



# Java Steps

*Release 1.0*

董少桓

September 21, 2011



# CONTENTS



# 簡介

## 1.1 Java 語言的特色

Java 原本是為了控制冰箱、冷氣、微波爐等家電用品而設計的程式語言。由於家電用品相當多樣，因此 Java 選用了一個與傳統的程式語言不一樣的執行模式：

- 傳統的程式語言在編譯後會產生 machine code（機器碼），然後直接在硬體上執行；
- Java 在編譯後則會產生 Byte Code 並間接的在 Java Virtual Machine（JVM）上執行。這個 JVM（Java 虛擬機器）其實是一個軟體，其功用是解譯並執行 Byte Code，而 JVM 仍然是在硬體上執行。

Java 也支援物件導向程式設計（Object-Oriented Programming），所謂「物件」，簡單的說有「屬性」也有「方法」，例如冷氣機的「屬性」可以包括：「開關」及「溫度」；而「方法」則可以包括：開機、關機及設定溫度等，為的是讓程式設計師，可以比較容易的使用物件導向程式語言，撰寫控制與應用我們生活周遭許多「物件」的程式。

因為 JVM 是軟體，所以 Java 也有跨平台的特性：只要為不同的處理器或作業系統設計其專屬的 JVM，Java 的程式便可以不需改寫的在這些處理器或作業系統上執行。這便是「Write once, runs everywhere 或一次編譯、到處執行」的由來。

Java 的設計搭上了全球資訊網的順風車，原因是 Java 的設計團隊可以寫一個能夠在瀏覽器中執行的 JVM，而讓 Java 的程式可以透過網路下載至瀏覽器中執行。這個「網路」＋「物件導向」的特性讓 Java 瞬間爆紅。

除了跨平台、物件導向、可透過網路動態的載入及執行程式等功能之外，Java 還支援多執行緒、例外狀態處理與自動記憶體回收的功能。多執行緒讓一個程式可以執行數個工作；例外狀態處理讓處理例外的程式碼也能夠物件化；而自動記憶體回收則讓程式設計師免除了使用低階的指標（pointers）來設計資料結構及管理記憶體的負擔。這個特色成了 C 語言程式設計師的福音，因為它可以為程式設計師減少許多不容易 debug 的錯誤。

然而，Java 的這些優點還是需要附上相對的代價：

1. Java 的執行速度比 C 慢；
2. Java 的程式碼雖然嚴謹、有 robust behavior、能在 compile-time 找出所有的 type error，但是其程式碼的長度也相對的增長。

## 1.2 安裝 JDK

在編譯與執行 Java 程式前，你的電腦必須先安裝 JDK (Java Development Kit)：

- 可以從這裡下載新版的 JDK <http://java.sun.com/javase/downloads/index.jsp>

在安裝完成後，也需要完成 path 及 classpath 的設定：

- path=C:\Program Files\Java\jdk1.6.0\bin;....
- classpath=.;C:\Program Files\Java\jdk1.6.0\lib;....

請注意：以上路徑中的 jdk1.6.0 會因版本的不同而異。此外，在設定 classpath 時要特別注意在一號的右邊要輸入這個「.」。這個點的意義是目前的目錄（current directory），也是執行 Java 程式時用來搜尋執行檔的目錄。

## 1.3 編譯及執行 Java 程式

有兩種方式可以編譯及執行一個 Java 程式。第一種是使用程式開發環境 (program development environment)，例如：Eclipse <<http://eclipse.org/>>\_；另一種則是使用一般的程式編輯器。以下是使用「記事本」寫 Java 程式時所需要進行的三個步驟：

1. 使用「記事本」輸入以下的程式並將檔案命名為 EnglishExam.java (注意：附檔名必須是.java 而不是.txt，而這個檔案的主檔名必須與 public class 後面的 EnglishExam 相同)：

```
public class EnglishExam {  
    public static void main(String argv[]) {  
        System.out.println("Your score is 97.");  
    }  
}
```

2. 執行「命令提示字元」並將目錄切換至儲存 EnglishExam.java 的目錄，然後執行：

```
javac EnglishExam.java
```

執行這個 javac 指令就是執行 Java 的編譯器 (compiler)，其結果是在同樣的目錄產生一個 EnglishExam.class 的 byte code 檔。

3. 上面的步驟如果有編譯錯誤則繼續修改程式。如果沒有編譯錯誤則可以執行：

```
java EnglishExam
```

執行後 Your score is 97。便會顯示在螢幕上。

在開發 Java 程式的過程中，有可能發生編譯錯誤 (compile-time error)。這時便需要再次的使用編輯器修改錯誤，直到沒有任何的編譯錯誤為止。編譯完畢之後，在程式執行時也有可能發生 run-time error。同樣的這時又需要使用編輯器修改、編譯、執行、除錯，直到沒有錯誤為止。

