a > 10 | b > 5));
        System.out.println("(a>10)||(b>5)的傳回值為 " + (a > 10 || b > 5));
        System.out.println("(a>10)^(b>5)的傳回值為 " + (a > 10 ^ b > 5));
    }
}
```


運算式與運算子

```
1 public class CH02_08
2 {
3     public static void main(String[] args)
4     {
5         int a = 15, b = 3;
6
7         System.out.println("(a>10)的傳回值為" + (a > 10));
8         System.out.println("!a>10)的傳回值為" + !(a > 10));
9         System.out.println("(a>10)&(b>5)的傳回值為" + (a > 10 & b > 5));
10        System.out.println("(a>10)&&(b>5)的傳回值為" + (a > 10 && b > 5));
11        System.out.println("(a>10)|(b>5)的傳回值為" + (a > 10 | b > 5));
12        System.out.println("(a>10)||(b>5)的傳回值為" + (a > 10 || b > 5));
13        System.out.println("(a>10)^(b>5)的傳回值為" + (a > 10 ^ b > 5));
14    }
15 }
```

■ 執行結果：

(a>10)的傳回值為true
!(a>10)的傳回值為false
(a>10)&(b>5)的傳回值為 false
(a>10)&&(b>5)的傳回值為 false
(a>10)|(b>5)的傳回值為 true
(a>10)||(b>5)的傳回值為 true
(a>10)^(b>5)的傳回值為 true

運算式與運算子

 說明：

◆ 行05：

- 宣告變數，並指定初值。

◆ 行07~13：

- 螢幕輸出邏輯運算結果。

運算式與運算子

位元運算子

位元運算子	用途	A=00000101 B=00000111	運算結果	註解
~	補數	~A	11111010	1轉換成0，0轉換成1
&	AND	A & B	00000101	只有1 & 1為1，否則為0
	OR	A B	00000111	只有0 0為0，否則為1
^	XOR	A ^ B	00000010	只有1^0或0^1為1，否則為0

運算式與運算子

 程式：

```
public class CH02_09
{
    public static void main(String[] args)
    {
        int a = 15, b = 3;

        System.out.println("15 & 3 的傳回值為 " + (a & b));
        System.out.println("15 / 3 的傳回值為 " + (a / b));
        System.out.println("15 ^ 3 的傳回值為 " + (a ^ b));
        System.out.println("~3 的傳回值為 " + (~b));
    }
}
```


運算式與運算子

```
1 public class CH02_09
2 {
3     public static void main(String[] args)
4     {
5         int a = 15, b = 3;
6
7         System.out.println(" 15 & 3 的傳回值為 " + (a & b));
8         System.out.println(" 15 | 3 的傳回值為 " + (a | b));
9         System.out.println(" 15 ^ 3 的傳回值為 " + (a ^ b));
10        System.out.println(" ~3 的傳回值為 " + (~b));
11    }
12 }
```

 執行結果：

15 & 3 的傳回值為 3
15 | 3 的傳回值為 15
15 ^ 3 的傳回值為 12
~3 的傳回值為 4

運算式與運算子

 說明：

◆ 行05：

- 宣告變數，並指定初值。

◆ 行07~10：

- 螢幕輸出位元運算結果。

運算式與運算子

移位運算子

移位運算子	用途	使用的語法	例子	運算結果	說明
<< (將數值的位元向左移動n個位元。向左移動後，超出儲存範圍的數字捨去，右邊位元則補上0。)	左移	【整數值】 << 【移位值】	5<<2	20	5的二進位值為00000101，位元左移兩個位元，將左移所空出的位元補上0，如000101 <u>00</u> 換成整數為20
			(-5)<<2	-20	-5的二進位值為11111010，位元左移兩個位元，將左移所空出的位元補上1，如111010 <u>11</u> 換成整數為-20
>> (是將數值的位元向右移動n個位元。向右移動後，超出儲存範圍的數字捨去，而左邊多出的位元就補上0。)	右移	【整數值】 << 【移位值】	20>>2	5	20的二進位值為00010100，位元右移兩個位元，將右移所空出的位元補上0，如 <u>00000</u> 101換成整數為5
			-20>>2	-5	-20的二進位值為11101011，位元左移兩個位元，將左移所空出的位元補上1，如 <u>1111</u> 1010換成整數為-5

運算式與運算子

 程式：

