

# Java 程式設計進階

## Java 程式結構

鄭安翔

ansel\_cheng@hotmail.com

---

# 課程大綱

- 1) 物件導向程式設計
  - 主類別與主方法宣告
  - 物件導向軟體開發階段
  - 類別與物件
- 2) Java程式結構

# 物件導向程式設計理論

## ■ 物件導向程式設計

### □ 與傳統程序式程式設計不同

- 傳統的程式是一系列對電腦下達的指令(函式)

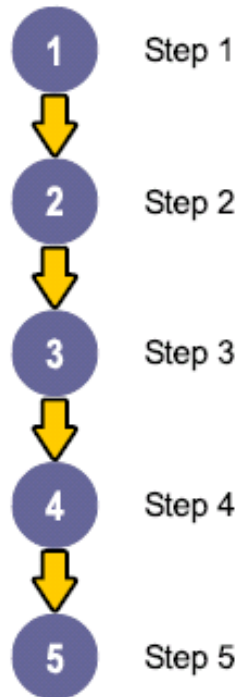
### □ 物件導向是一種抽象且擬人化的程式設計的方法

- 以物件作為程式的基本單元
- 物件將資料與操作封裝其中
- 物件能接受資料、處理資料並將資料傳達給其它物件
- 透過物件之間的交互作用來完成工作

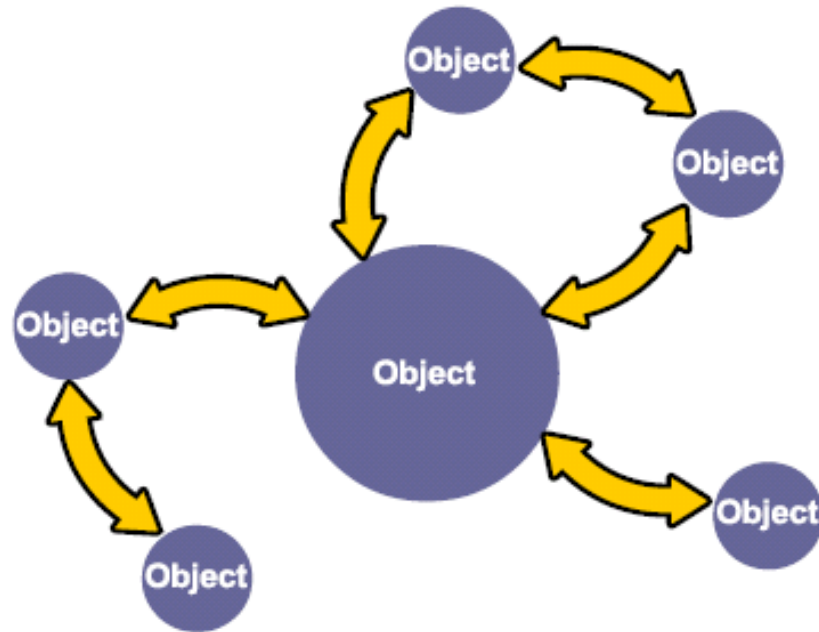
# 物件導向程式設計理論

- 物件導向程式設計
  - 物件導向程式設計優點
    - 物件(資料與流程)重複使用
    - 分散式開發
    - 提高了程式的靈活性和可維護性
    - 在大型專案設計中廣為應用

# 物件導向程式設計理論



**Procedural  
Language**



**Object-Oriented Language**

# 應用程式的主類別

## ■ 主類別

- 每個**Java**應用程式都需要一個主類別,作為程式的進入點,也稱為應用程式的啟始類別
- **Java SE**的應用程式中,主類別會包含 `main()` 方法
  - 主類別中的**main**方法,建立所需之其他物件
  - 利用物件之間的互動來完成工作

## ■ 類別在下列情況會加上`main()`方法

- 用來開始應用程式 (應用程式的起始類別)
- 執行程式來測試類別

```
01 public class OrderEntry{  
02     public static void main (String[] args) {  
03         Order order = new Order();  
04         Shirt s1 = new Shirt(.....);  
05     }  
06 }
```

```
01 public class Order {  
02     .....  
03 }
```

```
01 public class Shirt {  
02     .....  
03 }
```

# 程式的進入點 main Method

```
public static void main (String args[]) {  
    }  
}
```

Diagram illustrating the components of the `main` method signature:

