# Java Steps: Learning Java Step by Step

***Release 1.0***

董少桓

September 21, 2011

# CONTENTS

CHAPTER

ONE

# 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 machince 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＝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 ；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** 。

CHAPTER

TWO

# 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 。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。

CHAPTER

THREE

# 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
                ...
        }
}
```

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
*/
```

CHAPTER

FOUR

# 螢幕輸出及鍵盤輸入

螢幕輸出有幾種方式。第一種是前面章節已經使用過的 System.out.print 及 System.out.println。這兩種方法的的差別是前者沒有換行，而後者有換行。如果有數個資料需要一起印出時，則可以使用 + 進行串接。例如：

```
public class Show {
        public static void main(String argv[]) {
                int design, acting;

                design = 3;
                acting = 5;
                System.out.println( "Design is " + design + "and acting
        }
}
```

第二種方式是在 J2SDK5 之後才支援 [1] 。這個方式與 C 語言的 printf 功能類似。例如：

```
System.out.printf("Today is %s, %d.\n", "January", 18);
// %s 的位置替換成 January 這個 String
// %d 的位置替換成 18 這個整數
// \n 表示換行符號
```

顯示：

[1] System.out.printf(), http://www.java2s.com/Code/JavaAPI/java.lang/System.out.printf.htm

```
Today is January, 18
```

例如：

```
double score = 92.345
System.out.printf("My score is %.2f.\n", score);
// %.2f 的意義是小數點以下取兩位，並四捨五入。
System.out.printf("My score is %6.2f.%n", score);
// %6.2f 的意義是：包括小數點共 6 位，小數點以下取兩位，
// 並四捨五入。所以 9 的左邊多空一格。
```

顯示：

```
My score is 92.35.
My score is  92.35.
```

鍵盤輸入則可以透過 [2] 。例如：

```
// 使用時先載入 Scanner 所屬的 package
import java.util.*;  // * 的意義是 java.util 內所有的類別

// 定義物件：
Scanner scanner = new Scanner(System.in);

// 輸入字串：
String name = scanner.nextLine();

// 輸入整數：
int score = scanner.nextInt();

// 輸入 double
double height = scanner.nextDouble();

// 輸入 float
float weight = scanner.nextFloat();
```

以下是一個完整的範例：

[2] java.util.Scanner, http://www.java2s.com/Code/JavaAPI/java.util/Scanner.htm

```
import java.util.*;

public class EnglishExam {
        public static void main(String argv[]) {
                int vocab, grammar, listen, score;

                Scanner scanner = new Scanner(System.in);
                String name = scanner.nextLine();
                vocab = scanner.nextInt();
                grammar = scanner.nextInt();
                listen = scanner.nextInt();
                score = vocab + grammar + listen;
                System.out.printf("The total score of %s is %d.%n", name
        }
}
```

CHAPTER

FIVE

# 算術運算式

使用算術運算式要注意的是運算的優先次序。例如：先乘除、後加減。如果不記得運算的優先次序，那麼最簡便的方法是使用（）也就是指定內層的刮號先執行。以下是算術運算式的幾個範例：

使用算術運算式另一項要注意的是型態的轉換，也就是在一個算術式中同時有整數與浮點數時，Java 會將整數先轉換成浮點數然後再進行運算。

例如：

```
int i = (3 + 4) /  **3**  // 兩個整數相除，結果仍是整數，小數部分捨棄。傳回 2
double f;
f = (3 + 4) /  **3.0**    // 7 轉型成浮點數，然後與浮點數 3.0 相除。傳回 2.33
System.out.println(i);
System.out.println(f);
```

印出：

```
2
2.3333333333333335
```

如果需要指定轉換的型態，則可以使用以下的方式：

至於 i、f 的原來的值還是不變。

如同一般的算術運算式，設值運算式（=）也會傳回值。例如：

```
x = 3          // 傳回 3
y = (x = 3)    // 因為 x = 3 傳回 3, 所以 y 的值也設成 3
```

自動將資料範圍較小的型態轉為資料範圍較大的型態，稱為自動轉型 (promotion)。Java 資料型態範圍之大小次序為：

```
byte < short < int < long < float < double
```

以下的例子會出現 possible loss of precision 的錯誤：

這個錯誤是因為 a 會自動轉型成較大的 long 再跟 b 相加，但是 int 型態的 c 放不下 long 的資料。此時就要透過強制轉型 (casting)，將 a 轉型成 int：

```
int c = (int)(a + b);
```

