程式語言 (Programming Language)

Lec. 02 程式設計工具與架構

楊　吳　泉

義守大學資訊工程學系

web: http://elearning.isu.edu.tw

http://audtm.net/~wcyang

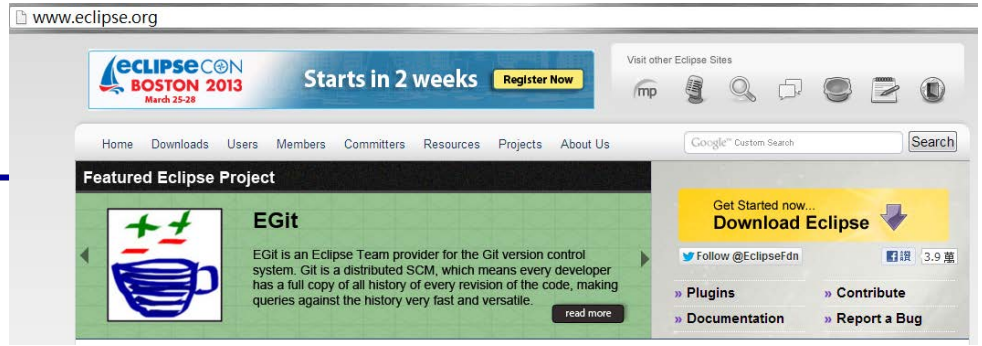
mail: wcyang@isu.edu.tw

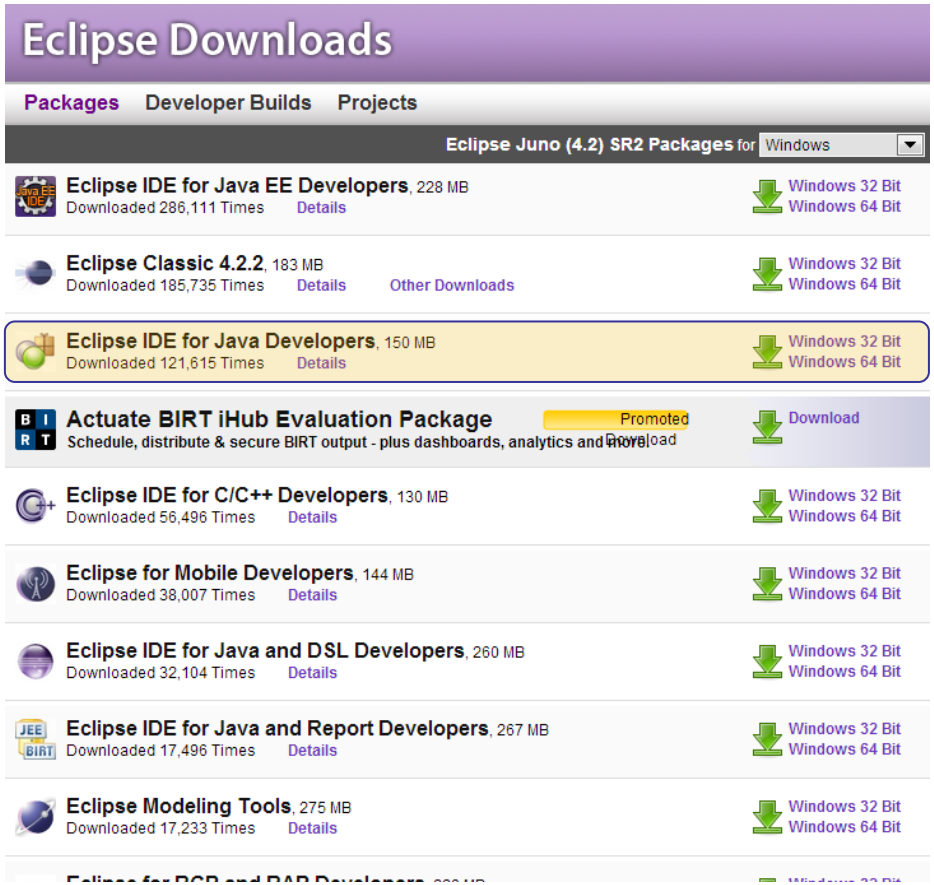
綱要 (Outline)

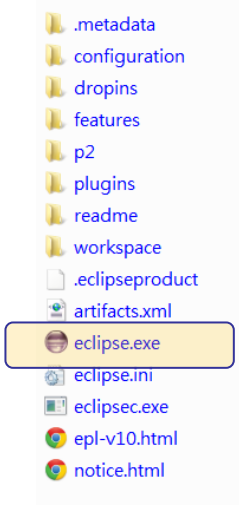
- Eclipse安裝與使用
- Java程式語法說明
  - ✓ Comments
  - ✓ Identifiers & Keywords
  - ✓ Data types
  - ✓ Variables
  - ✓ Expressions
  - ✓ Arrays
  - ✓ Flow Control
- 程式練習