```
public class CH02_10
{
    public static void main(String[] args)
    {
        System.out.println("5 << 2 的傳回值為 " + (5 << 2));
        System.out.println("-5 << 2 的傳回值為 " + (-5 << 2));
        System.out.println("5 >> 2 的傳回值為 " + (5 >> 2));
        System.out.println("-5 >> 2 的傳回值為 " + (-5 >> 2));
    }
}
```


運算式與運算子

```
1 public class CH02_10
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("5 << 2 的傳回值為 " + (5 << 2));
6         System.out.println("-5 << 2 的傳回值為 " + (-5 << 2));
7         System.out.println("5 >> 2 的傳回值為 " + (5 >> 2));
8         System.out.println("-5 >> 2 的傳回值為 " + (-5 >> 2));
9     }
10 }
```

 計算結果：

5 << 2 的傳回值為 20
-5 << 2 的傳回值為 -20
5 >> 2 的傳回值為 1
-5 >> 2 的傳回值為 -2

運算式與運算子

 說明：

◆ 行05~08：

- 螢幕輸出移位運算結果。

運算式與運算子

指定運算子

指定運算子	用途	A=4	運算過程	運算結果
=	指派	A=88		88
+=	相加後再指派	A+=2	A=A+2	6
-=	相減後再指派	A-=2	A=A-2	2
=	相乘後再指派	A=2	A=A*2	8
/=	相除後再指派	A/=2	A=A/2	2
%=	餘數後再指派	A%=2	A=A%2	0

運算式與運算子

 程式：

```
public class CH02_11
{
    public static void main(String[] args)
    {
        int A = 4;

        System.out.println("A 的值為 " + A);
        System.out.println("A+= 3+2 的值為 " + (A += 3 + 2));
        System.out.println("A-= 5-4 的值為 " + (A -= 5 - 4));
        System.out.println("A*= 2*3 的值為 " + (A *= 2 * 3));
        System.out.println("A/= 10/5+3 的值為 " + (A /= 10 / 5 + 3));
        System.out.println("A%= 10%3 的值為 " + (A %= 15 % 4));
    }
}
```

運算式與運算子

```
1 public class CH02_11
2 {
3     public static void main(String[] args)
4     {
5         int A = 4;
6
7         System.out.println("A 的值為 " + A);
8         System.out.println("A+= 3+2 的值為 " + (A += 3 + 2));
9         System.out.println("A-= 5-4 的值為 " + (A -= 5 - 4));
10        System.out.println("A*= 2*3 的值為 " + (A *= 2 * 3));
11        System.out.println("A/= 10/5+3 的值為 " + (A /= 10 / 5 + 3));
12        System.out.println("A%= 10%3 的值為 " + (A %= 15 % 4));
13    }
14 }
```

 執行結果：

A 的值為 4
A+= 3+2 的值為 9
A-= 5-4 的值為 8
A*= 2*3 的值為 48
A/= 10/5+3 的值為 9
A%= 10%3 的值為 0

運算式與運算子

 說明：

◆ 行05：

- 宣告變數，並指定初值。

◆ 行07~12：

- 螢幕輸出指定運算結果。
- 註：

- ◆ Java的指定運算子除了一次指定一個數值給變數外，還能夠同時指定同一個數值給多個變數。

例如：

