




## 第2章 JAVA語言的處理流程與開發環境

### 一、Java 語言的基本要素



#### 程式中的註解行

-  多行註解的指令為「`/*...*/`」。
-  單行的註解行指令，以雙斜線「`//`」作為註解行的開始，一直延伸到本行結束。
-  另外還有一種註解指令，其用途是給 javadoc 使用，可以產生Java 應用程式文件，其指令為「`/** ... */`」。

## 第2章 JAVA語言的處理流程與開發環境

### 一、Java 語言的基本要素



#### 程式中的空白字元



程式中的空白行(也就是行結束符號)、空白字元、水平跳位字元(tab)、換頁字元等都是「空白字元」。



空白字元的功能與註解行一樣，不會影響程式編譯的結果，但是可以增加原始程式碼的可閱讀性。

# 第2章 JAVA語言的處理流程與開發環境

## 二、Java 程式語言的語彙

 識別字

 關鍵字

 文字








 分隔子

 運算子



## 第2章 JAVA語言的處理流程與開發環境

### 識別字

-  在程式裡所使用到的每一個變數、常數、方法、類別及物件等，這些都是程式語言的識別字。
-  Java 的識別字必需符合下列規則。
  -  識別字的第一個字元必須是以字母「大小寫英文字母A至Z」、錢字符號「\$」或底線符號「\_」；
  -  識別字的其餘字元僅可由字母、數字、錢字符號「\$」或底線符號「\_」組合而成，不可包含空白；
  -  識別字不可以是保留字；
  -  識別字的長度沒有限制；
  -  識別字區分大小寫，因此大寫、小寫視為不同的識別字。





## 第2章 JAVA語言的處理流程與開發環境

### 方法名稱的習慣用法

-  **方法名稱：**習慣採用小寫字母，如 initial、start、run、display 等。
-  **如果方法名稱包含一個以上的英文字，則後續字的第一個英文字母則使用大寫。**
-  **例如：**displayArray、showLinkedList、quickSort、mergeSort 等等。

## 第2章 JAVA語言的處理流程與開發環境

### 關鍵字

-  「關鍵字」也叫做「保留字 (reserved word)」。
-  它是用來定義語言的語法。
-  不同的程式語言，所使用的關鍵字當然是不相同的。
-  「關鍵字」會出現在程式指令的特定位置，不同的關鍵字各有其不同的功能。






## 第2章 JAVA語言的處理流程與開發環境

### Java 的關鍵字

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

## 第2章 JAVA語言的處理流程與開發環境





### 文 字(1)

-  文字代表原始程式碼中的「原始 (primitive) 資料型態」、「字串 (String) 型態」與「空字元 (null)」的值。
-  「原始資料型態」包含「整數」、「浮點數」、「邏輯」與「字元」。
-  數值文字簡單的說就是代表數值的文字，可以區分為「整數」與「浮點數」。



## 第2章 JAVA語言的處理流程與開發環境

### 文 字(2)

-  整數的文字通常以十進位來表示。
-  在一般的程式語言中，除了十進位的表示法之外，還提供八進位與十六進位的表示法。
-  如果是使用八進位，只要在數字的開頭加一個 0 就可以了。
-  假如使用十六進位時，則要在數字的開頭加上 0x 或是 0X 。

## 第2章 JAVA語言的處理流程與開發環境

### 不同進位的數字文字範例

十進位	十六進位	八進位
257	0x101 0X101	0401
-2687	-0xA7F -0XA7F	-05177

## 第2章 JAVA語言的處理流程與開發環境

### 整數文字

-  整數文字會依據數值的大小，而有不同的內部儲存方式。
-  例如：宣告為 `int` 的整數變數，是以32位元儲存，因此其範圍是  
從 - 2,147,483,648 到 2,147,483,647
-  如果您要處理的整數的數值資料超過前述範圍，則需要使用長整數的資料型態 `long` 。



## 第2章 JAVA語言的處理流程與開發環境

### 整數文字

 long 是以 64 位元儲存，範圍

從 - 9223372036854775808( $-2^{63}$ )

到 9223372036854775807( $2^{63}-1$ )