Eclipse安裝與使用

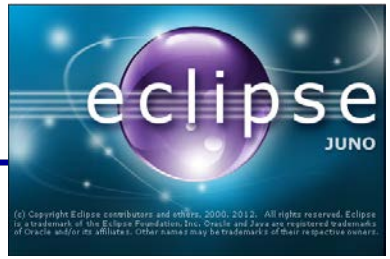
- 下載
  - ✓ 官網: http://www.eclipse.org
  - ✓ Download
    - Eclipse IDE for Java Developer
  - ✓ 解壓縮(Windows:ZIP檔案)後直接執行eclipse.exe



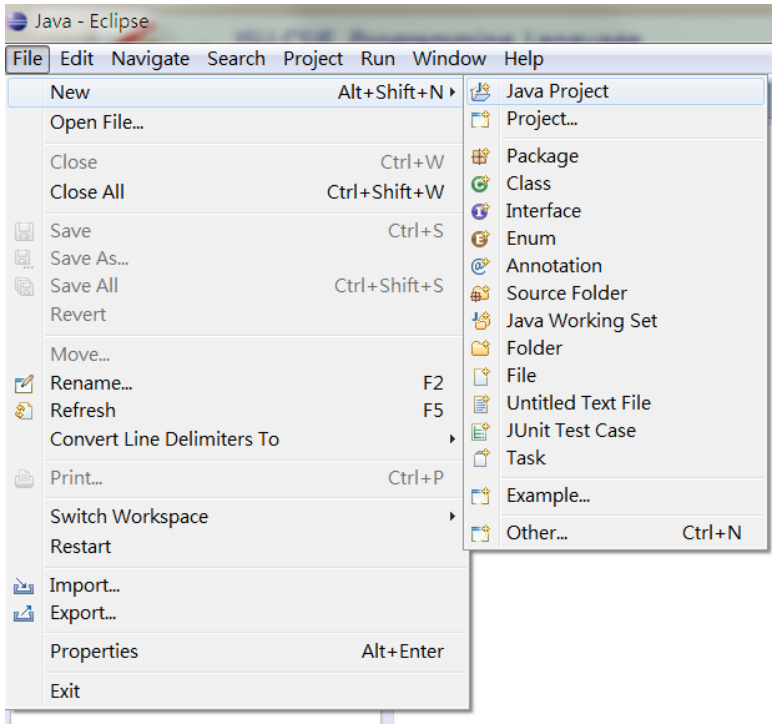


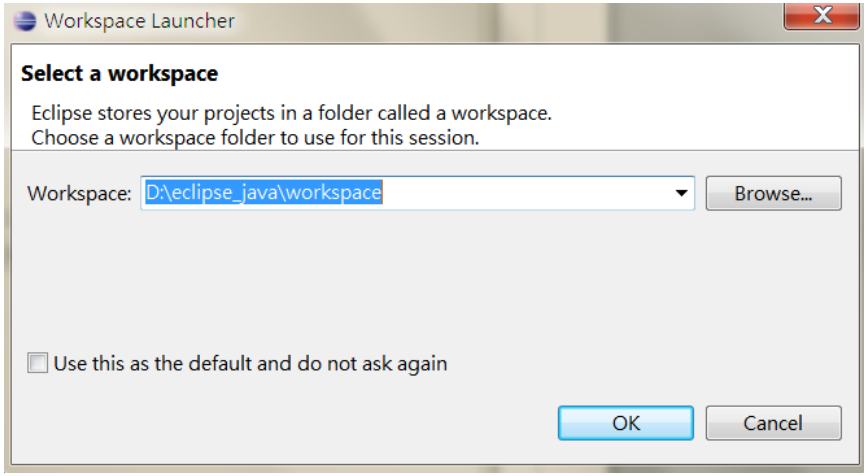


Eclipse執行 – 1



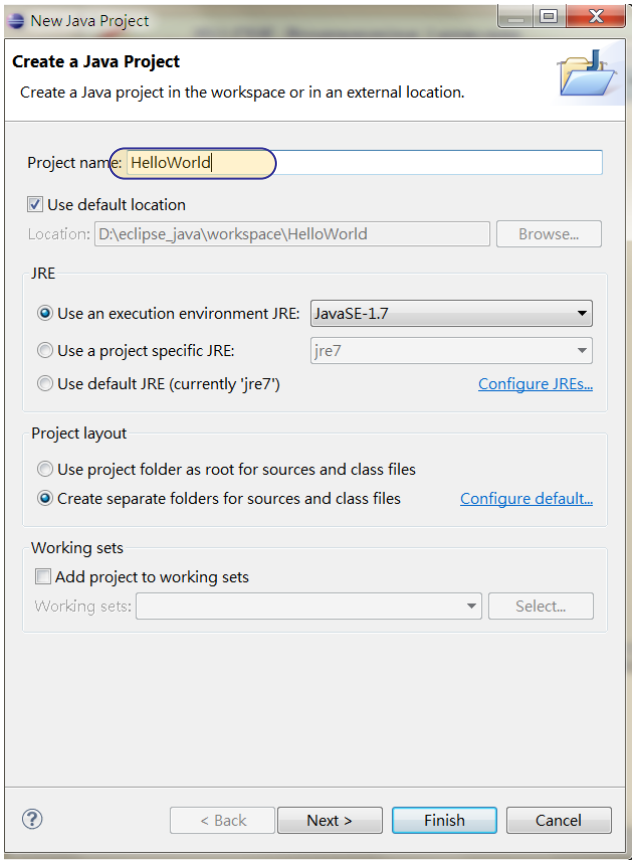
- 設定程式工作區
  - ✓ 設定程式執行目錄
- 開啟專案
  - ✓ New Java Project

