National University of Kaohsiung

Java Programming
Introduction

Based on slides from the authors
Revised by Ya-Ju Yu

What Is Programming?

- Programming is the activity of writing a sequence of instructions to tell a machine to perform a specific task.
  - A sequence of instructions
    - ✓program
  - A set of well-defined notations is used to write a program
    - ✓programming language
  - The person who writes a program
    - ✓designer


PROGRAMMER
WHAT MY MOM THINKS I DO
WHAT MY FRIENDS THINK I DO
WHAT SOCIETY THINKS I DO
[JavaScript Application]
Hello, world!
OK
WHAT ARTISTS THINK I DO
WHAT I THINK I DO
WHY WON'T YOU COMPILE?
WHAT I ACTUALLY DO

In Practice

- Programming is to provide a solution to a real-world problem using computational models supported by the programming language.
  - The computational solution is a program.
  - The program must be effective.

How and Where The Programs Run

- The programs are activated from the disk into the main memory.
- Now we call them the processes.
- CPUs contain the arithmetic and logic unit (ALU) and the registers.
  - ALU is responsible for the computational power.
  - Registers store the data to be used temporarily.
- The outputs are written back to the main memory and further stored into the disk if necessary.

Programming Languages

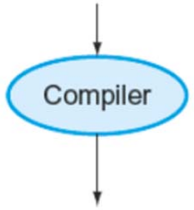
- A programming language is an artificial language to communicate with machines.
- Recall how you learned the 2nd nature language when you were a kid.
- Programming languages->syntax and semantics
  - Used to express algorithms
  - Used to control the behavior of machines
- How many programming languages in the world?
  - More than 1000.
  - Java: top 3

History

- 1st generation: machine code
- 2nd generation: assembly code
- 3rd generation: high-level programming languages
- Java is one of the 3rd-generation programming languages.

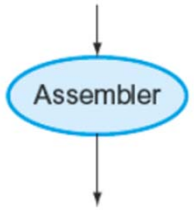
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```



Assembly language program (for MIPS)

```
swap:
      multi $2, $5,4
      add   $2, $4,$2
      lw    $15, 0($2)
      lw    $16, 4($2)
      sw    $16, 0($2)
      sw    $15, 4($2)
      jr    $31
```



Binary machine language program (for MIPS)

