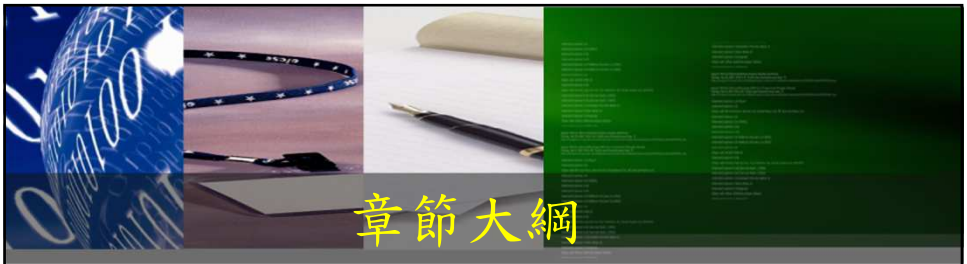
第二章 Java從零開始
(Java程式的基本結構)

課前指引

在本章中，我們將透過一個非常簡單的Java程式來說明Java的程式結構。



章節大綱

- 2.1 最簡單的Java程式範例
- 2.2 註解(comment)
- 2.3 package區
- 2.4 import區
- 2.5 類別區
- 2.6 Java程式的進入點main(…)
- 2.7 Java的敘述(statement)
- 2.8 println()輸出方法的簡易使用法
- 2.9 自由格式與空白字元
- 2.10 Eclipse IDE的活用

2.1 最簡單的Java程式範例

- 下面是一個簡單的Java程式範例，請逐字將之輸入到副檔名為『.java』的檔案中
  - 同時，Java程式的主檔名與程式內容有關，我們將在後面解釋其相關性。
    - 若您的IDE已經幫您建立了某些預設內容，請先將它刪除後再輸入
  - 範例2-1：ch2_01.java（隨書光碟 myJava\ch02\ch2_01.java）

3

2.1 最簡單的Java程式範例

```
1   /*
2    檔名:ch2_01.java
3    功能:簡單的Java程式範例
4    */
5   package myJava.ch02;
6   import java.lang.*;
7
8   public class ch2_01           //主類別
9   {
10      public static void main(String args[])
11      {
12          System.out.println("歡迎使用Java!");
13      }
14  }
15
16  class MyClass                 //一般類別
17  {
18  }
```

4

2.1 最簡單的Java程式範例

- 執行結果：

粗體代表使用者的輸入

```
C:\myJava\ch02>javac ch2_01.java
C:\myJava\ch02>cd\
C:\>java myJava.ch02.ch2_01
歡迎使用Java!
C:\>
```

非粗體代表程式的輸出

- 範例說明：
  - (1)程式真正的執行結果只有『歡迎使用Java!』，其他都是操作命令。
    - 其中javac ch2_01.java是編譯ch2_01.java程式
      - 它將會產生ch2_01.class及MyClass.class兩個類別檔。
    - 而java myJava.ch02.ch2_01是要求JVM執行ch2_01.class，必須在C槽根目錄下執行。
    - 您也可以如同第一章使用兩個Dos視窗分別進行編譯與執行。
    - 之後，我們將省略這些操作步驟，直接印出執行結果。

5

2.1 最簡單的Java程式範例

- (2)雖然範例2-1說明了Java Application的基本結構如下。

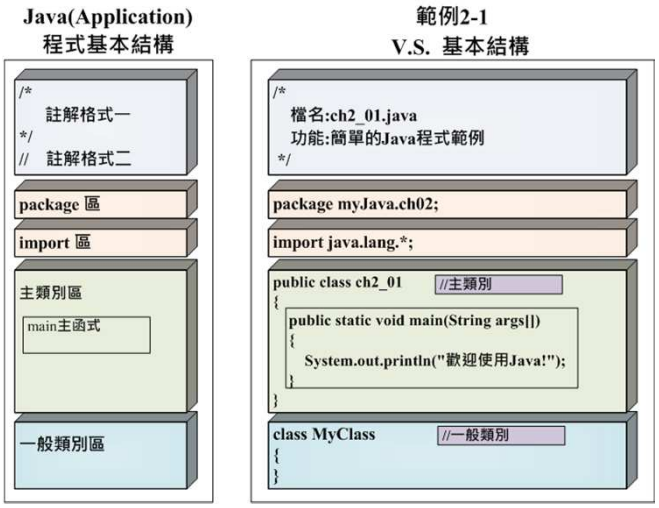


圖2-1 Java Application原始程式的基本結構

6

2.2 註解(comment)