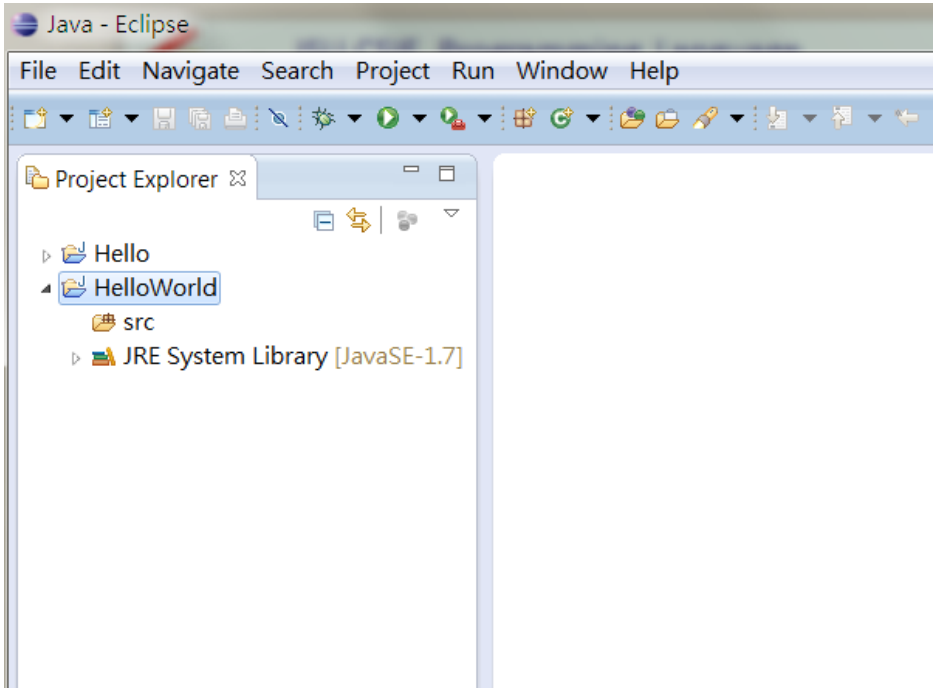




Eclipse執行 – 2

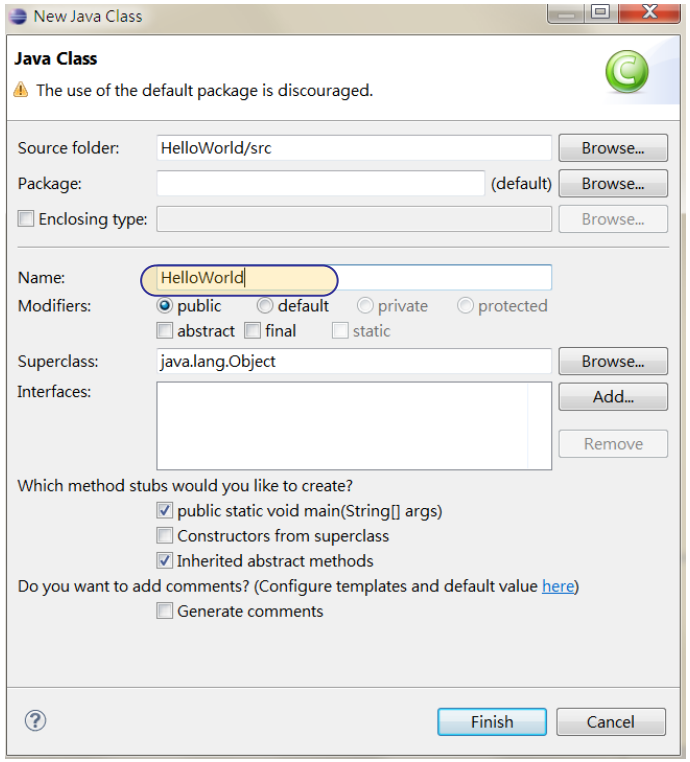
- Create a Java Project
  - ✓ 設定Project名稱