 為了要標示它是一個長整數，必須要在數值的尾巴加上l或L。






## 第2章 JAVA語言的處理流程與開發環境

### 整數資料型態

中文資料型態	資料型態	佔用記憶體	所能代表的數值範圍
位元組	byte	8 bits	$-2^7 \sim 2^7-1$
字元	char	16 bits	$0 \sim 2^{16}-1$
短整數	short	16 bits	$-2^{15} \sim 2^{15}-1$
整數	int	32 bits	$-2^{31} \sim 2^{31}-1$
長整數	long	64 bits	$-2^{63} \sim 2^{63}-1$

## 第2章 JAVA語言的處理流程與開發環境

### 浮點數

-  另一種文字形式的數值為浮點數文字。
-  所謂的浮點數是在整數的個位數右側有一小數點，小數點以右的數字表示小於1的數值。
-  浮點數在電腦內部的儲存，也分為 32 位元與 64 位元。
-  前者為宣告為浮點數 float 的變數值，後者則是雙倍精確度的 double 資料型態。
-  其表示法都是依照 IEEE 的標準。




## 第2章 JAVA語言的處理流程與開發環境

### 浮點數資料型態

中文資料型態	資料型態	佔用記憶體	所能代表的數值範圍
浮點數	float	32 bits	$1.40\text{e-}45 \sim 3.4028235\text{e}38$
雙倍精準度 浮點數	double	64 bits	$4.9\text{e-}324 \sim 1.7976931348623157\text{e}308$





## 第2章 JAVA語言的處理流程與開發環境



Java 語言的要素：  
「字元」、「字串」與「邏輯」

## 第2章 JAVA語言的處理流程與開發環境

### 字 元

-  「字元」文字實際代表的是一個字元或者是一個逃脫字元所代表的字。
-  前者通常使用單引號「'」將文字括住。
-  後者則是以倒斜線後面接此字元的代表字或字碼。
-  所謂的代表字，像換行符號便是「`\n`」。



## 第2章 JAVA語言的處理流程與開發環境

### 逃脫字元所表示的控制符號(1)

字元組合	標準名稱	說明
\n	NL or LF (new line or line feed)	換行
\b	BS (back space)	游標倒退
\r	CR (carriage return)	游標回至行首
\f	FF (form feed)	印表機跳頁
\t	HT (horizontal tab)	水平跳格

## 第2章 JAVA語言的處理流程與開發環境

### 逃脫字元所表示的控制符號(2)

字元組合	標準名稱	說明
\\	\	倒斜線
\'	'	單引號
\"	"	雙引號
\ ddd	0ddd	八進位ASCII碼
\ udddd	0xdddd	統一碼字元

## 第2章 JAVA語言的處理流程與開發環境




### 一些字元文字的例子

-  `'h'` 代表小寫英文字母h。
-  `'%'` 代表百分比符號。
-  `'\t'` 代表水平跳格字元。
-  `'\\'` 代表倒斜線符號。
-  `'\"'` 代表單引號。
-  `'\u2152'` 代表統一碼的字元。
-  `'\177'` 代表八進位的ASCII碼的字元。



## 第2章 JAVA語言的處理流程與開發環境

### 字 串

-  「字串」則是由零個或是一個以上的字元所組成。
-  通常使用雙引號來括住字串。
-  在字串當中可以使用逃脫字元或統一碼的字元來表示。

## 第2章 JAVA語言的處理流程與開發環境

### 一些字串的範例(1)



`"\""` 代表只含雙引號的字串。



`""` 代表空的字串。






`"This is a string"` 代表本句英文的字串。



`"This ia a " +` 代表由本行與下一行的兩個字串銜接而成的字串"two-line string"。

## 第2章 JAVA語言的處理流程與開發環境



### 一些字串的範例(2)

-  "Title\tName\t\n" 在字串中使用逃脫字元來控制游標。
-  "Java測試..." 在字串中使用中文字。
-  "Java\u7a0b\u5f0f..." 以統一碼來表示中文字，內容與上一列相同。



## 第2章 JAVA語言的處理流程與開發環境

### 邏 輯

-  邏輯的內容值只有兩種情況，一種是代表邏輯值為真 (true)，另一種情況則是代表邏輯值為偽 (false)。
-  在前面介紹關鍵字時，有特別提到代表邏輯的文字一樣不可以當做識別字。