- 在程式的任何地方都可以加入註解
  - 註解對於編譯器而言是沒有意義的
  - 沒有註解並不會影響程式的正確性。
  - 註解文字可以當作說明該程式或程式片斷之用，善用註解文字將有助於日後維護程式時，快速了解該段程式的功用。
- Java的註解符號有三種如下：
  - 區塊註解符號『/*…*/』
  - 單行註解符號『//』
  - 文件註解符號『/**…*/』

7

2.2 註解(comment)

- 區塊註解符號『/*…*/』
  - **在『/*』到『*/』之間的所有文字（可跨行）都將被編譯器忽略**。
    - 例如範例2-1的第1~4行為註解
  - 區塊註解具有跨行效果，為了明顯起見，我們會多加上一個『*』來對齊（但有無加星號都不影響註解效力）
    - 例如第1~4行的註解改寫如下較佳：

```
/*
 *   檔名:ch2_01.java
 *   功能:簡單的Java程式範例
 */
```

8

2.2 註解(comment)

- 單行註解符號『//』
  - **凡是『//』之後的整行文字都會被視為註解**。
  - 例如範例2-1的第8、16行的後半段為註解。
    - 而第1~4行的註解也可以改寫如下。

```
//--------------------------------
//    檔名:ch2_01.java
//    功能:簡單的Java程式範例
//--------------------------------
```

小試身手2-1
將範例2-1的第1~4行刪除，並修改為如上的單行註解格式，然後重新編譯與執行，看看執行結果是否相同？

9

2.2 註解(comment)

- 文件註解符號『/**...*/』
  - 文件註解只有在使用Javadoc工具製作說明文件時有用
  - 文件註解對於編譯器而言，與區塊註解完全相同，因為都是由『/*』開始，到『*/』結束。
  - Javadoc是一個程式，它可以將您的Java程式說明萃取成HTML檔案，這些HTML檔與JDK說明文件的HTML具有相同的外觀，其中包含類別、介面、變數、方法等相關資訊。
  - 在本書中，我們將不使用這種註解格式
    - http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

10

2. 3 package區

- package區是用來宣告該檔案中的類別屬於哪一個Package（類別庫），必須出現在程式的最上方，且只能有一個。
  - 一個Java程式檔最多只能隸於一個Package，以便於管理。
  - 如果不想要將該程式隸屬於任何Package，則可以省略package區的宣告
    - 此時程式檔將隸屬於『預設的Package』。

11

2. 3 package區

- Package（也可翻為程式包裹）是管理類別(Class)的容器，在Java中，管理套件採用檔案系統的目錄結構方式來管理，重點如下：
  - 1. 同一個Package內的類別名稱不能重複，因此有助於跨檔案開發程式時的管理。
  - 2. 相關類別通常被放在同一層目錄，以便方便管理及存取。
  - 3. 可開發屬於自己或公司的類別程式庫。
  - 4. 程式設計人員可以很容易取用所需要的類別。

12

2.3 package區

- 套件管理以目錄方式來管理，是基於檔案系統的習慣做為出發點，而Java也延續此一特點將Package對應到所屬的目錄結構。
  - 舉例來說，範例2-1位於C:\myJava\ch02\目錄內
  - 我們可以將目錄路徑『myJava\ch02\』的目錄階層符號『\』改為『.』作為Package名稱
  - 亦即『myJava.ch02』
  - 可確保Package的唯一性。