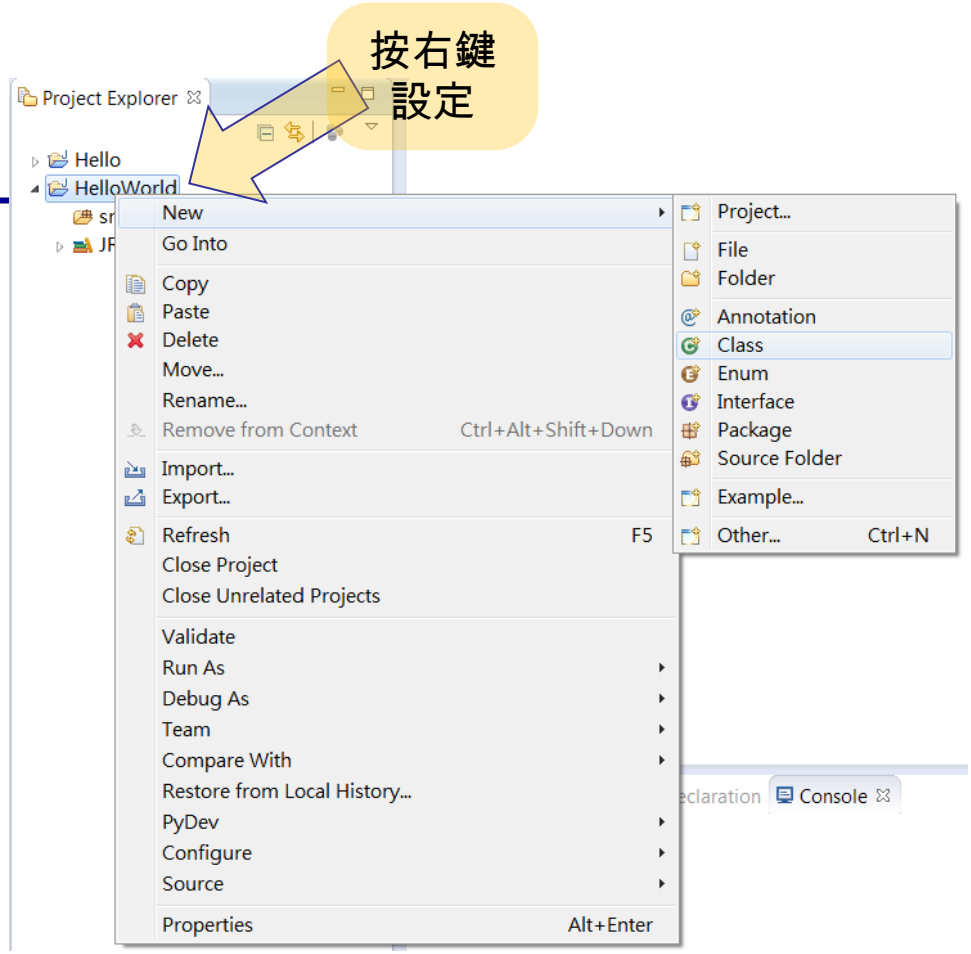




Eclipse執行 – 3

- 建立Java Class
  - ✓ New Class





Eclipse執行 – 4

- 程式編輯

Java - HelloWorld/src/HelloWorld.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- Hello
- HelloWorld
  - src
    - (default package)
      - HelloWorld.java
  - JRE System Library [JavaSE-1.7]

HelloWorld.java

```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.printl
    }

}
```

- println() : void - PrintStream
- println(boolean x) : void - PrintStream
- println(char x) : void - PrintStream
- println(char[] x) : void - PrintStream
- println(double x) : void - PrintStream
- println(float x) : void - PrintStream
- println(int x) : void - PrintStream
- println(long x) : void - PrintStream
- println(Object x) : void - PrintStream
- println(String x) : void - PrintStream