- `public static`: Modifiers
- `void`: No return data
- `main`: Name of the method
- `(String args[])`: String array argument

- 符合標準的`main()`,才可被JRE當作程式的進入點
- Modifier
  - 可加上`final`
  - 順序可不同
- 字串陣列可用另一方法表示
  - `String [] args`
  - 字串陣列名稱不一定要為`args`
- `java classname ABC XYZ 123`

Diagram illustrating the mapping of command-line arguments to the `args` array:

```
java classname ABC XYZ 123
```

Arguments are mapped to the `args` array as follows:

- `ABC` → `args [0]`
- `XYZ` → `args [1]`
- `123` → `args [2]`

# 陳述句 Statement

- `main()` 方法中陳述句

- 建立所需之其他物件

- `Greeting hello = new Greeting();`

- 呼叫物件的方法來互動

- `hello.greet();`

- 在命令提示字元上印文字

- `System.out.println("Hi");`

- `System` 類別的標準輸出成員 `out` (`PrintStream`型態)所提供的 `println()` 方法

- 輸出文字用雙引號 `" "` 包括

- 陳述句結束時要記得加分號；



## HelloWorld.java

```
01 public class HelloWorld {  
02     public static void main (String[] args) {  
03         System.out.println("Hello World!");  
04     }  
05 }
```



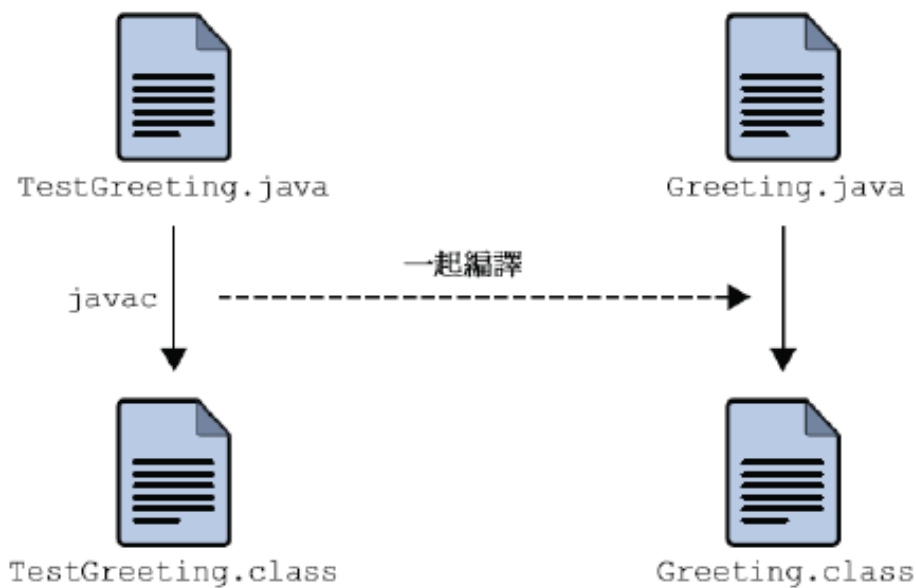
## Greeting.java

```
01 public class Greeting {  
02     public void greet() {  
03         System.out.println("Hello World!");  
04     }  
05 }
```