13

2.3 package區

- 是否宣告或如何宣告Package在編譯時（使用javac.exe編譯）是毫無影響的，只有在JRE執行時（使用java.exe執行）才會有所影響。
  - 範例2-1宣告了Package，因此執行時，必須先使用「cd\」切換到根目錄然後才能執行ch2_01.class類別檔，否則JRE會找不到ch2_01.class類別檔。
  - 第3~10章的程式中，我們都將package宣告為myJava.chxx
    - 因為我們的程式將會放在myJava\chxx目錄中
    - 而編譯後的class檔案將與java原始檔位於同一目錄下。

14

2.4 import區

- 使用現成的類別(class)可減少程式開發的時程。
  - 想要引用這些類別，就必須先將其在import區中宣告
  - 且import區必須位於package區之下，所有類別區之上，以便讓編譯器或執行器一開始就能夠找到類別所在位置。

15

2.4 import區

- Java的類別使用類別庫(package)來管理，而類別庫則是採用目錄階層式管理，為了方便使用，我們可以採用『*』代表該階層的所有類別
  - （但不包含子目錄/子類別庫內的類別）
- import java.lang.*;
  - 代表的就是引用java.lang（這是一個JDK提供的Package）下的所有類別
    - 例如java.lang.Object與java.lang.String。

16

2.4 import區

- Java的所有類別全部都繼承自 **java.lang.Object**
  - 換句話說，只要程式中未指定父類別的類別，Java都會自動指定Object作為其父類別。
  - 例如在範例2-1中第16行的MyClass就是以Object做為父類別
  - 以『java.lang.*』做為引入類別庫的宣告。如此就會引入java.lang.Object

17

2.4 import區

- Java會**自動引入java.lang類別庫**
  - java.lang類別庫存放著大量常使用的類別（例如字串類別、數學類別、Object類別等）
  - 故而省略範例2-1的「import java.lang.*;」，並不會造成程式的錯誤。
  - 但對於Java不會自動引入的類別庫，則必須明確宣告其import來源。

18

2.5 類別區

- Java是一個物件導向語言，以類別做為單位，所以每一個Java程式必須有一個主類別才能夠被執行。
  - 除了主類別外，我們也可以依照需要自行定義一般類別。

19

2.5.1 主類別區

- 主類別是程式中不可或缺的類別，並具有下列特點：
  - 1. 主類別具有唯一性，必須且只能有一個主類別。
  - 2. 必須包含一個主函式main()，稱之為main() method
    - 並且main()是整個程式的執行起始點。
  - 3. 封裝等級必須宣告為public，且只有主類別可以宣告為public。
    - （預設為public，所以public可省略）
  - 4. 主類別名稱必須與主檔名相同（不含副檔名）。
  - 5. 修飾字的宣告只能宣告為abstract、final或省略不寫。

20

2.5.1 主類別區

- 基於上述特點，我們可以將主類別的格式整理如下：

圖2-2 主類別格式（格式中，[ ]代表可以省略不寫、而 | 代表選其一）

21

2.5.1 主類別區

- 主類別區說明：
  - 1. public是封裝等級
    - 我們會在繼承與封裝一章中說明
    - 目前只需要了解主類別一定要宣告為public即可。
  - 2. abstract|final是修飾字
    - 同樣我們會在後面慢慢說明。
    - 目前我們都先暫時省略（不寫）修飾字。
  - 3. 主類別名稱必須與主檔名相同
    - 例如範例2-1的檔名為ch2_01.java，所以主類別名稱為ch2_01。

22

2.5.1 主類別區

- 4.[extends 父類別]是用來宣告繼承自哪一個父類別
  - 例如要撰寫Applet程式，就要宣告為extends Applet。
  - 目前我們都先不寫。
- 5.[implements 介面名稱] 是用來宣告實作哪一個介面
  - 例如要撰寫多執行緒程式，就要宣告為implements Runnable。
  - 目前我們都先不寫。
- 6.宣告變數(variable)是在類別中宣告屬於該類別的變數（也就是欄位）。
  - 格式說明請見2.5.3節。