一般的 Java 程式都是由一或多個類別 (class) 所組成，其中的一個類別至少要有一個命名為 public static void main 的方法 (method)，而這個程式就是

由 `main` 開始執行。（透過網路瀏覽器執行的 Java applet 不適用此規則。）

上例中的 `public class EnglishExam` 是指定 `EnglishExam` 這個類別是 `public` 是公用的，也就是可以被程式中其他的類別引用。而 `public static void main(String argv[])` 的意義是：

1. `public`：指定 `main` 為一個可以被其他類別使用的 `public method`；
2. `static`：指定 `main` 為一個類別方法（`static method`），一個類別方法隸屬於一個 `class`；
3. `void`：代表 `main` 執行完畢後回傳的型態，因為 `main` 沒有回傳任何數值，因此它的回傳型態是 `void`；
4. `String argv[]`：指這個方法的輸入參數是 `argv[]` 而 `String` 則是它的型態。`main` 的輸入參數 `String argv[]` 可以在執行一個 Java 程式時將字串（`String`）資料輸入這個程式。例如在編譯以下的程式之後：

以「命令提示字元」執行：

```
java HelloJava Basic C++
```

便會呼叫 `System.out.println` 並輸出：

```
Hello Basic C++
```

這個程式的 `argv[]` 代表 `argv` 這個變數是一個陣列，而 `argv[0]`、`argv[1]` 則取用 `argv` 內第 0、1 個儲存格的內容。

Java 程式中 用大刮號 `{ }` 標示的 Block（區塊）是用來組織程式層次關係的語法。

例如上例的程式就有兩個區塊，一組用來標示 `class` 的區塊，另一組則用來標示 `main` 的區域。區塊中可以包含其他的區塊，在撰寫程式時也應注意要把區塊的內容往右縮排。一組用來標示類別的區塊內，可以有數個變數與方法。而一組用來標示方法的區塊內可以有一或多句以「`;`」結束的程式碼。這些程式碼共同構成了這個方法的 `body`。

為 Java 程式中使用的名字命名，有一個不成文的規定：**類別名稱的第一個字母要用大寫**。方法或變數的第一個字母則是小寫，若有數個字合併時則後續的字的第一個字母也習慣用大寫。



# JAVA 的變數、方法與型態

一個電腦程式基本上是由兩部分組成：

1. 資料：這部分的程式碼要為所處理的資料命名、指定其型態、並存放於記憶體中；
2. 處理：這部分的程式碼要使用撰寫在方法（method）中的運算式、條件判斷式、迴圈，來存取資料、計算結果、然後輸出。

這兩部分的內容（程式碼），不但是機器要知道如何執行，更要讓人（程式設計師）容易寫，也容易讀。而讓程式碼易寫、易讀的最基本的方法，就像收拾家中的衣櫃一般：按照種類與性質，分別放置整齊。

Java 語言整理程式碼的最基本的單元稱做「類別」。一個類別可以有儲存資料的「變數」與負責處理資料的「方法」。

「類別」可以提供兩種整理程式碼的方式：一種是以程式的邏輯結構當作分類的方式（俗稱的結構化程式設計）；另一種則是以物件的種類來歸類。而有的時候，一個類別也可以同時提供這兩種方式。

以程式的邏輯結構（例如將迴圈的邏輯放入一個方法中，或將一個判斷分數高低的邏輯放入一個方法中）來分類時，所使用到的變數稱為類別變數或 static field，而所用到的方法稱為類別方法或 static method。稱做 static（靜態）的原因是，這類的變數與方法是在程式開始執行時便在記憶體中產生了，而且它們的壽命一直到程式結束時才結束。

以物件來歸類的話，則是將類別內的程式碼，看成是產生這種類別的物件的「規格」。由於只是規格，所以只有在使用這個類別製造物件時，所對應的記憶體才會產生。而這種在程式執行時「動態」產生的物件實例中的變數與方法，稱為實例變數（instance variable），與實例方法（instance method）。

類別方法或實例方法，也需要自己有儲存資料的地方，而在類別方法或實例方法中儲存資料的變數，都叫區域變數（local variable），意思是在一個方法所屬的區域內才可以使用的變數。

Java 如何分區呢？主要是使用大刮號，例如：

```
{
    {
        ...
    }
    {
        ...
    }
    ...
}
```

程式語言的句子與一般說話的語言一樣是由基本的字詞（names）組合而成。Java 為這些字詞命名的規定是一個字詞可以包含一個或多個英文字母、數字、\_ 及 \$ 所組成的字元，而第一個字元不可以是數字。

Java 語言有 public, class, void, if, while, for 等保留字 <[http://download-ltnw.oracle.com/javase/tutorial/java/nutsandbolts/\\_keywords.html](http://download-ltnw.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html)>\_。除了開始認識這些保留字的意義與用法之外，程式設計師所要學習的第一件事，就是為儲存資料的 **變數** 與執行處理的 **方法** 命名。