當資料型態由小轉為大時，會自動轉型；當資料型態由大轉為小時，則需強迫轉型。

CHAPTER

SIX

# 類別變數與類別方法

非物件導向程式語言（例如：C），程式設計師主要是使用函式（function、全域變數與區域變數將一個大的程式分割成幾個小的部份，以簡化程式的撰寫。這些觀念在 Java 中仍然可以使用，而使用的方式是透過類別變數與類別方法。

舉例而言，如果要為英文檢定考試寫一個計算成績的程式，那麼這個程式應該有一個計算成績的方法。例如：

```
public class EnglishExam {
        public static int englishScore(int v, int g, int l) {
                return v + g + l;
        }
        public static void main(String argv[]) {
                System.out.print("The score of the exam is ");
                System.out.println(englishScore(24, 27, 32));
        }
}
```

執行結果：

```
The score of the exam is 83
```

public static int englishScore(int v, int g, int l) 定義了 englishScore 這個類別方法，這個方法有三個命名為 v, g, l 的輸入參數，它們的型態都是 int。這個方法的輸出型態也是 int，也就是會使用 return 傳出一個 int 的值 (v + g +

l)。Java 使用 static 這個保留字來定義類別方法。因為這種方法是靜態的，也就是在程式執行時，呼叫這個方法的程式碼，一定都會執行相同的方法。

在 main 中的 System.out.println(englishScore(24, 27, 32)) 將 24, 27, 32 傳入 englishScore 中，並依序成為 v, g, l 三個輸入參數的值，而這三個數相加的結果 83 會繼續的傳入 System.out.println，然後顯示在螢幕上。一個方法的輸入參數也是那個方法的區域變數。所以 v, g, l 三個輸入參數也是 englishScore 的區域變數。

除了直接將數值傳入方法中以外，還可以將變數或其他也有傳回值的式子，寫在方法呼叫中傳入的位置。例如：

```
public class EnglishExam {
        public static int englishScore(int v, int g, int l) {
                return v + g + l;
        }
        public static void main(String argv[]) {
                int a = 3, b = 4;
                System.out.print("The score of the exam is ");
                System.out.println(englishScore( **a, b, a + b** ));
        }
}
```

執行結果：

```
The score of the exam is 14
```

Java 會在得到 a, b, a + b 的數值後，才將 3, 4, 7 傳入 englishScore 中。也就是先得到數值再傳入，然後 v, g, l 便使用傳入的數值生成三個區域變數。這個特性稱為 call-by-value **或傳值呼叫** 。v, g, l 三個參數之所以也是區域變數，因為這三個變數的可見範圍（scope）只包含 englishScore 的區塊。

如果一個方法沒有傳回值，那麼這個方法的輸出型態便是 void。例如：

```
public class EnglishExam {
        public static  **void displayScore** (int v, int g, int l) {
                System.out.print("The score of the exam is ");
                System.out.println(v + g + l);
```

```
        }

        public static void main(String argv[]) {
                displayScore(24, 27, 32);
        }
}
```

執行結果：

```
The score of the exam is 83
```

displayScore 這個方法將字串顯示在螢幕上，不需要傳回值，因此它的輸出型態是宣告成 void，而 main 的輸出型態也是 void。

不同的類別中也可以定義同名的方法。這個功能稱做 overloading 。而 Java 是以 **類別名稱. 方法名稱 (0 或多個參數)；** 呼叫宣告在不同類別的類別方法。例如：

```
public class Exam {
        public static void main(String argv[]) {
                int voc = 3, grammar = 7, listen = 8;
                System.out.print("The score of the english exam is ");
                System.out.println(EnglishExam.displayScore(voc, grammar
                System.out.print("The score of simple english exam is ")
                System.out.println(SimpleEnglishExam.displayScore(voc, g
        }
}

class EnglishExam {
        public static int displayScore(int v, int g, int l) {
                return v + g + l;
        }
}

class SimpleEnglishExam {
        public static int displayScore(int v, int g, int l) {
                return v + g + 0;
        }
```

```
}
```

執行結果：

```
The score of the english exam is 18
The score of simple english exam is 10
```

一個類別中也可以有同名的方法，但是他們必須有不同的輸出入型態。例如：

```
public class Exam {
        public static void main(String argv[]) {
                int voc = 3, grammar = 7, listen = 8;
                System.out.print("The int score of the exam is ");
                System.out.println( **EnglishExam.displayScore(3, 7, 8)*
                System.out.print("The double score of the exam is ");
                System.out.println( **EnglishExam.displayScore(3.0, 8.0,
        }
}

class EnglishExam {
        public static int  **displayScore(int v, int g, int l)**  {
                return v + g + l;
        }
        public static double  **displayScore(double v, double g, double
                return v + g + l;
        }
}
```