23

2.5.1 主類別區

- 7.main method
  - 存取等級必須是public
  - 並且使用static宣告為class method
  - void代表沒有回傳值。
  - 前面的public static void是固定不可以改變的。
  - 至於()內String[] args、String []args、String args[]則為main的參數，也就是用來接受命令列執行時後面的引數，它是以字串格式傳入
    - （三種都是傳入字串陣列的格式，擇其一即可，args為陣列名稱，可取其他名稱代替）。
- 8.除了main method之外，我們也可以宣告其他屬於主類別的method。
  - 格式說明請見2.5.4節。

24

2.5.1 主類別區

延伸學習：**術語的不同**

不同的物件導向程式語言採用不同的術語，例如在上一章所介紹的屬性，其意義代表屬於該類別的變數。但在C++中，C++將類別所屬的變數稱之為成員變數(member variable)，將類別所屬的函式稱為成員函式(member function)。而Java則將類別所屬的變數稱之為**變數(variable)**或**欄位(field)**，將類別所屬的函式稱為**方法(method)**。

25

2.5.1 主類別區

延伸學習：**Java的變數**

Java的變數一共可分為四種，實體變數(Instance Variables)、類別變數(Class Variables)、區域變數(Local Variables)、參數(Parameters)，其意義如下，我們將於後面章節中陸續介紹。

- **參數**：又稱為參數變數，只使用在method呼叫時接收傳遞資料之用。（詳見第6章說明）
- **區域變數**：僅限用於某個method之內的變數。
- **實體變數**：隸屬於某個物件的變數。必須透過物件才能存取該變數。
- **類別變數**：隸屬於某個類別的變數，所有由該類別產生的物件都共用此變數，並且不需要產生物件，就可以透過類別存取該變數。類別變數必須使用static關鍵字宣告。

在圖2-2中，有一塊「宣告變數」區，該區是宣告實體變數與類別變數，兩者的差別在於是否使用static關鍵字宣告。

26

2.5.2 一般類別區

- 一般類別的規定比主類別少，格式可以整理如下：
  - 【說明】：在第6章之前，我們都不會使用到一般類別，所以我們將於第7章再一併說明。

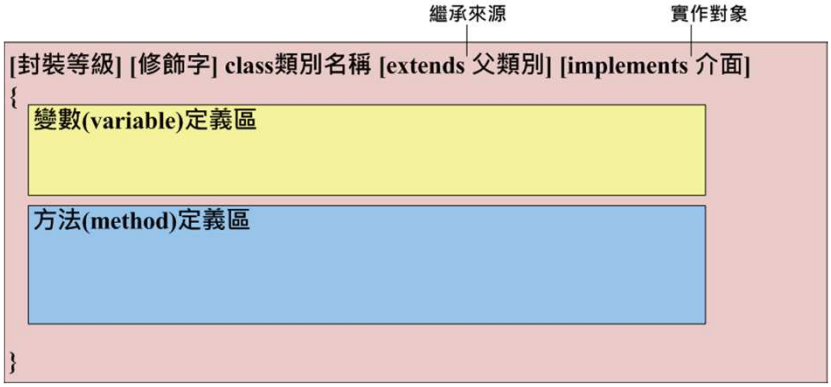


圖2-3　一般類別格式（格式中，[ ]代表可以省略不寫）

27

2.5.3 類別的變數

- 在類別的變數定義區中定義的變數
  - 稱為該類別的成員變數(member variable)，或欄位(fields)
  - 有時為了方便說明，也會以成員資料來解釋。
- 成員變數
  - 該類別專屬的資料欄位，若類別生成物件後，將會是物件的資料項目。
  - 在類別宣告單一變數的語法如下：

```
[封裝等級]  [修飾字]  資料型態  變數名稱;
```