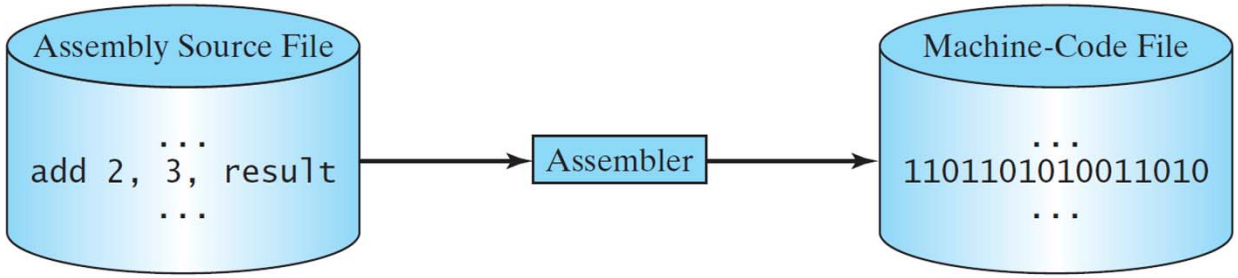
```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001111000010010000000000000100
10101111000010010000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

1st-Generation Programming Languages

- Computers understand instructions only in binary, which is a sequence of 0s and 1s. (Why?)
- Each computer has its own set of instructions.
- So the programs at the very early stage were machine-dependent.
- These are so-called the machine language, aka (as known as) machine code.
- Pros:
  - Most efficient for machines
- Cons:
  - Hard to program for human
  - Not portable
- Still widely used in programming lower level functions of the system, such as drivers, interfaces with firmware and hardware.

2nd-Generation Programming Languages

- An assembly language uses mnemonics to represent instructions as opposed to the machine codes.
- Hence, the code can be read and written by human programmers.
- Yet, it is still machine-dependent.



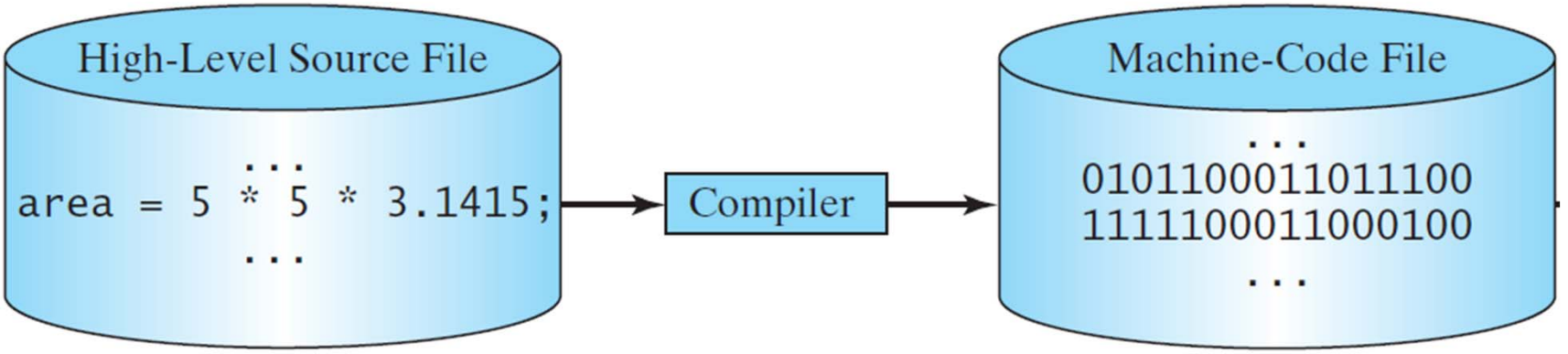
- To run on a computer, it must be converted into a machine readable form, a process called assembly.

3rd-Generation Programming Languages