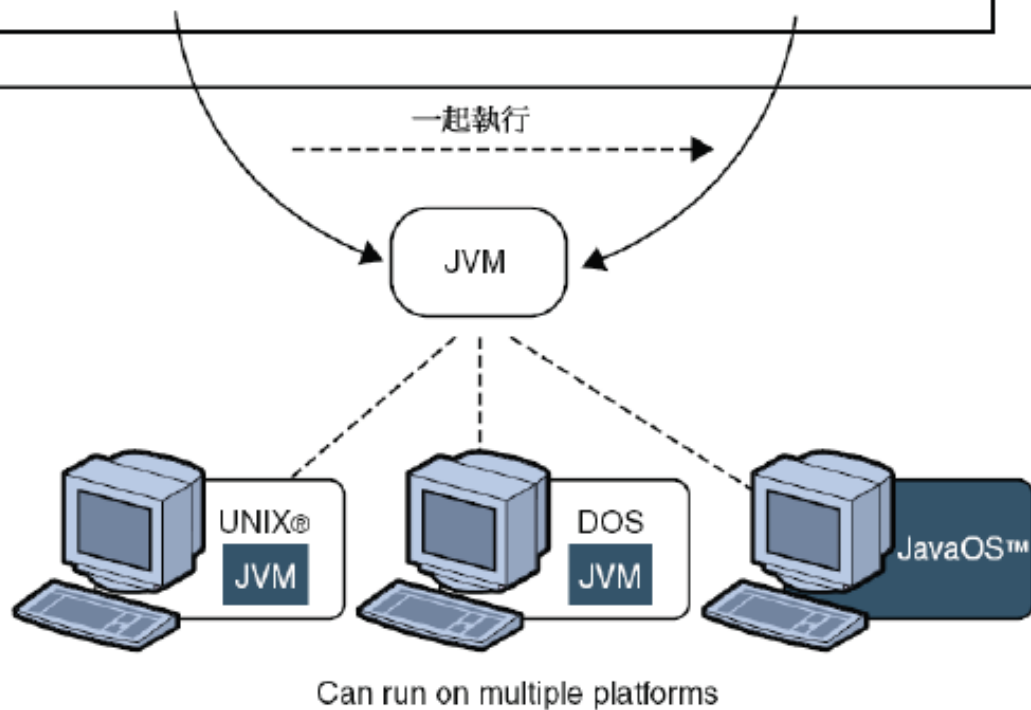
## TestGreeting.java

```
01 public class TestGreeting {  
02     public static void main (String[] args) {  
03         Greeting hello = new Greeting();  
04         hello.greet();  
05     }  
06 }
```

編譯

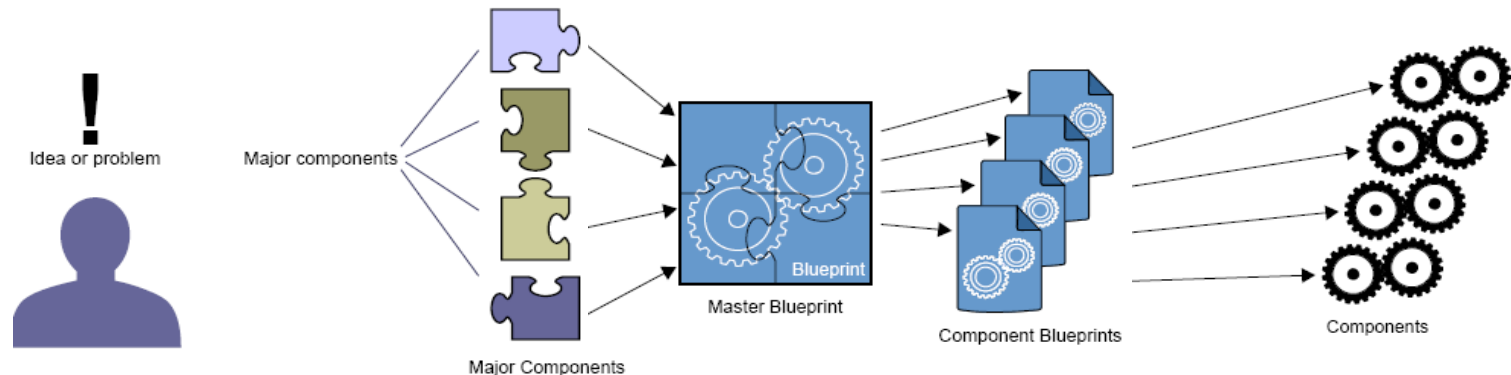


執行



# 物件導向軟體開發三階段

- Object Oriented Analysis : 物件導向分析
- Object Oriented Design : 物件導向設計
- Object Oriented Programming : 物件導向程式開發