28

2.5.3 類別的變數

- 【說明】：
  - 定義成員變數可在前宣告封裝等級（我們將在類別與物件一章中說明各種封裝等級）。
  - 若未加上static修飾字，則該變數將存在於該類別生成的每一個物件中，且彼此獨立，稱為**實體變數(Instance variable)**。
    - 若使用static宣告，則每一個物件都共用該變數，稱為**類別變數(class variable)**。
  - 資料型態可以分為原始資料型態與非原始資料型態，我們將於下一章說明。
  - 變數名稱可自由選擇由多個Unicode字元構成的識別字(identifier)
    - 但不能是Java的關鍵字、true、false及null等保留字
    - Java的關鍵字及保留字如下：

29

2.5.3 類別的變數

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

老師的叮嚀

上述的Java關鍵字(Keyword)都具有特殊意義，我們將於本書的各章節中陸續加以介紹。至於goto等保留字(reserved word)雖然在Java中沒有任何意義，但由於在一般程式語言中具有特殊意義，因此，Java將之列為保留字，同樣不可以做為變數名稱、方法名稱或類別名稱。

30

2.5.3 類別的變數

Coding 注意事項

識別字其實包含兩大類，一類是由使用者所定義，另一類則是由編譯器內定。
由編譯器內定的識別字為關鍵字或保留字。
而使用者定義的識別字，則是做為變數名稱、方法名稱、類別名稱、常數等等。

雖然這些識別字可以由任意的Unicode字元構成，但不可以在其中包含空白字元及#、$等特殊符號，同時數字只能在第二個字元後出現。並且Java的識別字**具有大小寫之分**，例如Length與length是不同的兩個識別字。

31

2.5.3 類別的變數

Coding 偷撇步

雖然使用者可自行定義識別字，以做為變數、方法、類別等名稱，但最好取一些有意義的單字或組合字，以便於日後的維護。而在Java的內建類別中，我們可以發現一些「識別字的命名規則」，您可以跟隨相同的命名規則來進行識別字的命名：

常數：全部字元都使用大寫，若包含兩個單字以上，則可以在其中加上底線，例如：PI,MAX_LEN。

變數：全部字元都使用小寫，但若為兩個單字以上，則第二個單字開頭字母為大寫，例如：width,rectWidth,varRectWidth。

方法：（同變數名稱命名規則）全部字元都使用小寫，但若為兩個單字以上，則第二個單字開頭字母為大寫，例如：show,cal,calArea,calRectArea。

類別：第一個字母為大寫（通常可命名為C，代表class），其後由一個單字以上組成，每個單字的開頭字母為大寫，其餘為小寫，例如：CPerson,CStack,CCircularQueue。

32

2.5.4 方法

- 在類別的成員定義區中定義的方法(method)
  - 也稱為該類別的成員函式(member function)。
  - method的定義格式如下：

```
回傳值型態 [修飾字] method名稱(資料型態 參數1,資料型態 參數2, ...)
{
    ... 程式敘述群 ...（含區域變數的宣告）
    [return 運算式;]
}
```

33

2.5.4 方法

- 類別的方法說明：
  - 1.程式透過邏輯來完成其目的，而方法就是Java程式用以實作程式邏輯之處。
  - 2.程式敘述群中包含許多的Java合法敘述，其中也包含區域變數的宣告。
  - 3.return是用來中斷method的執行，並且可以回傳資料，如果不想回傳資料，則回傳值型態應該宣告為void。
  - 4.宣告方法時，若未加上static修飾字，則該方法必須由該類別生成的物件來執行，稱為**實體方法(Instance method)**。
    - 若使用static宣告，則該方法可以直接透過類別指定的方式來執行，稱為類別方法(class method)。

34

2.5.4 方法

- 5. 參數是做為接收傳遞變數之用，它可以做為該method內的合法變數來使用，而不必再宣告一次。
- 6. 程式的運作是依靠變數的變化來完成。
  - 變數對於程式設計師而言，是一個運用對象，而對於電腦而言，則會佔據某一小塊記憶體以保存資料。
  - 在method內宣告的變數稱為區域變數，它的宣告語法與類別變數宣告類似，不過不包含封裝等級。
- 7. 宣告區域變數與其他敘述可以交錯進行，但只有宣告過的區域變數才能夠被使用。