Press 'Alt+/' to show Template Proposals

- *println*

```
public void println()
```

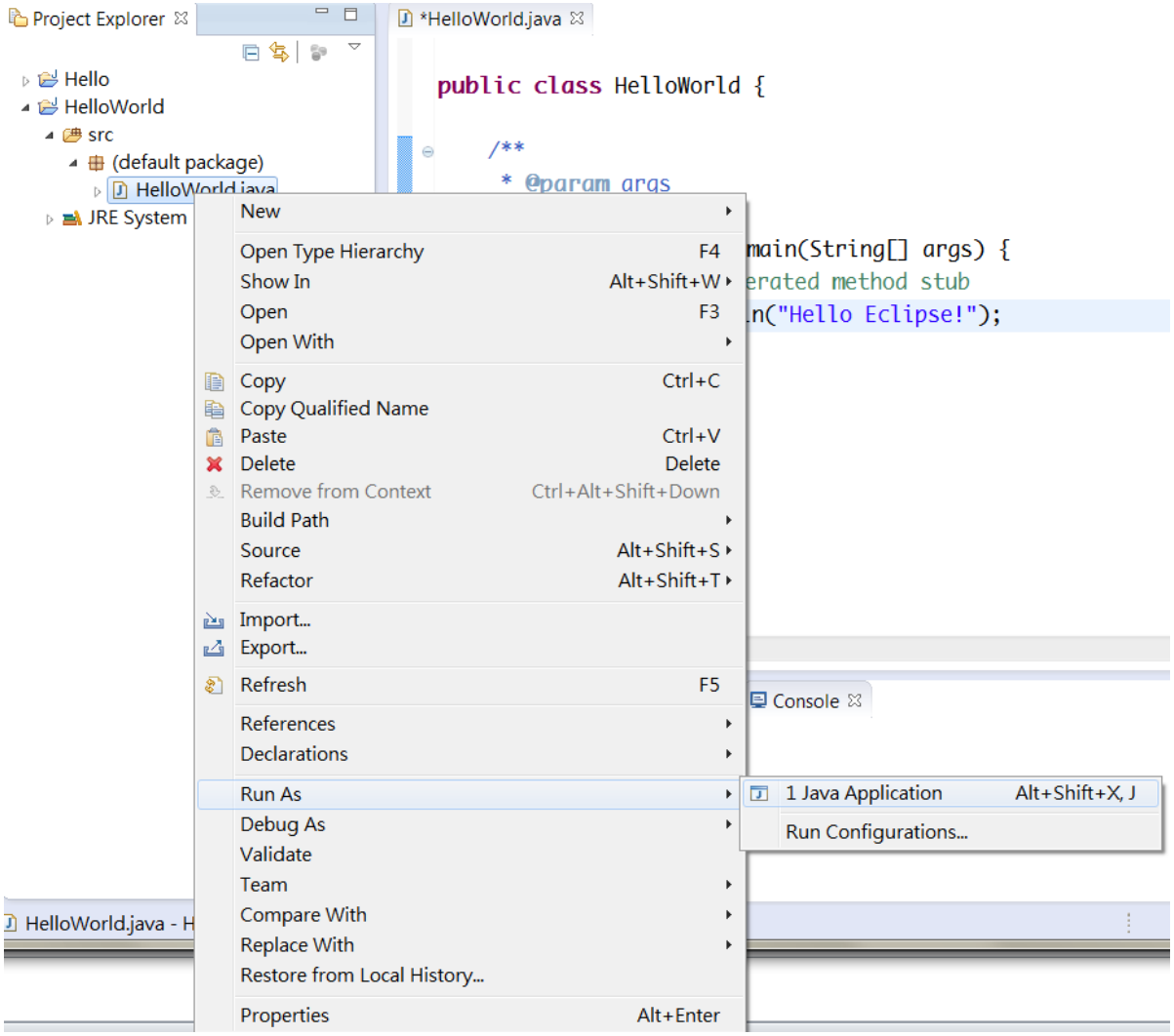
Terminates the current line by writing the line separator string. The line separator string is defined by the system property `line.separator`, and is not necessarily a single newline character (`'\n'`).