# 物件導向分析

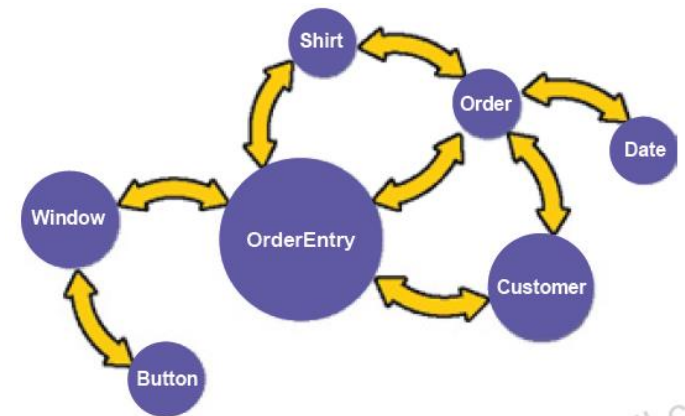
## ■ 使用物件導向的思考方式來分析應用系統需求

### □ 模組化

- 將系統拆分為多個可重複使用的物件
- 將需永續保存的資料,分門別類成為物件的屬性
- 將重覆使用的操作,獨立成物件的行為Operation

### □ 抽象化

- 將物件實際實作的細節隱藏
- 公開互動介面供其他物件使用



# 物件導向設計

## ■ Design Class

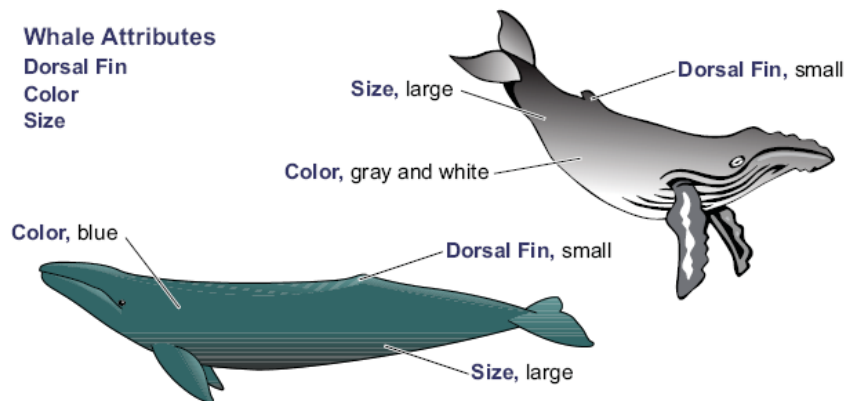
- 為物件設計類別(藍圖)

## ■ Modeling Class

- 視覺化所設計的類別

## ■ 重複使用類別元件

- JDK類別函式庫 Library
- 其他廠商開發之Java元件
- 自行開發的元件



Shirt
<b>+shirtID: int</b> <b>+colorCode: char</b> <b>+size: String</b> <b>+price: double</b> <b>+description : String</b>
<b>+Shirt (color: char, price: double, description: String)</b>
<b>+calculateShirtID() : int</b> <b>+displayInformation()</b>

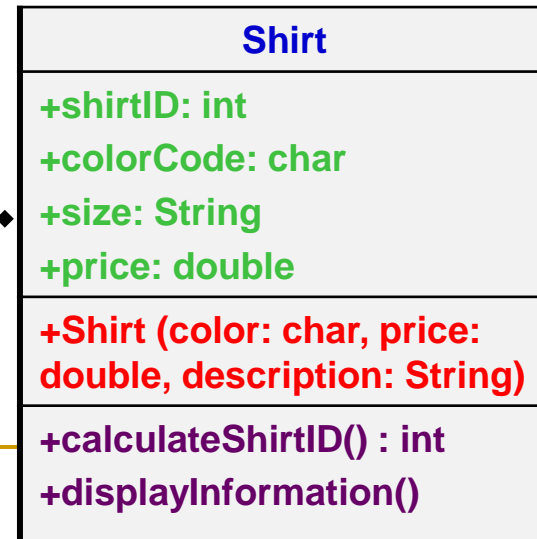
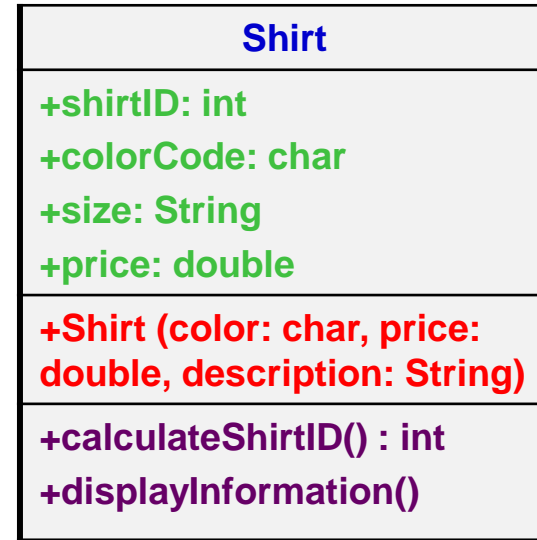
# UML (Unified Modeling Language)