- The high-level programming languages use English-like words mathematical notation, and punctuation to write programs.
- They are closer to the human languages.
- Pros:
  - Portable, machine-independent
  - Human-friendly
- For example, C, C++, and Java

3rd-Generation Programming Languages

- Note that the machines understand and execute only the machine codes as before.
- The translation is accomplished by a compiler, an interpreter, or a combination of both

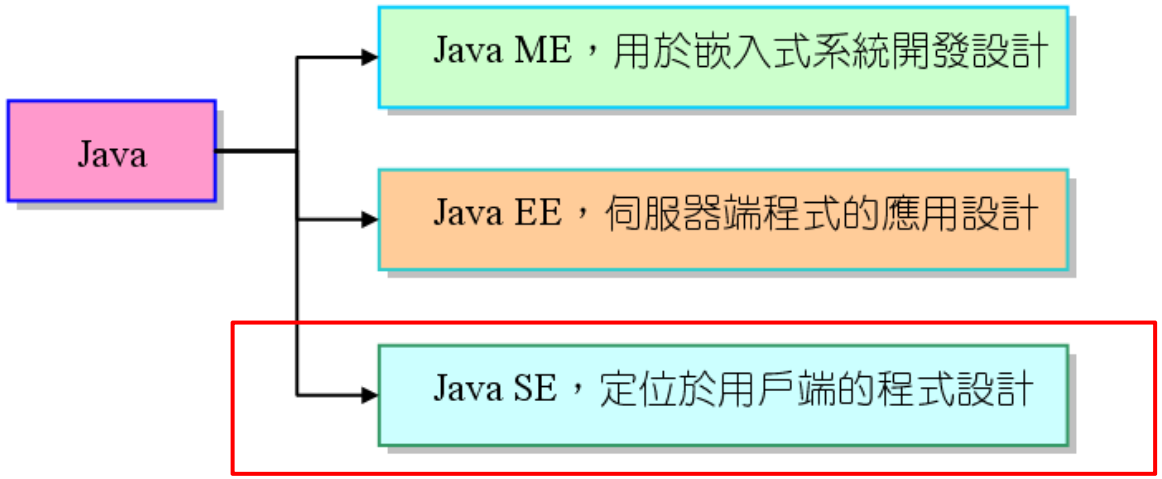


Java的歷史

- Java於1995年誕生，由美國加州的昇陽電腦公司（Sun Microsystems, Inc.）所推出
  - Java語言的前身是Oak，即橡樹之意
  - Sun將Oak修改成為Java，並將它轉移到Web上
  - Web於1995年開始盛行，Java也隨之一炮而紅
  - 2009年4月，SUN被Oracle（甲骨文）公司併購

Java的應用領域

- 下圖為Java的三大應用領域：



Micro Edition, Enterprise Edition, Standard Edition

Java的特點 (1/3)

- 嬌小且完美的語言
  - Java的語法簡潔、勻稱，容易撰寫且易於維護
- 具物件導向的功能
  - Java具有物件導向的特點
  - 它是全新的語言，因而可以將物件導向的特性發揮到極至
- 完全支援網際網路
  - Java所撰寫的程式很容易地在Browser裡呈現
  - 可藉此讓所有能夠上網的電腦都能執行Java程式

Java的特點 (2/3)

- 通用的語言
  - 凡是C/C++所能做到的事，Java也都能做到
- 跨平台的語言
  - 可在不修改程式碼的情況下，便能在不同的作業系統執行
- 具有豐富的函數庫
  - Java語言背後有龐大的函數庫支撐
  - 這些函數庫包含了繪圖函數庫、圖形使用者介面函數庫（graphical user interface）、網路設計函數庫...等等

Java的特點 (3/3)

特殊的處理機制

- 「多執行緒」機制可在同一時間執行不同的程序
- 「垃圾收集」機制可將無用的變數所佔用之記憶體釋放
- 「例外處理」機制可依情況拋出例外，使得程式不會因此中斷執行

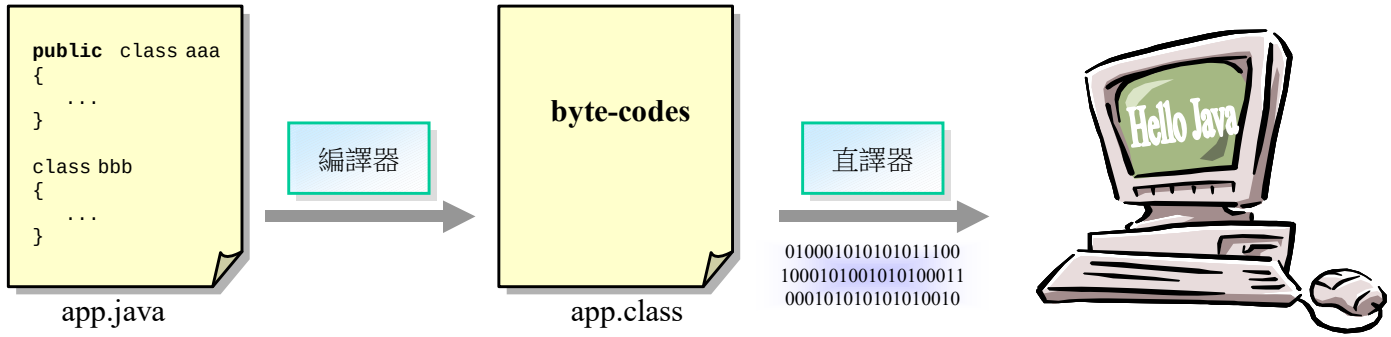