Press 'Tab' from proposal table or click for focus

Javadoc Decl

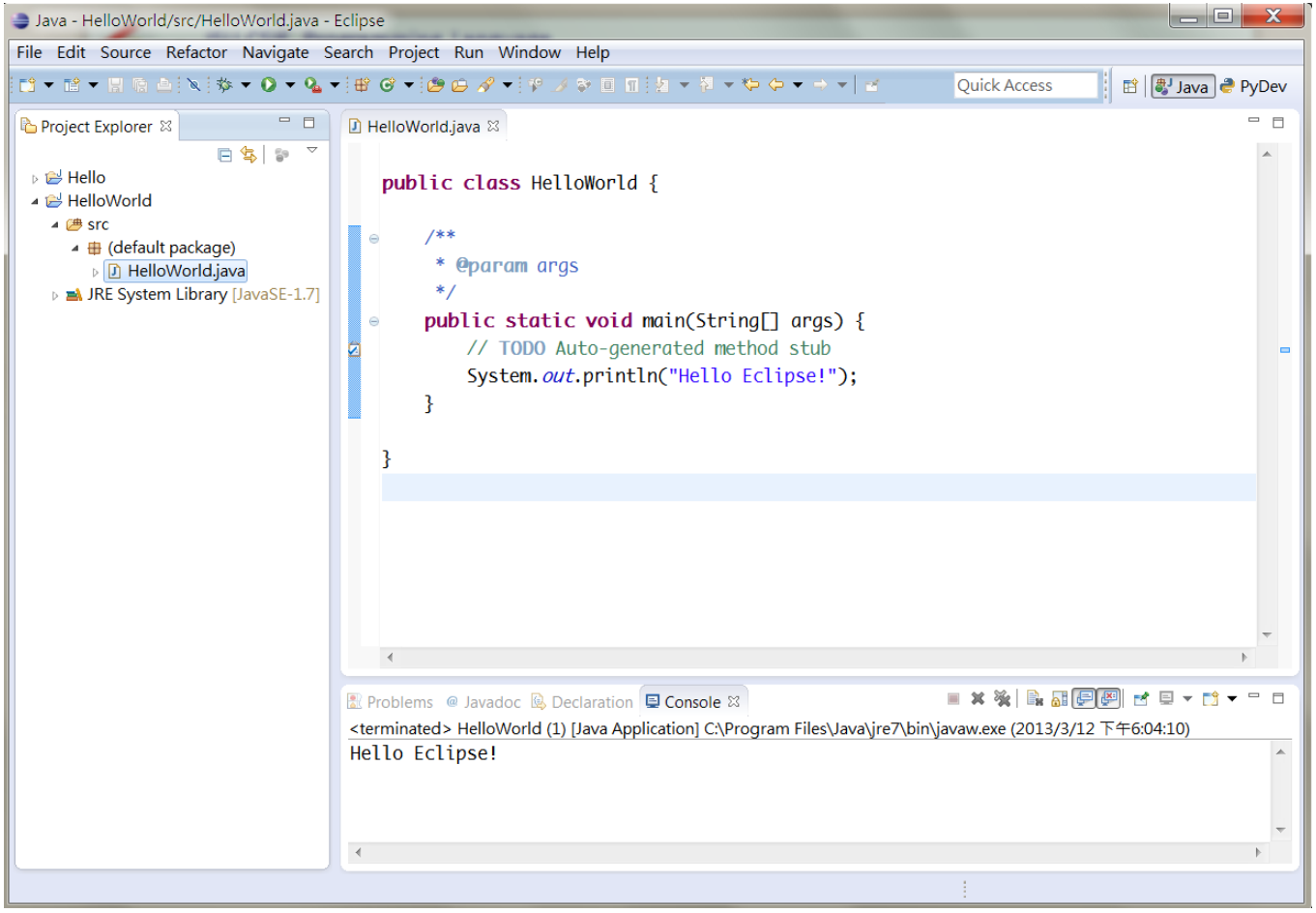
Eclipse執行 – 5

- 程式執行 - 1



Eclipse執行 – 6

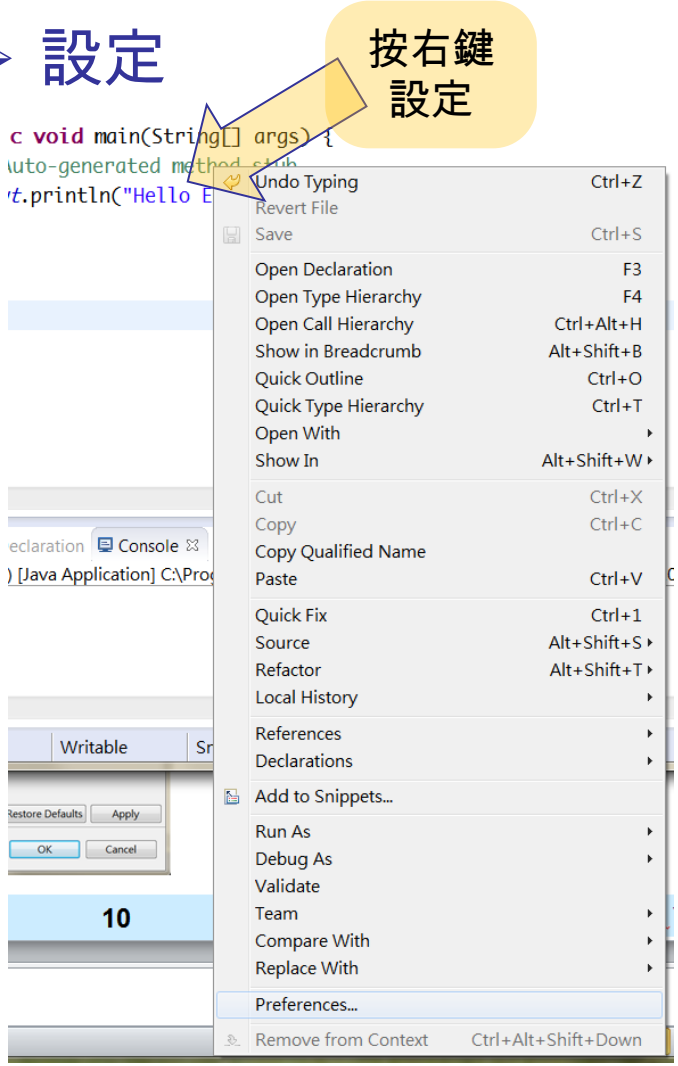
- 程式執行 - 2

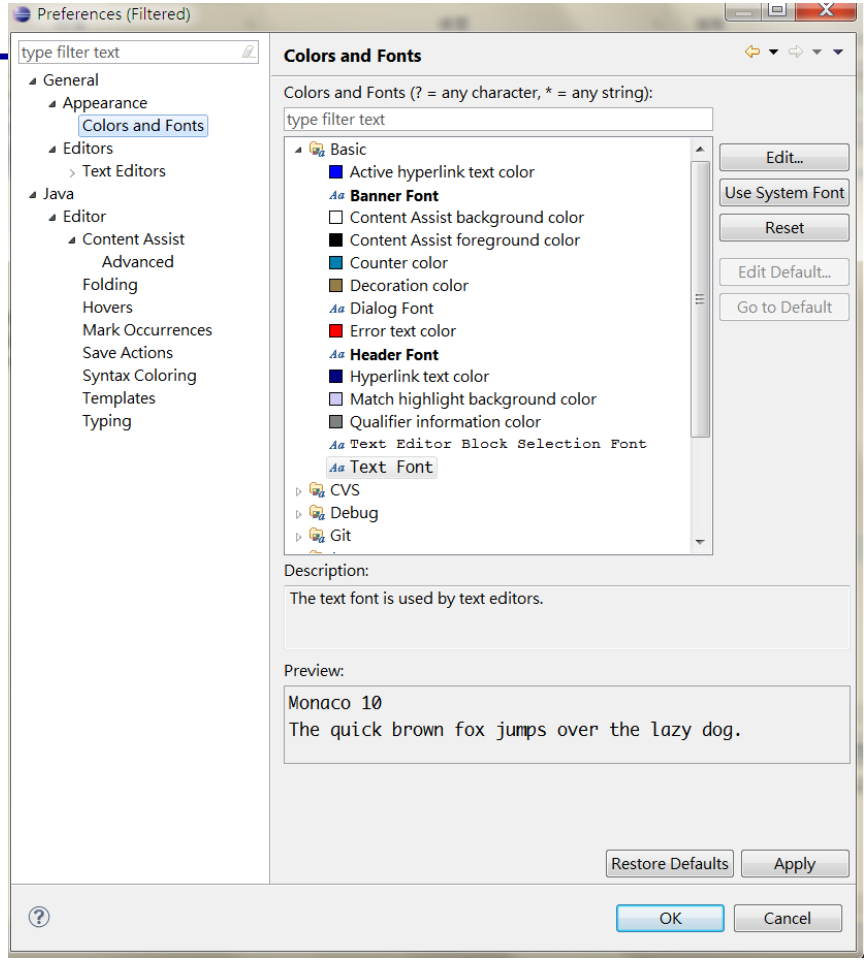


Eclipse執行 – 7

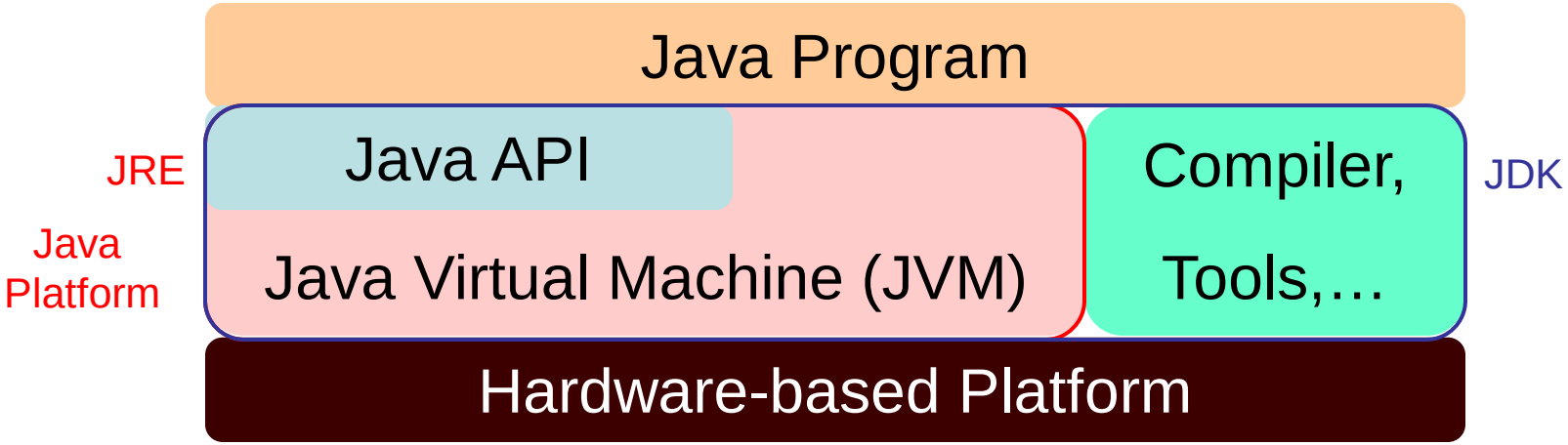
- 設定

按右鍵
設定





Java程式概念 – 平台架構



Java	Java	Java
Sample codes	Documentations	Source codes

Java程式概念 – 程式架構

- Java程式可分成
  - ✓ Applications: 獨立應用程式
  - ✓ Applets: 小程式, 附屬於網頁執行之程式
  - ✓ 其他: Servlets, Java Beans, …

```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello Eclipse!");
    }

}
```

- 基本Java外觀

[package定義;]
[import 相關類別;]

```
[修飾字] class 類別名稱
   [extends 父類別]
   [implements 相關介面s]
   [throws 相關例外s] {

   變數與方法定義;

   [ //如果需要主程式(main method)
     public static void main(String args[]) {
       變數與敘述 ;
     }
   ]
}
```

Java 程式語法(Program Syntax)

- 註解 (Comments)
- 識別字(Identifiers) & 關鍵字(Keywords)
- 資料型態 (Data types)
- 變數 (Variables)
- 運算式 (Expressions)
- 陣列 (Arrays)
- 流程控制 (Flow Control)
- 範例
  - ✓ 計算sum = 1+3+5+…+199 (假設需先將1,3,5,…存入陣列)

Comments

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

- Java支援三種註解之方式

  (1) 單行註解, 以 // 為開頭延續至該行結束

  ex: int I=0; //設定迴圈變數

  (2) 多行註解, 以 /* 為開頭, 延續任意多行, 直到 */ 為止.

  ```
  /* This is my first program.
     Date: 2000.09.05
     Version: 1.3
  */
  ```

  (3) 說明文件註解,以 /** 為開頭, 直到 */ 為止.