```
int x,y,z ;
```

```
x=y=z=200; /* 同步指定值給不同變數 */
```


運算式與運算子

運算子執行的優先順序

- ◆ 在處理一個多運算子的運算式時，有一些規則與步驟是必須要遵守，如下所示：
 - 當遇到一個運算式時，先區分運算子與運算元。
 - 依照運算子的優先順序作整理的動作。
 - 將各運算子根據其結合順序進行運算。
- ◆ 運算子彼此間運算的順序，如下表所示：

優先序	運算子	結合律
1	() 、 []	由右至左
2	++ 、 -- 、 - 、 ! 、 ~	由左至右
3	* 、 / 、 %	由左至右
4	+ 、 -	由左至右

運算式與運算子

優先序	運算子	結合律
5	<< 、>> 、>>> (無正負性位元右移)	由左至右
6	< 、> 、<= 、>=	由左至右
7	== 、 !=	由左至右
8	&	由左至右
9	^	由左至右
10		由左至右
11	&&	由左至右
12		由左至右
13	? : (條件選擇)	由右至左
14	=	由右至左
15	+= 、 -= 、 *= 、 /= 、 %= 、 &= 、 = 、 ^=	由右至左

輸出與輸入資料

■ 要與電腦進行互動，就是能對電腦進行資料輸入，而電腦會依據資料做處理，然後再輸出處理結果。

■ 由螢幕（主控台）輸出：

◆ 宣告方式：

System.out.print(資料);

System.out.println(資料);

◆ 說明：

- System.out：

- ◆ 代表系統的標準輸出。

輸出與輸入資料

- println與print：
 - ◆ 它們的功能是將括弧內的字串，作印出的動作。
 - ◆ 差別在於print在印出內容後不會換行，而println則會自動跳行。
- 資料：
 - ◆ 格式可以是任何型態，包括變數、常數、字元、字串或物件等。

輸出與輸入資料

◆ 程式：

```
public class CH02_12
{
    public static void main(String[] args)
    {
        String myStringA = "第一個字串";
        String myStringB = "第二個字串";
        String myStringC = "會串聯在一起";

        System.out.println("[J A V A 基本輸出練習]");
        System.out.print(myStringA);
        System.out.print(" 與 ");
        System.out.print(myStringB + "\n");
        System.out.print(myStringC);
    }
}
```

輸出與輸入資料

```
1 public class CH02_12
2 {
3     public static void main(String[] args)
4     {
5         String myStringA = "第一個字串";
6         String myStringB = "第二個字串";
7         String myStringC = "會串聯在一起";
8
9         System.out.println("[JAVA基本輸出練習]");
10        System.out.print(myStringA);
11        System.out.print(" 與 ");
12        System.out.print(myStringB + "\n");
13        System.out.print(myStringC);
14    }
15 }
```

◆ 執行結果：

[JAVA基本輸出練習]
第一個字串 與 第二個字串
會串聯在一起

輸出與輸入資料

◆ 說明：

- 行05~07：
 - ◆ 宣告變數，並指定初值。
- 行09~13：
 - ◆ 螢幕輸出結果。

輸出與輸入資料

- 在主控制台模式下，由鍵盤輸入的資料，按下Enter鍵後，電腦在讀取時，會把按Enter鍵之前所鍵入的字元、文字、數字、符號，當成是一個字串資料，讀入程式中，再依使用的方法，分成字元、字串。
- 字串又可再細分成文字字串與數字。
- 共同部分：
 - ◆ 在Java程式處理鍵盤輸入的流程中，必須載入Java.io.*套件，也就是在程式碼的開頭處，撰寫下面的敘述：

```
import java.io.*;
```


輸出與輸入資料

- ◆ 在main()方法後面，加上 throws IOException，該行敘述如下：

public static void main(String[] args) throws IOException

■ 由鍵盤（主控台）輸入「字元」：

- ◆ 宣告字元型別的變數，敘述如下：

char 變數名稱;

- ◆ 讀取由鍵盤輸入的字元，並指派給字元型別的變數保存，敘述如下：

變數名稱 = (char)System.in.read();

輸出與輸入資料

- 說明：

- ◆ System.in：

- ◆ 代表系統的標準輸入。

- ◆ read()：

- read()方法的功能是先從輸入串流（例如鍵盤輸入的字串）中讀取一個位元組的資料（即字元）後，再傳出0~255之間的整數型態資料（ASCII碼）。

輸出與輸入資料

◆ 程式：

```
import java.io.*;

public class CH02_13
{
    public static void main(String[] args) throws IOException
    {
        char myData;