編譯與直譯

- 程式語言可分編譯式與直譯式兩種
  - 常見的編譯式語言如 C、FORTRAN與COBOL等均是
  - 直譯式的語言如 BASIC、JavaScript、VBScript、Python、Ruby 及其它Script語言

- Java程式的執行則是先編譯，後直譯

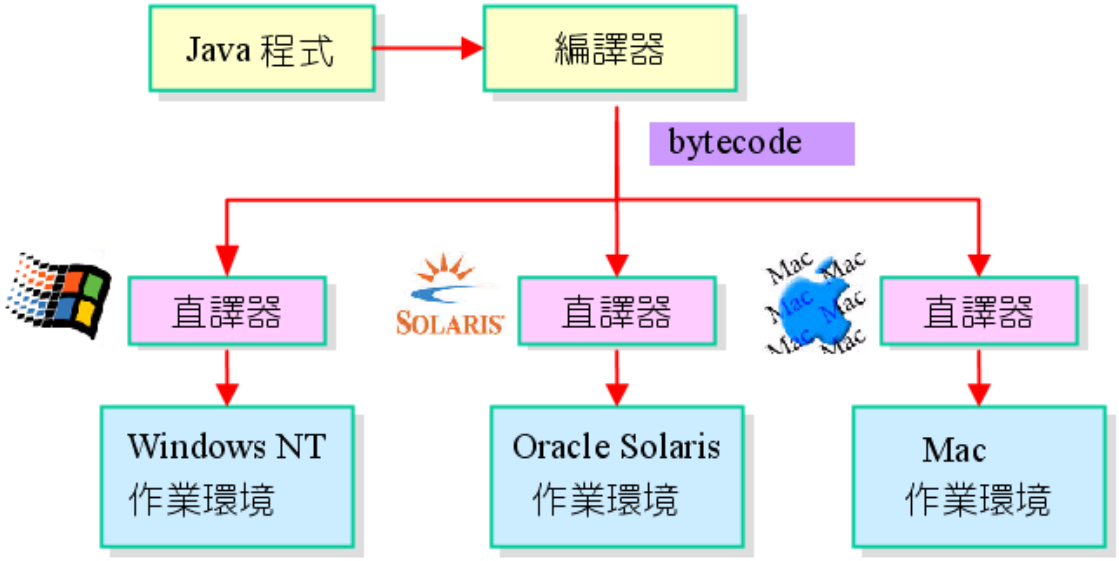
Java的位元碼

- Java的編譯與執行的程序:
  - 將程式碼編譯成與平台無關（platform-independent）的機器碼，稱之為「位元碼組」（byte-codes）
  - 經編譯後，可在裝有JVM上的平台直接執行
- 下圖說明Java相關的執行流程：



byte-codes 的執行

- byte-codes最大的好處是--可跨越平台來執行：



Java的重要性

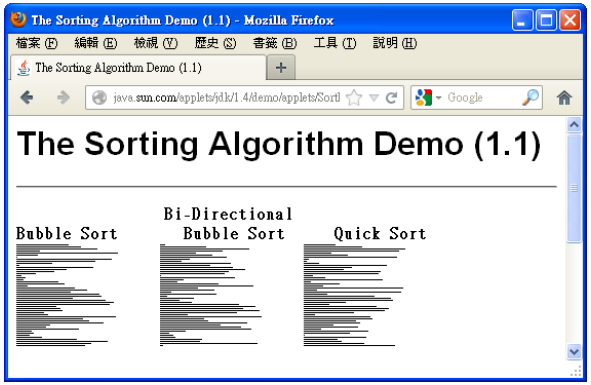
- 在學術領域裡，Java已成為最重要的教學程式語言之一
- 許多產品，如手機、PDA，甚至連小朋友玩的樂高（Lego）也是使用Java來延伸

Java程式的分類

- Java程式可分為下列兩種：
  - (1) Java應用程式（Java application）

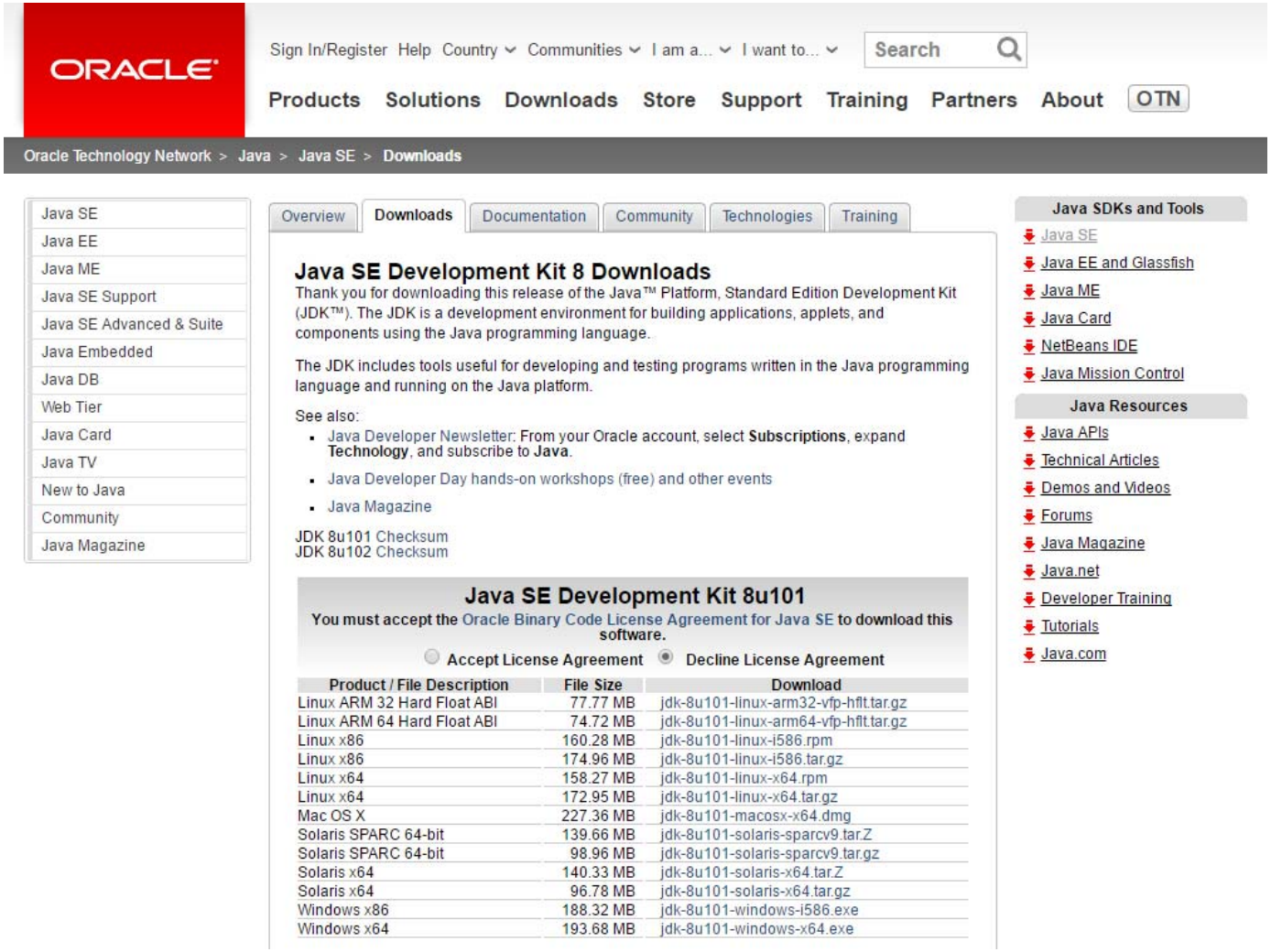
獨立的Application 程式

  - (2) 應用在www上的小程式（Java applet）

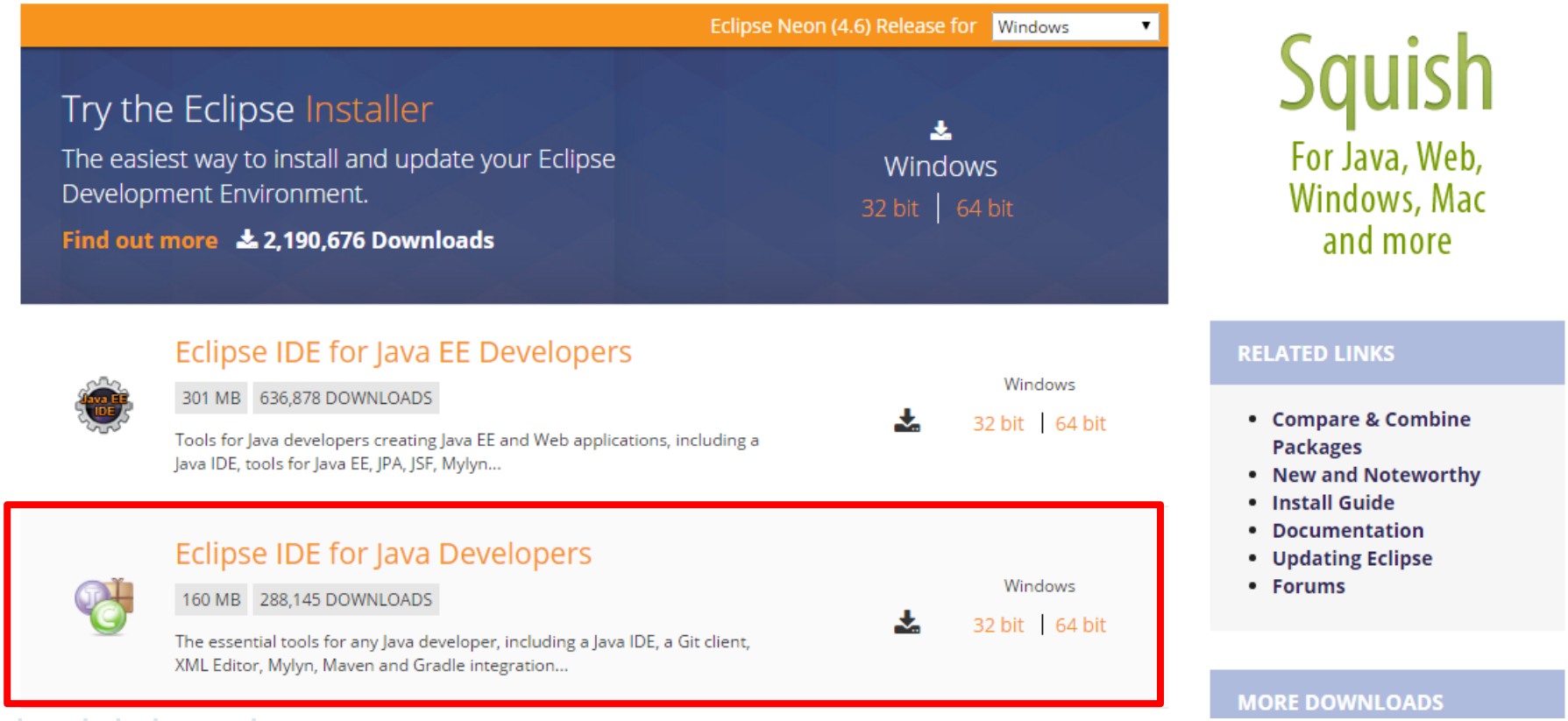


在網頁上執行的Applet 程式 (http://java.sun.com/applets/jdk/1.4/demo/applets/SortDemo/example1.html)

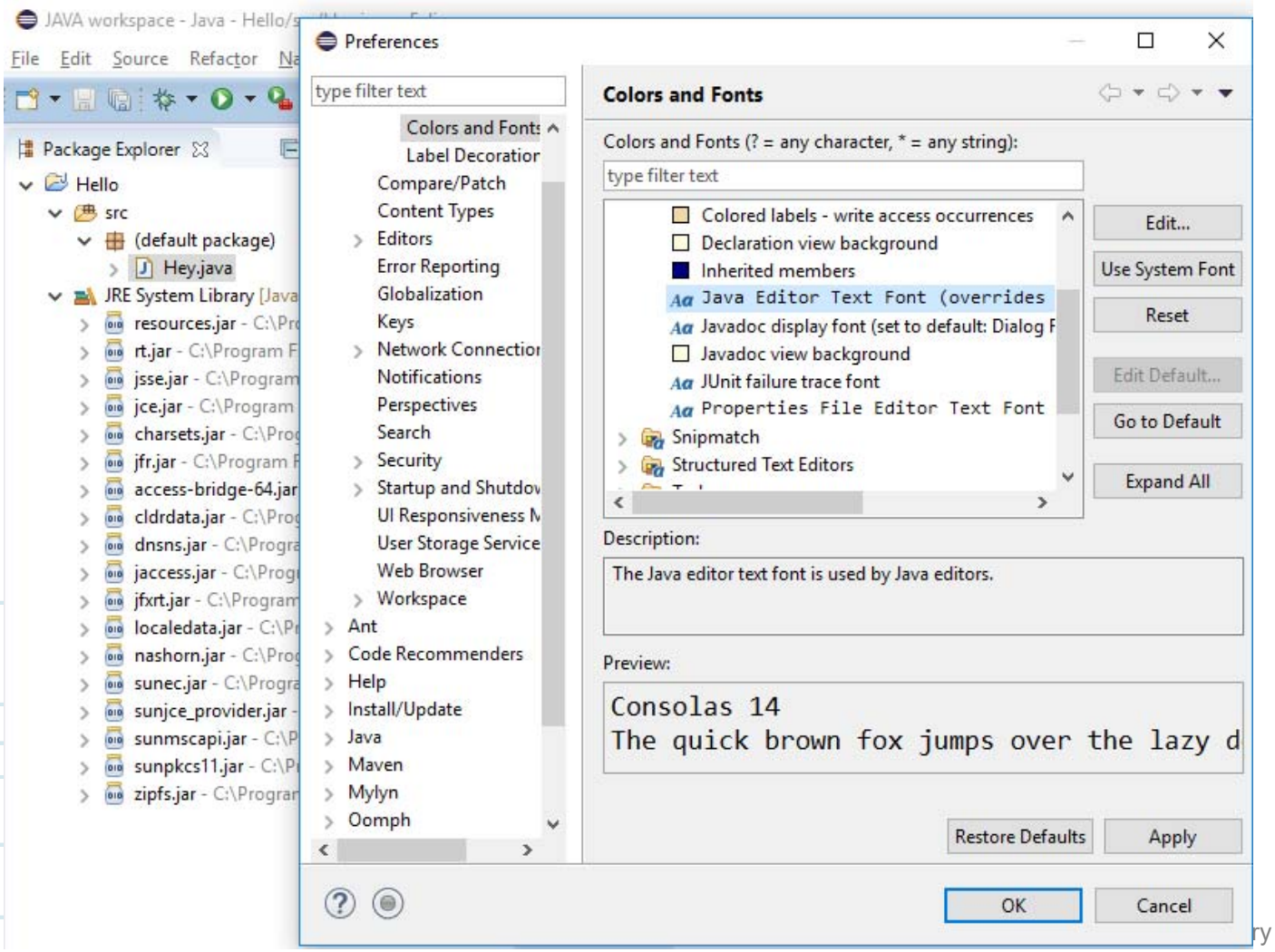
Install JAVA JDK



Install Eclipse

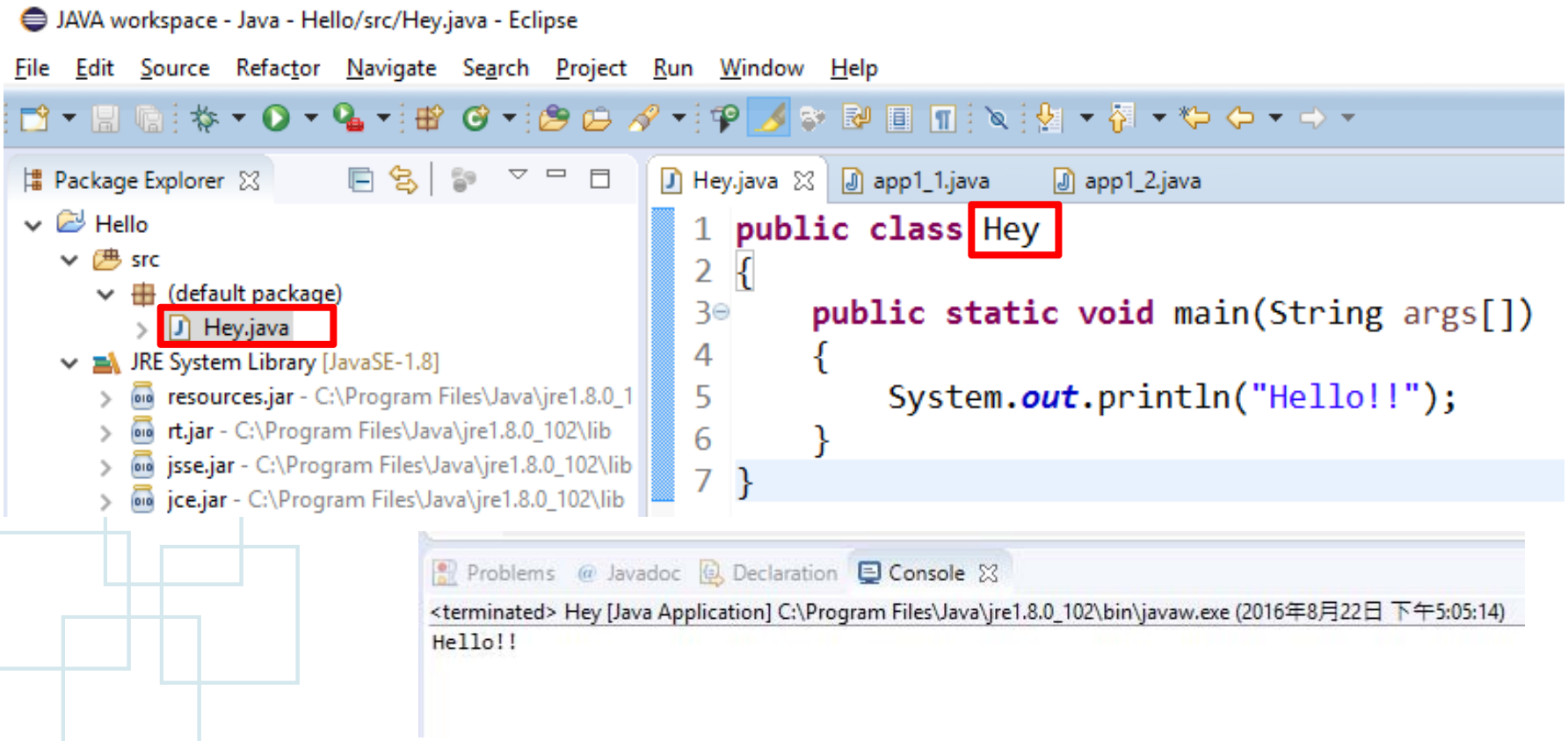


Modify Font Size and Color



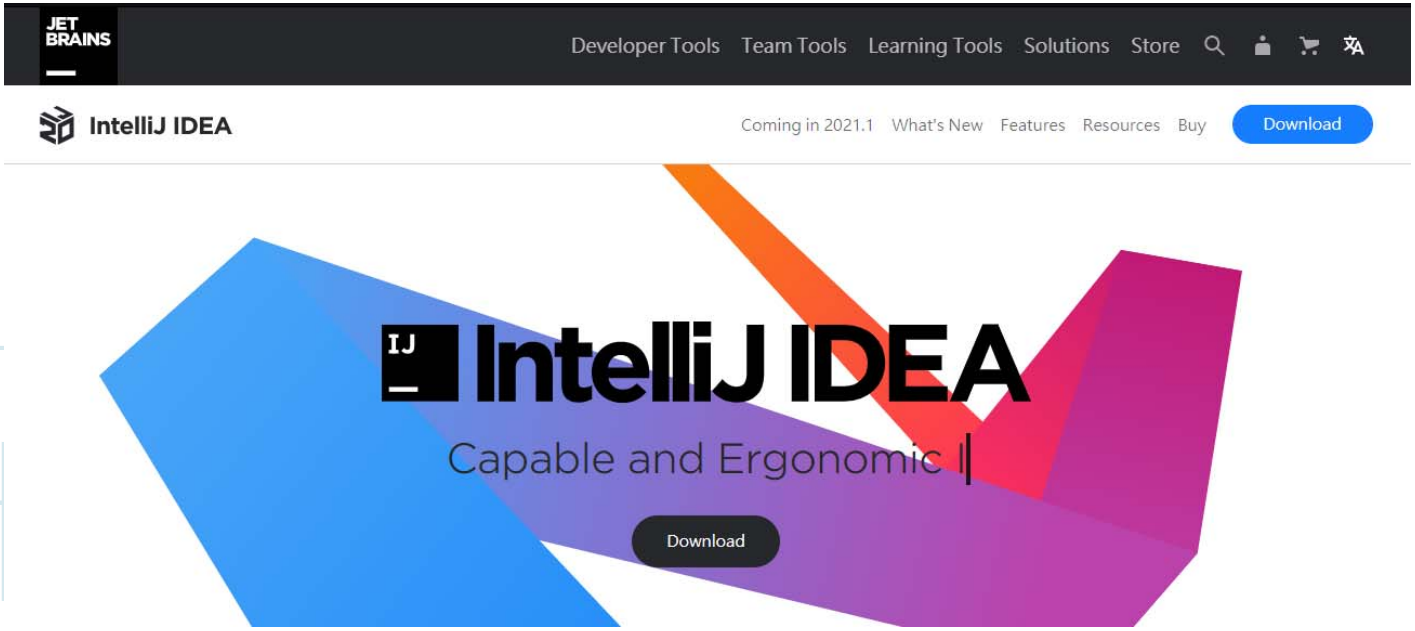
First JAVA Code

- 需一致



IntelliJ IDEA

- 提供Swing元件的圖形化GUI編輯工具
- 支援Android and iOS的應用程式開發，Google最新開發工具Android Studio就是基於IntelliJ IDEA的整合開發環境



Java 程式的起點--main

- main()方法是 Java 程式真正執行時的起點, 當 Java 程式執行時, 會從這個區塊內的程式開始, 循序執行, 一直到這個區塊結束為止。