  (利於javadoc處理成HTML文件)

Identifiers & Keywords (1/2)

- 識別字用以指Java程式中元件所用的符號名稱.

  包括:類別名稱, 方法名稱, 參數名稱, 變數名稱

  以字母, 底線(_), Unicode之貨幣符號(如:$,£,¥)為開頭

  之後接任意數目之字母, 數字, 底線, 貨幣符號

  (JAVA 採用Unicode字元集)

  (識別字變數命名以方便記憶與辨識為原則)

  Ex: ProcessUnit, DigitalSignature, Applet

  Ex: theKey, result

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

Identifiers & Keywords (2/2)

- 保留字意指Java語言所使用之關鍵字, 實字或修飾詞

  (實字(literal)意指定義某些意義之保留字)

  Ex: while, for, if, super

  Ex: int, long, short, byte, char

  Ex: null, true, false

  Ex: public, abstract, protected, private

PS: 識別字之命名不得與保留字相同

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

Data Types (1/2)

- 基本資料型態

(1) 整數: byte, short, int, long

→8, 16, 32, 64-bit 2'complement integer

(2) 浮點數: float, double

→32, 64-bit floating point number

(3) 布林: boolean

→{true, false}

(4) 字元:char

→16-bit unicode

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

Data Types (2/2)

Ex: int i,j,k;

Ex: float x,y,z;

Ex: float s=1.0;

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

說明:

(1) Java為Strong type之程式語言.

(2) Java API中package java.lang中有一些資料類別,
如Integer, Float, String, ...提供許多方便之方法可用.

(3) 類別與陣列是屬於參考型態之資料行, 另行介紹.

Variables (1/3)

- 變數宣告

變數使用前需先宣告

ex: int i,j;

float a,b;

char c,d;

public final pi=3.1216;

設定兩邊變數之型別需相同

ex: c='A';

i=j+2;

a=b+i;

j=(int) b;

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

Variables (2/3)

- 變數範圍

(1) 變數之存在範圍:

若在method內宣告, 變數使用範圍限於該method之內.

若在method外宣告, 變數使用範圍限於該class之內.

(2) 若是子類別宣告同名稱之變數(重複定義),

新宣告會蓋掉上一層之宣告.

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

Variables (3/3)

- 變數特性修飾字(*)
  - (1) static: 變數為附屬該類別之所有物件所共用
  - (2) final: 變數不可再修改, 一般使用於常數
  - (3) 無修飾: 一般之變數 (default)

- 存取權限修飾字(*)
  - a. 相同之類別可存取.
  - b. 繼承該類別之子類別可存取.
  - c. 相同之package之類別可存取.
  - d. 一般程式可存取.
  - (1) public: a,b,c,d.
  - (2) protected: a,b,c. (default)
  - (3) private: a.

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

Expressions

- 基本運算子
  - ✓ 算術算子: + - * / %
  - ✓ 邏輯算子: || && !
  - ✓ 位元算子: | & ^ ~ << >> >>>(unsigned)
  - ✓ 設定算子: = (+,-,*,/,%,|,&,^,<<,>>,>>>)=
  - ✓ 關係算子: == != < > <= >=
  - ✓ 遞增遞減: ++ --
  - ✓ 條件算子: ?:
  - ✓ 其他: . [] () ; , instanceof

➢ 撰寫程式時, 需注意運算子之優先順序.

- ✓ ()括號可以改變優先順序

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

Arrays

- 陣列---相同型態資料之集合

Ex: int m1[]=new int[200];

Ex: int[] m1=new int[200];

Ex: String names[]={”John”,”Tom”,”Mary”};

Ex: String names[]=new String[3];

names[0]=”John” ;

names[1]=”Tom” ;

names[2]=”Mary” ;

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

Flow Control

```java
//我的第2個練習程式
public class Sum {

    public static void main(String[] args) {
        int sum=0;
        int[] a = new int[100];
        for(int i=0; i<100; i++) a[i]=i+i+1;
        for(int i=0; i<100; i++) sum += a[i];
        System.out.println("sum = "+sum);
    }

}
```

- 流程控制指令
  - ✓ if
  - ✓ if … else
  - ✓ for
  - ✓ while
  - ✓ do … while
  - ✓ switch
  - ✓ break, continue, return, catch, throw