35

2.5.4 方法

Coding 偷撇步

因為方法代表的是物件導向裡的「操作」，因此方法的命名通常以動詞為主，有時也會加上一個名詞在後方（就像是英文中的及物動詞+受詞那樣），甚至會加上一些如by,with,after等等的介係詞讓方法名稱可以更一目了然地理解該方法在做些什麼。有時也會參考回傳值型態及方法被呼叫時的樣子來命名方法名稱。

事實上，當您學會程式設計後，若想要在職場上寫出容易維護的程式，還需要注意更多關於變數、方法、類別的命名原則，進一步的資訊可參考筆者審校的「我的程式碼會說話」一書，但這需要您具備更多程式經驗後，才能體會其中的意義。

36

2.6 Java程式的進入點main(…)

- 一個完整的Java程式必定有一個唯一的主類別（主類別名稱同檔名），而主類別中必定有一個唯一的main方法
  - main方法將是整個程式的進入點（即起始點）。
- main方法的格式如下：

```
public static void main(String[] args|String []args|String args[])
{
  … 程式敘述群 …
  [return;]
}
```

37

2.6 Java程式的進入點main(…)

- 說明：
  - (1)main 方法的存取等級必須是public，並使用static宣告為class method，其中的void代表沒有回傳值。所以前面的public static void是固定不可以改變的。
  - (2)至於()內的參數可以為String[] args或String [] args或String args[]，它可以接受命令列執行時後面的引數
    - （三種都是傳入字串陣列的格式，擇其一即可，args為陣列名稱，可取其他名稱代替）。
    - 我們將於函式一章中再詳述，目前暫時不對參數進行任何應用。
  - (3)程式敘述群包含宣告區域變數與其他敘述，並可交替交錯進行，但只有宣告過的區域變數才能夠被使用。
  - (4)return可以提前中斷main方法的執行，當main方法執行完畢（執行到最後一行或遇到return），則整個Java程式就執行完畢。

38

2.7 Java的敘述(statement)

- Java的程式由類別組成
  - 而類別內則包含了定義變數及方法
    - 在各方法內則包含宣告區域變數以及其他各類敘述。
- 換句話說，程式是由許多的敘述所組成
  - 您也可以把敘述當作是指令來看待，即每一個敘述代表要求JVM進行某項運作。

39

2.7 Java的敘述(statement)

- Java的敘述分為許多種類，而這些敘述又可以依照數量分為單一敘述與區塊(block)敘述
  - 區塊敘述又稱為區段(segment)，它包含了一個以上的敘述，因此必須使用一對大括號{}包裝起來
  - {}在Java的程式中時常可以看到，其目的都是為了將多個敘述包裝起來成為一個區段
    - 例如main方法底下的{}之間的所有敘述，就是代表main方法所要執行的指令，我們也可以稱之為main方法的主體。

40

2.7 Java的敘述(statement)

- 其他如類別之下，也會出現{}，其中所包含的則稱為該類別的主體。

```
package myJava.ch02;
import java.lang.*;

public class ch2_01              //主類別
{
    public static void main(String args[])
    {
        System.out.println("歡迎使用Java!");
    }
}

class MyClass                    //一般類別
{
}
```

主類別的主體

main()方法的主體

MyClass類別的主體

圖2-4 類別與方法主體

- 在往後的章節中，在介紹某些敘述時（如if敘述），也會看到{}的出現，其主要目的都是為了將許多單一敘述集合起來，當作一個單位來看待。

41

2.8 println()輸出方法的簡易使用法

- 在範例2-1中，main方法的主體內只包含了單一個敘述
  - System.out.println("歡迎使用Java!");
- System類別的全名是java.lang.System
  - 故我們在第6行import java.lang.*
    - 事實上，由於java.lang太常被使用，所以即使未將之import，編譯器仍然會內定自動將之import進來。