```
01 public class SecondJava {
02    public static void main(String[] argv) {
03       System.out.println("這是我所寫的第二個 Java 程式，");
04       System.out.println("顯示更豐富的資訊囉！");
05    }
06 }
```

執行結果

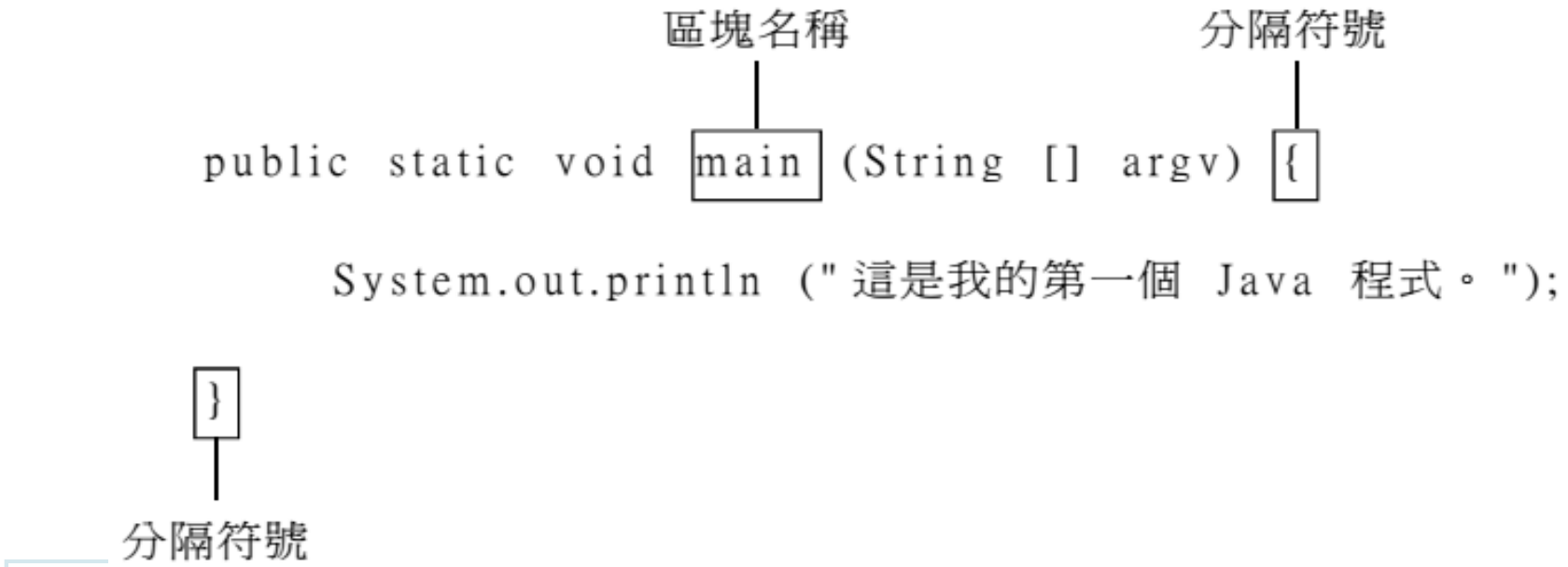
```
這是我所寫的第二個 Java 程式，
顯示更豐富的資訊囉！
```

敘述結尾分號

- 同一個敘述可以分成多行撰寫,和寫在同一行是一樣的效果；多個簡單敘述也可以寫在同一行， 結果和每一個敘述單獨撰寫成一行相同。