中文資料型態	資料型態	佔用記憶體	所能代表的數值範圍
布林邏輯	boolean	16 bits	true、false





## 第2章 JAVA語言的處理流程與開發環境



Java語言的要素：  
「分隔子」與「運算子」

## 第2章 JAVA語言的處理流程與開發環境

### 分隔子

-  分隔子在Java語言中所扮演的角色，是做為程式中元素與元素之間的區隔。
-  例如：在運算式中我們可以使用成對的小括號，將要優先運算的式子括住。
-  要表示複合程式區塊時，是用左右大括弧，將其中所包含的程式碼括住。
-  另外，多數的程式指令結束，要使用分號來與其他的指令區隔。







## 第2章 JAVA語言的處理流程與開發環境

### Java語言所使用到的分隔子

分隔子	說明
( )	用來定義方法中參數的串列、運算式的優先處理之運算、迴圈或條件指令之判斷式等等。
[ ]	用來定義陣列、陣列元素的存取。
{ }	用來分隔程式區塊、例如類別或是方法的主體、標示特定的程式區塊。
,	用來分隔多個參數的方法或是多個變數等的宣告。
;	用來區隔指令的結束。
.	物件與其成員的分隔，例如物件所屬的方法或成員變數。

## 第2章 JAVA語言的處理流程與開發環境

### 運算子

-  運算子多數是用來對變數或常數之間做運算。
-  例如我們可以將代表數值的變數之間進行一般的加、減、乘、除等數學運算。
-  或者是比較兩數值之間的大小。
-  或是將數值或字串指派給其他變數。

## 第2章 JAVA語言的處理流程與開發環境

### 運算式(1)

-  運算式的主要成元包含運算子 (Operator) 與運算元 (Operand)。
-  所謂運算子，指的是可以對運算元執行特定功能的特殊符號。



## 第2章 JAVA語言的處理流程與開發環境

### 運算式(2)









#### 運算子可以分成五類

-  指定 (Assignment) 運算子
-  算術 (Arithmetic) 運算子
-  條件 (Conditional) 運算子
-  關係 (Relational) 運算子
-  位元 (Bitwise) 運算子

## 第2章 JAVA語言的處理流程與開發環境

### 運算子之結合律與優先順序

-  每一種運算子都可以再細分為一元運算子與二元運算子。
-  一元運算子只需要一個運算元就可以運算。
-  二元運算子則需要兩個運算元才能夠運算。
-  運算子的結合律與優先順序為其重要特性。
-  優先順序用來決定同一式子擁有多個運算子時，每一個運算子進行運算的優先順序。
-  而結合律則決定了在同一運算式中，相同優先順序的運算子執行的順序。

## 第2章 JAVA語言的處理流程與開發環境



### Java語言要素：運算子的類別



## 第2章 JAVA語言的處理流程與開發環境

### 「指定運算子」



#### 指定運算子

將數值、文字、字串、計算式子，表示式或方法回傳的值等文字，指定給特定的變數。當然也可以是變數之間的指定，例如：將y變數的值指定給x，便是  $x=y$ 。其實就是算術中的等號運算的功能，在電腦語言中也是用等號(=)來表示。注意關係運算子的等於，是以兩個等號來表示。

結合指定運算子與算術、位元運算子稱為複合指定運算子，此為C/C++ 與 Java 語言特有的運算子，例如程式設計者常會鍵入 `sum = sum + 5`，有了複合指定運算子 `+=`，所以前面的指令可以寫成 `sum += 5`。

## 第2章 JAVA語言的處理流程與開發環境

### 「算術運算子」與「關係運算子」



#### 算術運算子

算術運算子用來執行一般的數學運算，包括負數(-)、加(+)、減(-)、乘(\*)、除(/)、餘數(%)、遞增(++ )及遞減(-- )等。



#### 關係運算子

比較運算子又稱為關係運算子，用於資料之間的大小比較，比較的結果可得到邏輯的 true 或 false。包括小於(<)、大於(>)、小於等於(<=)、大於等於(>=)、等於(==)與不等於(!=)。



## 第2章 JAVA語言的處理流程與開發環境

### 「條件運算子」



#### 條件運算子

當同一個運算式同時存在兩個以上的關係運算子時，每兩個關係運算子之間必須使用條件運算子連結，及關係運算的 例如：  
AND(&&)、OR(||)、NOT(!)和條件指定(?:)。