42

2.8 println()輸出方法的簡易使用法

- System.out.println("歡迎使用Java!");
  - out是java.lang.System的一個靜態欄位（類別變數）
    - out型別為PrintStream類別
    - println是PrintStream類別的一個方法
  - System.out代表的是標準輸出，通常指的是螢幕或印表機等（預設為螢幕）。
    - println之意為print與line，代表印出文字並且換行。
    - 此外，還有一個方法稱為print，其功能則是單純印出文字而不換行。

43

2.8 println()輸出方法的簡易使用法

- 在範例2-1中，我們使用System.out.println印出了一段文字『歡迎使用Java!』並換行
  - 其中，我們將該段文字以『"』包裝起來，代表一個字串常數
  - System.out.println還能夠印出各類常數與變數的值。
  - 我們將於下一章中看到更多關於System.out.println與System.out.print的使用範例。
    - 請注意，在這兩個方法中，如果要列印的資料不只一筆，則可以利用『+』來將之連結，並且『+』在Java中，也可以作為連結字串之用。

44

2.9 自由格式與空白字元

- Java語言採用自由格式撰寫，換句話說，您可以去除程式中各敘述間的所有空白字元(包含spaces、tabs…等等)及換行符號（carriage、return），編譯器仍會正確編譯程式
  - 例如：您可以將範例2-1中主類別的內容改寫如下：

```
public class ch2_01          //主類別
{ public static void main(String args[])
   {   System.out.println("歡迎使用Java!");   }
}
```

45

2.9 自由格式與空白字元

- 省略空白字元以及換行符號會使程式碼更難閱讀。
- 為了日後維護的方便，強烈建議讀者應該培養程式碼縮排及適當換行的習慣。

Coding 注意事項

『"』內的空白字元不會被編譯器忽略，因為『"』在Java語言中，是用來包裝字串常數。

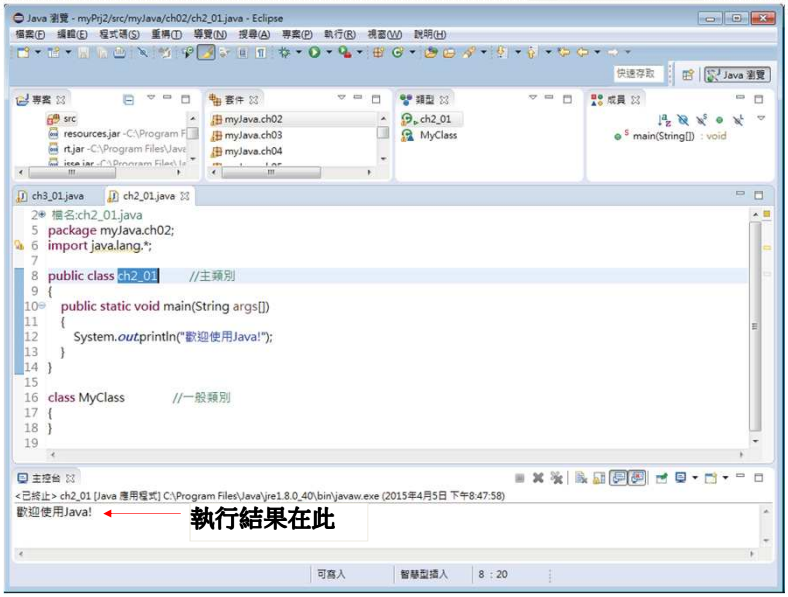
46

2.10 Eclipse IDE的活用

- 在前面章節中，我們採用的是編輯器（如記事本）+javac.exe+java.exe(JVM)來編輯、編譯與執行Java Application
  - 本書建議，初學者盡量以此模式來學習本書，比較能夠理解Java的架構。
  - 但在學會Java之後，為了方便開發更大型的專案，不可避免地，應該搭配IDE的協助以縮短開發時程並利於管理
  - 因此，建議讀者參照附錄B的指示，下載、安裝、設定與執行Eclipse IDE
    - 例如在Eclipse中執行範例2-1的結果如下圖：

47

2.10 Eclipse IDE的活用



48

本章結束

Q&A討論時間