        System.out.print("[基本輸入練習]\n");
        System.out.print("請輸入文字：");

        myData = (char) System.in.read();
        System.out.println("輸入的資料為：" + myData);
    }
}
```

輸出與輸入資料

```
1 import java.io.*;
2
3 public class CH02_13
4 {
5     public static void main(String[] args) throws IOException
6     {
7         char myData;
8
9         System.out.print("[基本輸入練習]\n");
10        System.out.print("請輸入文字：");
11
12        myData = (char) System.in.read();
13        System.out.println("輸入的資料為：" + myData);
14    }
15 }
```

◆ 執行結果：

[基本輸入練習]
請輸入文字：Andy ← 輸入
輸入的資料為：A

輸出與輸入資料

◆ 說明：

- 行01：
 - ◆ 載入Java.io.*套件。
- 行05：
 - ◆ main()方法後面，加上 throws IOException（即拋出IO例外物件，也就是預防輸入時所產生的意外，後面章節會詳細介紹）。
- 行07：
 - ◆ 宣告字元變數。
- 行09~10：
 - ◆ 螢幕輸出字串。
- 行12：
 - ◆ 讀取輸入的字元。

輸出與輸入資料

- 行13：
 - ◆ 由螢幕顯示輸入的字元。

輸出與輸入資料

■ 由鍵盤（主控台）輸入「字串」：

◆ 宣告BufferedReader類別的物件，敘述如下：

BufferedReader 物件名稱;

◆ 建立已宣告的BufferedReader類別物件，敘述如下：

物件名稱 = new BufferedReader(new InputStreamReader(System.in));

註：

上述兩行宣告，可合併為一行。

**BufferedReader 物件名稱 = new BufferedReader(new
InputStreamReader(System.in));**

輸出與輸入資料

- ◆ 讀取由鍵盤輸入的字串，並指派給字串變數，敘述如下：

String 變數名稱 = 物件名稱.readLine();

輸出與輸入資料

◆ 程式：

```
import java.io.*;

public class CH02_14
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader keyin;
        keyin = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("鍵入字串：");
        String st = keyin.readLine();
        System.out.println("顯示字串：" + st);
    }
}
```

輸出與輸入資料

```
1 import java.io.*;
2
3 public class CH02_14
4 {
5     public static void main(String[] args) throws IOException
6     {
7         BufferedReader keyin;
8         keyin = new BufferedReader(new InputStreamReader(System.in));
9         System.out.print("鍵入字串：");
10        String st = keyin.readLine();
11        System.out.println("顯示字串：" + st);
12    }
13 }
```

◆ 執行結果：

鍵入字串：ABCDE
顯示字串：ABCDE

輸出與輸入資料

◆ 說明：

- 行01：
 - ◆ 載入Java.io.*套件。
- 行05：
 - ◆ main()方法後面，加上 throws IOException。
- 行07：
 - ◆ 宣告BufferedReader類別的物件。
- 行08：
 - ◆ 建立keyin物件。
- 行09：
 - ◆ 螢幕輸出字串。
- 行10：
 - ◆ 利用keyin物件的readLine()方法，讀取由鍵盤輸入的字串，並指派給字串變數st。

輸出與輸入資料

- 行11：
 - ◆ 由螢幕顯示輸入的字串。

輸出與輸入資料

 由鍵盤（主控台）輸入「數字」：

- ◆ 雖然由鍵盤輸入的數字會被視為字串，但只要經過資料轉換，仍然可以將文數字（數字型態的字串）變成數值（文數字可以轉換成各種數值型態的資料），再存入變數。

- 將文數字轉換成int型別：

int 整數變數名稱= Integer.parseInt(字串變數名稱);

- 將文數字轉換成long型別：

long 長整數變數名稱= Long.parseLong(字串變數名稱);

輸出與輸入資料

- 將文數字轉換成float型別：

float 浮點數變數名稱= Float.parseFloat(字串變數名稱);

- 將文數字轉換成double型別：

double 倍精度浮點數變數名稱= Double.parseDouble(字串變數名稱);

輸出與輸入資料

◆ 程式：