執行結果：

```
The int score of the exam is 18
The double score of the exam is 18.0
```

另一個 overloading 的例子是：十。十可以用來將數字相加，也可以將字串合併。例如：

```
int a = 4, b = 5;
System.out.print(3 + a + b);
```

執行結果：

```
12
```

例如：

```
.. code-block:: java
```

String a = ``xy'', b = ``Z''; System.out.print(``3'' + a + b);

執行結果：

```
3xyz
```

使用類別方法在程式中有許多好處：

1. 增加程式碼的再用性：同樣的計算步驟，只需要透過呼叫類別方法便可以重複使用。
2. 讓程式碼的細節，被隱藏在類別方法中：程式設計師在完成類別方法的撰寫後，便只需要知道那個類別方法的輸入、輸出與功用即可，而不用擔心執行的細節。
3. 容易除錯：除錯的過程可以一個類別方法、一個類別方法的進行，容易找出錯誤的根源。
4. 容易擴充類別方法內程式碼的功能：只要在類別方法內擴充其功能，而不用在每次呼叫時都重複的擴充。例如以下的程式碼擴充了成績的計算方式，所有 displayScore 的呼叫的計算結果都同步改變：

除了使用 static 宣告類別方法外，還有也是使用 static 宣告的類別變數。以下是一個在程式中內建三筆考試成績的資料，呼叫 displayScore 計算成績後，將三筆資料加總並存入 total 這個類別變數中的範例：

```
.. code-block:: java
```

public class Exam { public static int total = 0;

public static void main(String argv[]) { total = displayScore(3, 4, 5); // total = 12 total = total + displayScore(4, 5, 6); // total

= 27 total = total + displayScore(1, 2, 3); // total = 33 System.out.print(``The total score is "); System.out.println(total);

}

public static int displayScore(int v, int g, int l) { return v + g + l;

}

}

執行結果：

```
The total score is 33
```

程式設計師也可以使用不是定義在自己類別中的類別變數，而 Java 是以 **類別名稱. 變數名稱** 使用定義在其他類別中的類別變數。以下便是一種將 total 宣告在另一個類別 EnglishExam 中的寫法是：

```
public class Exam {
        public static void main(String argv[]) {
                EnglishExam.computeScore(3, 4, 5);
                EnglishExam.computeScore(4, 5, 6);
                EnglishExam.computeScore(1, 2, 3);
                System.out.print("The total score is ");
                System.out.println(EnglishExam.total);
        }
}

class EnglishExam {
        public static int total = 0;

        public static void computeScore(int v, int g, int l) {
                total = total + (v + g + l);
        }
}
```

執行結果：

```
The total score is 33
```

以下則是一個為考試成績的計算，加入權重的範例。在這個範例中是以 Exam.wV, Exam.wG, Exam.wL 來使用這三個類別變數：

```
public class Exam {

        public static double wV = 0.3, wG = 0.3, wL = 0.4;

        public static void main(String argv[]) {
                int voc = 3, grammar = 7, listen = 8;
                System.out.print("The score of the english exam is ");
                System.out.println(EnglishExam.displayScore(voc, grammar
                System.out.print("The score of simple english exam is ")
                System.out.println(SimpleEnglishExam.displayScore(voc, g
        }
}

class EnglishExam {
        public static double displayScore(int v, int g, int l) {
        return v * Exam.wV + g * Exam.wG + l * Exam.wL;
        }
}

class SimpleEnglishExam {
        public static double displayScore(int v, int g, int l) {
                return v * Exam.wV + g * Exam.wG + 0;
        }
}
```

執行結果：

```
The score of the english exam is 6.2
The score of simple english exam is 3.0
```

類別變數與區域變數，在變數的可用「區域」與存在的「時間」上都不相同。類別變數若是定義為 public，則它的可用區域便包括整個程式，而且在整個程式執行時都存在。區域變數則是在程式執行到一個區塊或方法內

時，那個區塊或方法的區域變數才存在，一旦離開那個區塊或方法，便消失了。因此區域變數的可用區域，只在定義該區域變數的區塊或方法內。

以下是一個「計算蛋與水果總價」的程式及其執行過程的動畫：

```
class Market {
        static int sEgg = 5, sFruit = 20;
                static int getMoney(int nEgg, int nFruit) {
                return sEgg * nEgg + sFruit * nFruit;
        }
}

public class Ex1 {
        public static void main(String argv[]) {
                int egg = 20, fruit = 30;
                System.out.print("Money:");
                System.out.println(Market.getMoney(egg, fruit));
        }
}
```

執行結果：

```
Money:700
```

Media:Ex1new.swf 觀看執行過程