```
01 public class SecondJavaWithMultiLines {
02    public static void main(String[] argv) {
03       System.out.println
04       ("這是我所寫的第二個Java程式，");
05       System.out.println(
06       "顯示更豐富的資訊囉！");
07    }
08 }
```

分隔符號 (Separator)



為程式加上註解 (Comment)

- Adding comments for your code is important.

```
01 // 以下就是我們所要撰寫的第一個 Java 程式
02 public class CodeWithComment {
03     public static void main(String[] argv) {
04       // 到上面這兩行爲止都是固定的程式骨架
05
06       // 在 main()方法中就是我們要執行的程式
07       System.out.println(" 我的第一個 Java 程式。 ");
08
09       // 以下也都是固定的程式骨架
10     } // main() 方法的結束括號
11 }    // CodeWithComment 的結束括號
```

Executing in Command Prompt

- Javac xxx.java
- You may get an error message
- You should set the permissions of xxx.class
- Java Hey
- Then, you will get the executing result.

```
C:\Program Files\Java\jdk1.8.0_102\bin>javac Hey.java
Hey.java:1: error: error while writing Hey: Hey.class (Access is denied)
public class Hey
       ^
1 error

C:\Program Files\Java\jdk1.8.0_102\bin>javac Hey.java

C:\Program Files\Java\jdk1.8.0_102\bin>java Hey
Hello!!