與 if-else 敘述有類似功能的條件運算子 (?:)，條件運算子的用法如下：

條件式 ? 值1 : 值2 ;

( 當條件式為 true 時指定「值1」，相反的，若條件式為 false 時，則指定「值2」。) )



## 第2章 JAVA語言的處理流程與開發環境

### 「位元運算子」



#### 位元運算子

位元 (Bitwise) 運算子可以分為兩類：位移 (Shift) 運算子與布林運算子：位元AND(&)、位元OR(|)、位元XOR(^)等等。位移運算子可以用來把一個整數的各個二進位元向左或是向右移，布林運算子則可以逐位元進行布林運算。





## 第2章 JAVA語言的處理流程與開發環境



### Java 程式的開發程序

## 第2章 JAVA語言的處理流程與開發環境

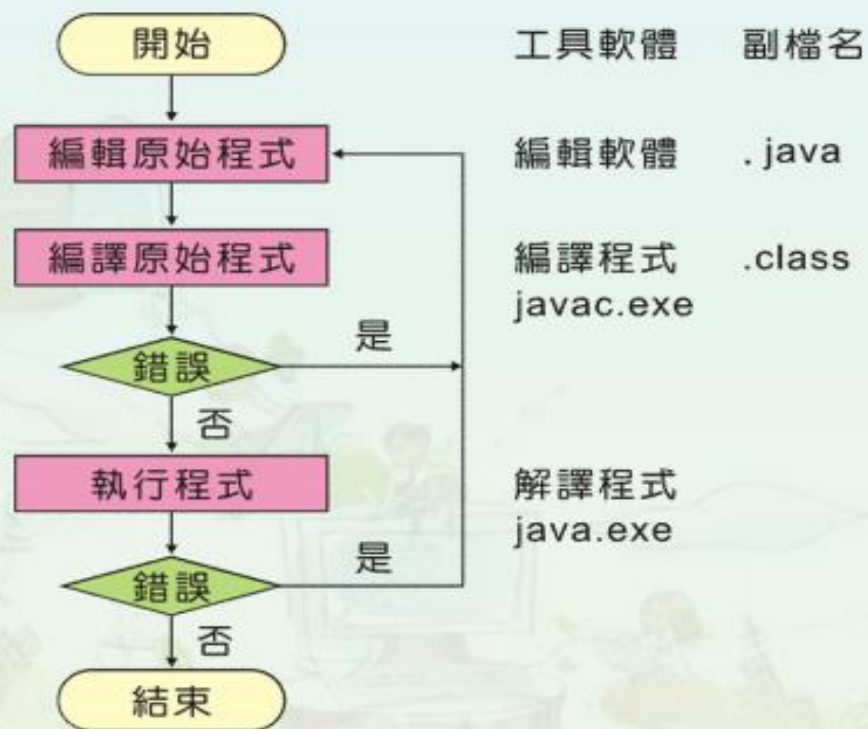
### Java 程式的開發程序

-  與一般編譯語言一樣，分為「編輯原始程式」、「編譯原始程式」及「執行程式」三個步驟。
-  在「編譯」及「執行」的步驟如果有錯誤，需要找出錯誤的地方，接著修改原始程式碼，然後再重新編譯與執行。
-  經過這三個步驟，便可以建立正確的可執行程式碼。
-  但是這並不表示程式在任何狀況下絕對可以執行。



## 第2章 JAVA語言的處理流程與開發環境

### Java 程式開發的流程







## 第2章 JAVA語言的處理流程與開發環境

### 編輯原始程式(1)

-  首先以「編輯程式」編輯 Java 原始程式碼 (source code)檔案，副檔名為.java。
-  這裡所提到的「編輯程式」，是指純文字編輯軟體。
-  例如，視窗的 notepad.exe 程式、「命令提示字元」下的 edit.com。

## 第2章 JAVA語言的處理流程與開發環境





### 編輯原始程式(2)

-  一般整合的開發環境，會提供編輯原始程式的環境。
-  有些整合的開發環境允許使用者設定自己的編輯程式。
-  完成程式編輯後，將它儲存成副檔名為 .java 的檔案。
-  注意檔案的名稱與 Java 程式的類別名稱一致，習慣上採用第一個字母大寫。