```
import java.io.*;

public class CH02_15
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader keyin = new BufferedReader(new
                                                    InputStreamReader(System.in));

        System.out.print("輸入整數字串：");
        String st = keyin.readLine();
        System.out.println();

        int numI = Integer.parseInt(st);
        long numL = Long.parseLong(st);
        float numF = Float.parseFloat(st);
        double numD = Double.parseDouble(st);
    }
}
```

輸出與輸入資料

```
System.out.println("顯示整數數值：" + numI + "\n");  
System.out.println("顯示長整數數值：" + numL + "\n");  
System.out.println("顯示浮點數數值：" + numF + "\n");  
System.out.println("顯示倍精度浮點數數值：" + numD + "\n");  
}  
}
```


輸出與輸入資料

```
1 import java.io.*;
2
3 public class CH02_15
4 {
5     public static void main(String[] args) throws IOException
6     {
7         BufferedReader keyin = new BufferedReader(new InputStreamReader(System.in));
8
9         System.out.print("輸入整數字串： ");
10        String st = keyin.readLine();
11        System.out.println();
12
13        int numI = Integer.parseInt(st);
14        long numL = Long.parseLong(st);
15        float numF = Float.parseFloat(st);
16        double numD = Double.parseDouble(st);
17
18        System.out.println("顯示整數數值：" + numI + "\n");
19        System.out.println("顯示長整數數值：" + numL + "\n");
20        System.out.println("顯示浮點數數值：" + numF + "\n");
21        System.out.println("顯示倍精度浮點數數值：" + numD + "\n");
22    }
23 }
```

輸出與輸入資料

◆ 執行結果：

輸入整數字串： 123456

顯示整數數值：123456

顯示長整數數值：123456

顯示浮點數數值：123456.0

顯示倍精度浮點數數值：123456.0

◆ 說明：

- 行01：

- ◆ 載入Java.io.*套件。

- 行05：

- ◆ main()方法後面，加上 throws IOException。

輸出與輸入資料

- 行07：
 - ◆ 宣告BufferedReader類別的物件。
- 行08：
 - ◆ 建立keyin物件。
- 行09：
 - ◆ 螢幕輸出字串。
- 行10：
 - ◆ 利用keyin物件的readLine()方法，讀取由鍵盤輸入的字串，並指派給字串變數st。
- 行11：
 - ◆ 將st字串的內容轉換成int型別的數值，並存入變數num中。
- 行12：
 - ◆ 由螢幕顯示輸入的數字。

資料來源

- 蔡文龍、何嘉益、張志成、張力元，JAVA SE 10基礎必修課，台北市，碁峰資訊股份有限公司，2018年7月，出版。
- 吳燦銘、胡昭民，圖解資料結構-使用Java(第三版)，新北市，博碩文化股份有限公司，2018年5月，出版。
- Ivor Horton，Java 8 教學手冊，台北市，碁峰資訊股份有限公司，2016年9月，出版。
- 李春雄，程式邏輯訓練入門與運用---使用JAVA SE 8，台北市，上奇科技股份有限公司，2016年6月，初版。
- 位元文化，Java 8視窗程式設計，台北市，松崗資產管理股份有限公司，2015年12月，出版。
- Benjamin J Evans、David Flanagan，Java 技術手冊 第六版，台北市，碁峰資訊股份有限公司，2015年7月，出版。
- 蔡文龍、張志成，JAVA SE 8 基礎必修課，台北市，碁峰資訊股份有限公司，2014年11月，出版。
- 陳德來，Java SE 8程式設計實例，台北市，上奇科技股份有限公司，2014年11月，初版。
- 林信良，Java SE 8 技術手冊，台北市，碁峰資訊股份有限公司，2014年6月，出版。
- 何嘉益、黃世陽、李篤易、張世杰、黃鳳梅，徐政棠譯，JAVA2 程式設計從零開始--適用JDK7，台北市，上奇資訊股份有限公司，2012年5月，出版。