C:\Program Files\Java\jdk1.8.0_102\bin>
```

Exercise 1

- Please write a Java code so that the screen can show the following result

```
*
**
***
****
*****
```

Exercise 2

- Please write a Java code so that the screen can show the following result

春眠不覺曉，處處聞啼鳥

夜來風雨聲，花落知多少

簡單的 Java 程式

- 下面的程式碼可印出兩行字串：

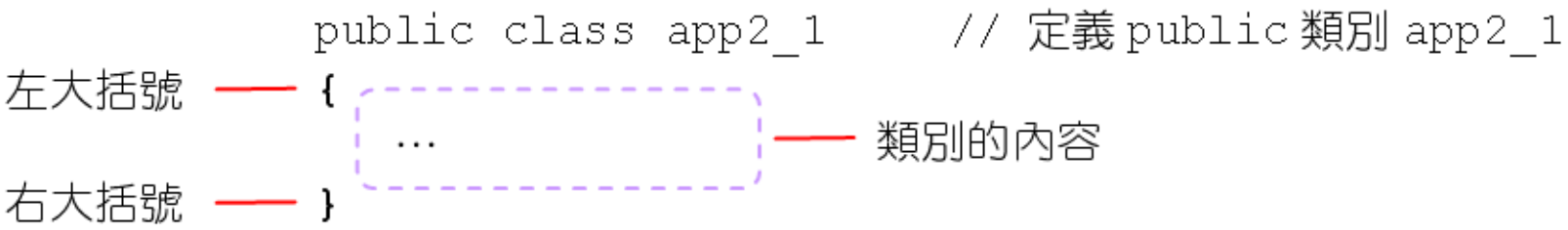
```
01  // app2_1, 簡單的 Java 程式
02  public class app2_1       // 定義 public 類別 app2_1
03  {
04     public static void main(String args[])  // main() 函數, 主程式開始
05     {
06        int num;             // 宣告整數 num
07        num=2;               // 將 num 設值為 2
08        System.out.println("I have "+num+" dogs");   // 印出字串及變數內容
09        System.out.println("You have "+num+" dogs, too");
10     }
11  }
```

```
/* app2-1 OUTPUT---

I have 2 dogs
You have 2 dogs, too
------------------*/
```

類別

- Java程式是由類別（class）所組成
  - 下面的程式片段即為定義類別的範例：



- public指的是對於類別的存取方式為共有
- 在完整的Java程式裡，至少需要有一個類別

程式執行的起始點

- 每一個Java程式必須有一個main()，而且只能有一個
- 通常main() 會是如下面的敘述片段：

```
public static void main(String args[])  // main()，主程式開始
{
  ...
}
```

- main() 之前必須冠上修飾子 public static void

變數使用的規則

- 變數的宣告：
  - ✓`int num;` // 宣告num為整數變數
  - ✓`int num,num1,num2;` // 同時宣告num,num1,num2為整數變數
- 變數的資料型態：
  - ✓char 字元 如 'a'、'A' 等
  - ✓String 字串 如 "Have a nice day!!" 等
  - ✓int 整數
  - ✓long 長整數
  - ✓short 短整數

  } 如12、-27 等
  - ✓float 單精度浮點數
  - ✓double 倍精度浮點數

  } 如12.762、-37.483 等

變數名稱與其限制

- 變數名稱可以是英文字母、數字或底線
  - 名稱中不能有空白字元（錯誤：my cats）
  - 第一個字元不能是數字（錯誤：2cats）
  - 變數有大小寫之分（mycats與MyCats是不同變數）

- 請依程式的需求來決定變數的名稱
  - 通常變數會以其所代表的意義來取名
  - 不能使用到關鍵字
  - 簡單的變數名稱會增加閱讀及除錯的困難度

literal

- Java把諸如整數12與字元 'a' 等，一眼便能看出其內容的數值稱為literal
- literal是
  - 數字（整數、浮點數）與
  - 文字（字元、字串）的總稱
- 我們偏好把literal翻成「字面值」，因為單看literal的字面，就可以知道它的內容為何

變數的設值

- 宣告的時候設值

```
int num=2;                    // 宣告變數，並直接設值
```

- 宣告後再設值

```
int num1,num2;                // 宣告變數
num1=2;                       // 設值給變數
num2=30;
```

- 在程式中適當的位置宣告變數並設值

```
for(int num=1; num<=10; num++)
{
   ...
}
```

需要用到變數時，再行宣告與設值

為什麼要宣告變數

- 宣告變數有許多好處，諸如：
  - 方便編譯器找到錯誤的變數名稱
  - 避免變數名稱打錯（如數字0與英文字母O）
  - 除錯容易
  - 增加程式的可讀性
  - 方便管理變數

println() 函數

- 使用println() 印出變數與字串：

```
01  // app2_2,使用 println()
02  public class app2_2                          // 定義 public 類別 app2_2
03  {
04     public static void main(String args[]) // main() 函數,主程式開始
05      {
06         int num=2;
07         System.out.println("I have "+num+" dogs"); // 印出變數及字串內容
08      }
09  }
```

印出 I have　印出 2　印出 dogs

```
/* app2_2 OUTPUT---
I have 2 dogs
-------------------*/
```

有引數的情形

- 下面的範例可接收兩個引數 "Tom" 與 "Jerry"：

```
01  // app2_3,有引數的情形
02  public class app2_3
03  {
04     public static void main(String args[])
05     {
06        System.out.println(          + " & " +          );
07     }
08  }
```

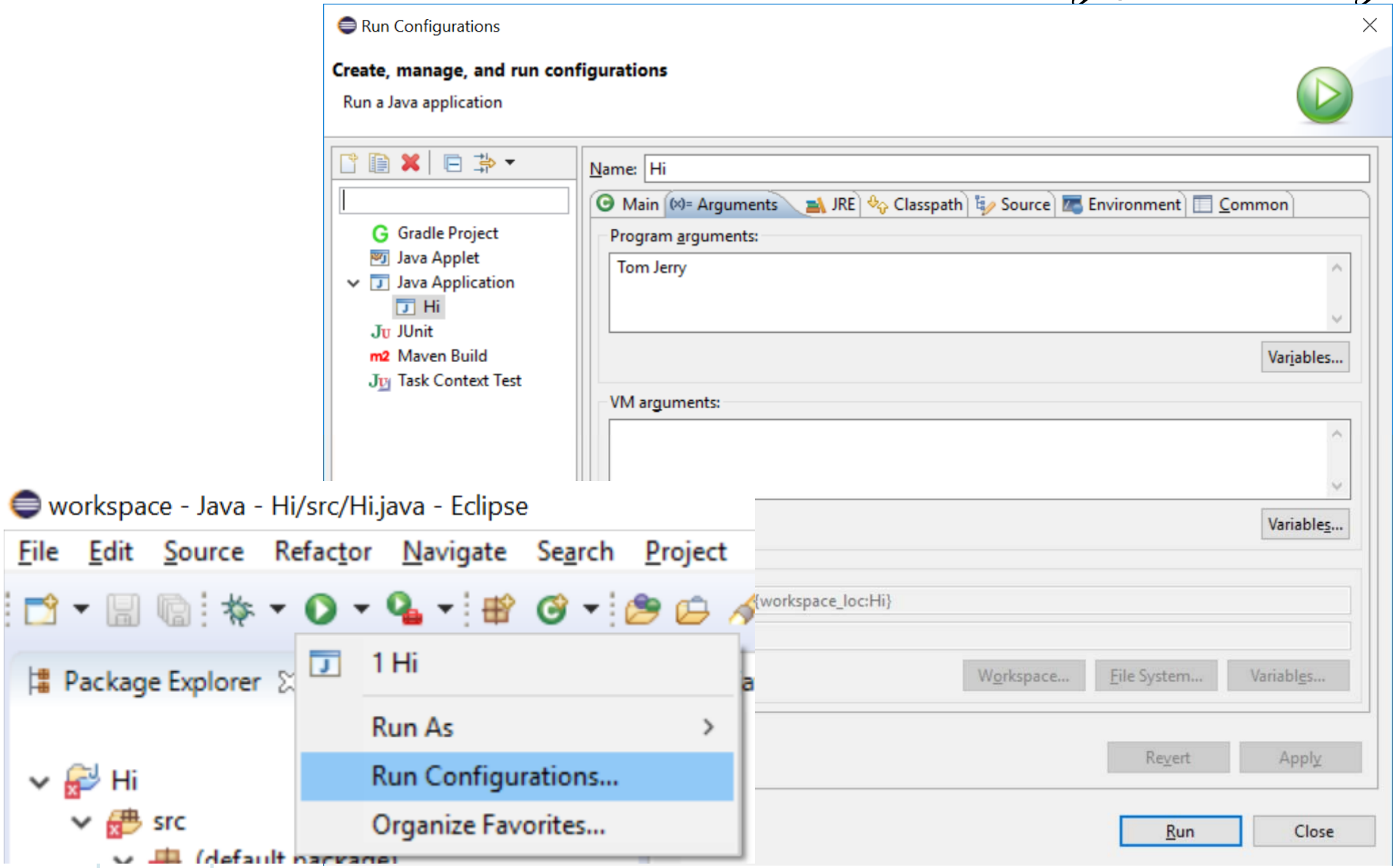
"Tom"由args[0]接收

"Jerry"由args[1]接收