## 第2章 JAVA語言的處理流程與開發環境

### 編譯原始程式

-  將原始程式編譯 (compile) 成「位元組碼 (byte code)」，副檔名為.class。
-  這個階段需要使用到 JDK 的編譯程式 javac.exe，編譯程式提供許多參數來指定編譯時需要用到的功能選項。
-  在一般情況的使用，只要在原始程式所在的資料夾中執行，編譯程式名稱後留一個空白，接 Java 原始程式檔案的全名，即可進行編譯的工作。
-  所產生的位元組碼檔案會存放在目前資料夾，而且檔案名稱與原始程式檔案名稱一樣。

## 第2章 JAVA語言的處理流程與開發環境

### 執行程式(1)

-  完成程式編譯，接著就可以執行程式。
-  這個階段需要使用到 JDK 的相關程式，依程式的類別有所不同。
-  如果是 Java 應用程式 (Application)，則使用 `java.exe` 解譯程式。
-  假如是 Java 網頁程式 (Applet)，則使用 `appletviewer.exe` 程式來執行。



## 第2章 JAVA語言的處理流程與開發環境

### 執行程式 (2)

-  在一般情況的使用，只要在位元組碼檔案所在的資料夾中執行，在解譯程式名稱後留一個空白，接 Java 位元組碼的全名，即可執行該 Java 程式
-  如果是 Java 網頁程式，我們需要另外建立 html 網頁，使用 <Applet> 標籤將位元組碼檔案載入執行。執行時在 appletviewer 後留一個空白，接 html 網頁的檔案名稱，即可執行該 Java 網頁程式





## 第2章 JAVA語言的處理流程與開發環境



### Java 程式的開發環境

## 第2章 JAVA語言的處理流程與開發環境

### Java 程式的開發環境(1)

-  Java 程式的開發環境包括「JDK工具開發環境」、「簡易的整合開發環境」與「專業的整合開發環境」。
-  「JDK工具開發環境」是使用 JDK 所提供的工具，直接在 Windows 的「命令提示行」下或類似 UNIX 主機的提示字元的模式下進行編譯與執行工作，編輯程式則是我們自己選用的文字編輯軟體。

## 第2章 JAVA語言的處理流程與開發環境

### Java 程式的開發環境(2)



「簡易的整合開發環境」為免費的開發軟體，提供的功能有限，但是足夠用來學習 Java 語言。



「專業的整合開發環境」則是提供功能完整，可以實際開發大型專案的環境。相對的操作介面較為複雜，可以分為免費版與商業版兩大類。



## 第2章 JAVA語言的處理流程與開發環境

### JDK 工具開發環境



這是安裝好 Sun JDK 的開發環境，然後進行作業系統的「搜尋路徑」與「JDK環境變數」相關設定，就可以使用 JDK 工具與編輯程式來撰寫與開發 Java 程式。



Windows 的「命令提示行」下。



UNIX 主機的提示字元的模式下。

## 第2章 JAVA語言的處理流程與開發環境

### 簡易的整合開發環境(1)

-  提供從程式編寫、編譯、測式與執行的環境。
-  相對於第一種陽春的「JDK 工具開發環境」，可以算是有相當程度的改善。
-  常見「簡易的整合開發環境」有 Jeliot、Jedit、Jcreator、TextPad與UltraEdit 等。

## 第2章 JAVA語言的處理流程與開發環境

### 簡易的整合開發環境(2)



其中 TextPad 與 UltraEdit 又屬於這類當中較為簡單的整合方式，其主要功能是運用現有的文字編輯軟體來撰寫程式。







在編輯軟體中，以執行外掛程式的方式來編譯與執行程式，因此相當容易使用。



## 第2章 JAVA語言的處理流程與開發環境

### 專業的整合開發環境

-  常用的「專業的整合開發環境」有 Eclipse、NetBeans 與 Jbuilder 等。
-  前二者為開放原始碼軟體，後者則是 Borland 公司所開發的軟體。
-  這些「專業的整合開發環境」相當複雜，相對的功能較強，可以開發各類不同的應用程式。
-  本書選用的是 Eclipse 「專業的整合開發環境」。