**變數**（variable）是程式中的一種字詞。一個變數有一個 **名字**（name）、一個 **資料值**（value）、一塊儲存資料值的 **記憶體** 以及這個資料值的 **型態**（type）（如 int、double 等）。由於一個變數的型態，定義了這個變數的值所需要的記憶體的大小，所以一個 Java 程式在編譯時，就可以知道如何為這些變數，在執行時，配置適當大小的記憶體空間，以存放這些變數的值。

整的來說，Java 有三種變數：

1. **區域變數** (local variable) ：宣告在方法內或參數部分的變數；
2. **類別變數** (class variable or static field) ：在一個類別中以 static 宣告的變數；
3. **實例變數** (instance variable or non-static field) ：在一個類別中沒有使用 static 宣告的變數。

Java 也有兩種方法：

1. **類別方法** (class method or static method) ：這種方法以 static 宣告。呼叫的方式是 C.m(...)，其中 C 是類別名稱，m 是方法名稱，... 則是 0 至多個傳入的參數。
2. **實例方法** (instance method or non-static method) ：這種方法不以 static 宣告，隸屬於一個類別所產生的實例。呼叫的方式是 o.m(...)，其中 o 是這個類別或其子類別的實例，而 m 是其方法名稱，... 則是 0 至多個傳入的參數。

Java 之所以有種類這麼多的變數與方法，是因為 Java 同時支援結構化（例如：C 與 Basic）與物件導向兩種普遍使用的程式設計方式。撰寫結構化程式時需要使用類別變數與類別方法。類別變數在概念上與結構化程式語言的全域變數 (global variable) 一致；而類別方法在概念上則與結構化程式語言的函式 (function) 或程序 (procedure) 一致。實例變數、實例方法，則屬物件導向程式設計的功能。一般的 Java 程式可以同時使用結構化與物件導向並存的方式設計程式。

這份講義介紹 Java 結構化程式設計的語法及語意，另一份講義《Java 物件導向程式設計》則介紹 Java 物件導向程式設計的功能。

此外，Java 變數的型態也有兩大類：

1. primitive type ，包括： `int`、`double`、`boolean`、`char` 等。
2. reference type ，包括：
  - (a) 類別型態：經由類別 (class) 的宣告而得到。如果 Car 是一個類別，而 aCar 是一個這個類別的變數，則 Car 便是 aCar

的型態 (type)。之所以稱為 reference type，是因為 aCar 這個變數在記憶體中的位置，實際上是存著指向 (reference) 一個 Car 實例的地址。

- (b) 介面型態：經由介面 (interface) 的宣告而得到。
- (c) 陣列 (array) 型態。
- (d) enum 型態：一種特別的類別宣告方式，用於宣告月份、一週的七天等。

## 區域變數與基本資料型態

區域變數是一個方法的參數或是宣告在一個方法的區塊中。以下是宣告區域變數的幾個範例，其中 `int` 代表整數，而 `double` 代表倍精準浮點數，宣告的意義是告訴編譯器一個變數的型態是什麼：

```
public class EnglishExam {
    public static void main(String argv[]) {
        // argv 是傳入參數，同時也是一種區域變數

        // 宣告三個區域變數
        int vocabulary;
        int grammar;
        int listening;
        ...
    }
}
```

多個變數的宣告，可以合併在一行：

```
public class EnglishExam {
    public static void main(String argv[]) {
        // 將以上三個區域變數，合併在一行
        int vocabulary, grammar, listening;
        ...
    }
}
```

宣告變數時，也可以同時指定數值：

```
public class EnglishExam {
    public static void main(String argv[]) {
        // 在宣告時也將數值存入這三個區域變數中
        int vocabulary = 24;
        int grammar = 26;
        int listening = 33;
        ...
    }
}

public class EnglishExam {
    public static void main(String argv[]) {
        // 也可以這樣寫：
        double vocabulary = 22.5, grammar = 25.4, listening = 32.5;
        ...
    }
}
```

有了變數之後，可以使用設值運算符號（=），將數值存入區域變數中，如果一個區域變數尚未被存入數值，則其預設值（default value）會被存入，而數字的預設值是 0。例如：

```
public class EnglishExam {
    public static void main(String argv[]) {
        int vocabulary, grammar, listening;
        int score;

        vocabulary = 22;
        grammar = 26;
        score = vocabulary + grammar + listening;

        System.out.print("The score of the exam is ");
        System.out.println(score);
        // listening 的預設值是 0，所以印出 48
    }
}
```

以上程式碼執行的結果為：

```
The score of the exam is 48
```

Java 的註解是以 // 或 /\* \*/ 表示，例如：

```
// 這是註解
/*
    這也是註解
    這還是註解
*/
```