```
/* app2_3 OUTPUT---

Tom & Jerry
----------------------*/
```

```
C:\Program Files\Java\jdk1.8.0_102\bin>java Hey Tom Jerry
TomHello!!Jerry
```


Run Configurations
Create, manage, and run configurations
Run a Java application
Gradle Project
Java Applet
Java Application
Hi
JUnit
Maven Build
Task Context Test
Name: Hi
Main
Arguments
JRE
Classpath
Source
Environment
Common
Program arguments:
Tom Jerry
Variables...
VM arguments:
Variables...
{workspace_loc:Hi}
Workspace...
File System...
Variables...
Revert
Apply
Run
Close
workspace - Java - Hi/src/Hi.java - Eclipse
File Edit Source Refactor Navigate Search Project
Package Explorer
Hi
src
1 Hi
Run As
Run Configurations...
Organize Favorites...

識別字 (identifier)

- 變數、常數、類別或者是函數的名稱為識別字
- 識別字的習慣命名原則：

識別字	命名原則	範例
常數	全部字元皆由英文大寫字母及底線組成	PI MAX_NUM
變數	英文小寫字母開始，若由數個英文單字組成，則後面的英文字由大寫起頭，其餘小寫	radius circleArea myPhoneNumber
函數	英文小寫字母開始，若由數個英文單字組成，則後面的英文字由大寫起頭，其餘小寫。函數和變數的命名方式相同，不同的是函數名稱後面會加上()	Show() addNum() mouseClicked()
類別	英文大寫字母開始，若由數個英文單字組成，則後面的英文字由大寫起頭，其餘小寫	Caaa CCustomer MaxSize

關鍵字（keyword）

- 關鍵字是編譯程式本身所使用的識別字
- Java提供的關鍵字如下：

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

程式錯誤的分類

- 語法錯誤（syntax error）
  - 程式含有不合語法的敘述，它無法被編譯程式翻譯
  - 編譯器可以檢查出語法錯誤

- 語意錯誤（semantic error）
  - 又稱邏輯錯誤，就是語法正確，但執行結果不對
  - 編譯器無法檢查出語意錯誤

語法錯誤

找出程式中的錯誤

```
01  // app2_4,有錯誤的程式
02  public class app2_4                    // 定義 public 類別 app2_4
03  {
04     public static void main(String args[])
05      {
06        int num1=2;                      // 宣告整數變數 num1，並設值為 2
07        int num2=3;                         宣告整數變數 num2，並設值為 3
08
09        System.out.println("I have "+num1+" dogs");
10        System.out.println("You have "+num2+" dogs")
11     )
12  }
```

```
/* app2_4 OUTPUT 除錯後的結果 ---

I have 2 dogs
You have 3 dogs
--------------------------------*/
```

語意錯誤

- 下面是語意錯誤的程式：

```
01  // app2_5,語意錯誤的程式
02  public class app2_5   // 定義 public 類別 app2_5
03  {
04     public static void main(String args[])
05      {
06        int num1=2;     // 宣告整數變數 num1，並設值為 2
07        int num2=3;     // 宣告整數變數 num2，並設值為 3
08
09        System.out.println("I have "+num1+" dogs");
10        System.out.println("You have "+num2+" dogs");
11        System.out.println("We have "+(num1-num2)+" dogs");
12     }
13  }
```

```
/* app2_5 OUTPUT---
I have 2 dogs
You have 3 dogs
We have -1 dogs
-------------------*/
```

提高程式的可讀性 (1/3)

- 將程式碼縮排，可提高可讀性

```
01  // app2_6, 有縮排的程式碼
02  public class app2_6        // 定義 public 類別 app2_6
03  {
04      public static void main(String args[])
05      {
06          System.out.println("Hello Java!");
07          System.out.println("Hello world!");
08      }
09  }
```

程式碼往內縮，因為整個 main() 函數是隸屬於 app2_6 類別

程式碼再往內縮，因為這兩行是屬於 main() 函數的內容

```
/* app2_6, 2_7 OUTPUT---

Hello Java!
Hello world!
-----------------------*/
```

提高程式的可讀性 (2/3)

- 未將程式碼縮排，閱讀起來較為困難

```
01  // app2_7, 沒有縮排的程式碼
02  public class app2_7        // 定義 public 類別 app2_7
03  {
04  public static void main(String args[])
05  {
06  System.out.println("Hello Java!");
07  System.out.println("Hello world!");
08  }
09  }
```

```
/* app2_6, 2_7 OUTPUT---

Hello Java!
Hello world!
-----------------------*/
```

提高程式的可讀性 (3/3)

- 將程式碼加上註解，有助於程式的閱讀與偵錯

```
// app2_8, examples
// created by wien hong
```

以「//」符號註解

```
/* This paragraph demonstrate the capability
   of comments used by Java   */
```

於「/*」和「*/」符號之間的文字均是註解