## ■ 類別圖

## ■ 物件圖



## ■ 類別之間關係



# 物件概念：以 turtle 物件為例

## ■ turtle 物件

### □ 初學者在學習物件的教學工具

- 操控一隻烏龜(Turtle)在畫面(Playground)上移動
- 提供了視覺化操作的指令，完成繪圖或動畫

### □ 下載 jturtle

- <https://sourceforge.net/projects/jturtle/>
- jturtle-0.1.1.jar

# 設定ClassPath

- set ClassPath
  - jturtle-0.1.1.jar
  - %ClassPath%



```
命令提示字元
Microsoft Windows [版本 10.0.18362.535]
(c) 2019 Microsoft Corporation. 著作權所有，並保留一切權利。

D:\JavaClass\workspace>set ClassPath
CLASSPATH=.

D:\JavaClass\workspace>set ClassPath=D:\JavaClass\Exercises\ch1\
jturtle-0.1.1.jar;%ClassPath%

D:\JavaClass\workspace>set ClassPath
CLASSPATH=D:\JavaClass\Exercises\ch1\jturtle-0.1.1.jar;.

D:\JavaClass\workspace>
```



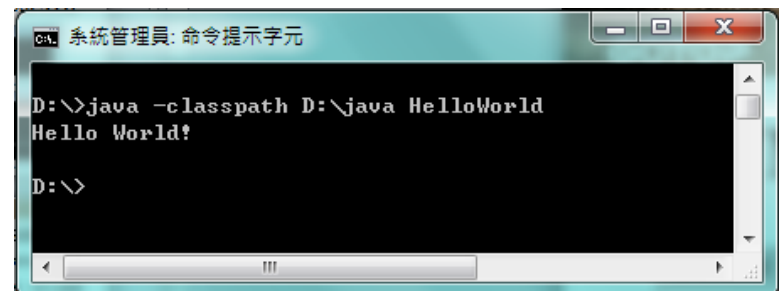
# Classpath

## ■ ClassPath

- ❑ 指定類別路徑資訊
- ❑ 在編譯或執行時期,告訴JVM到哪些路徑下尋找檔案
  - 原始檔(.java) / 類別檔(.class)



```
GA. 系統管理員: 命令提示字元
D:\Java>java HelloWorld
Hello World!
D:\Java>cd ..
D:\>java HelloWorld
錯誤: 找不到或無法載入主要類別 HelloWorld
D:\>
```



```
GA. 系統管理員: 命令提示字元
D:\>java -classpath D:\java HelloWorld
Hello World!
D:\>
```

# 建立Turtle物件

- 引入turtle套件
  - `import ch.aplu.turtle.*;`
- 建立Turtle物件
  - `Turtle name1 = new Turtle();`
    - 建立新的Turtle物件及其Playground
  - `Turtle name2 = new Turtle(name1);`
    - 在已存在的Playground中建立新的Turtle物件

# 物件屬性

## ■ 物件屬性

- 物件的資料狀態，也稱為實體屬性
- 每一物件各自擁有自己的資料,互不影響
- 透過物件名稱 . 讀取物件屬性

物件名稱.屬性

- 修改物件屬性

物件名稱.屬性 = 屬性值 ;

- 透過方法設定物件屬性

# Turtle 物件屬性

屬性	說明	設定方法	備註
color	烏龜顏色	setColor(java.awt.Color color)	java.awt.Color.BLACK, BLUE, GREY...
showTurtle	烏龜是否顯示	showTurtle() hideTurtle()	顯示烏龜 隱藏烏龜
pen	畫筆	setPenColor(Color color) setLineWidth(double lineWidth)	設定畫筆顏色 設定畫出線的寬度
penUp	畫筆是否提起	penUp() penDown()	畫筆提起，移動時不畫線 畫筆放下，移動時畫線
position	位置	setPos(double x, double y)	烏龜相對於中心點 (0.0, 0.0) 的位置
speed	移動速度	speed(double newSpeed)	速度單位為pixels/sec
angle	移動方向	setHeading(double degrees) right(double degrees) left(double degrees)	設定烏龜移動方向，0 向北，單位為角度 向右(順時針)轉指定角度 向左(逆時針)轉指定角度

# 物件方法

## ■ 物件方法

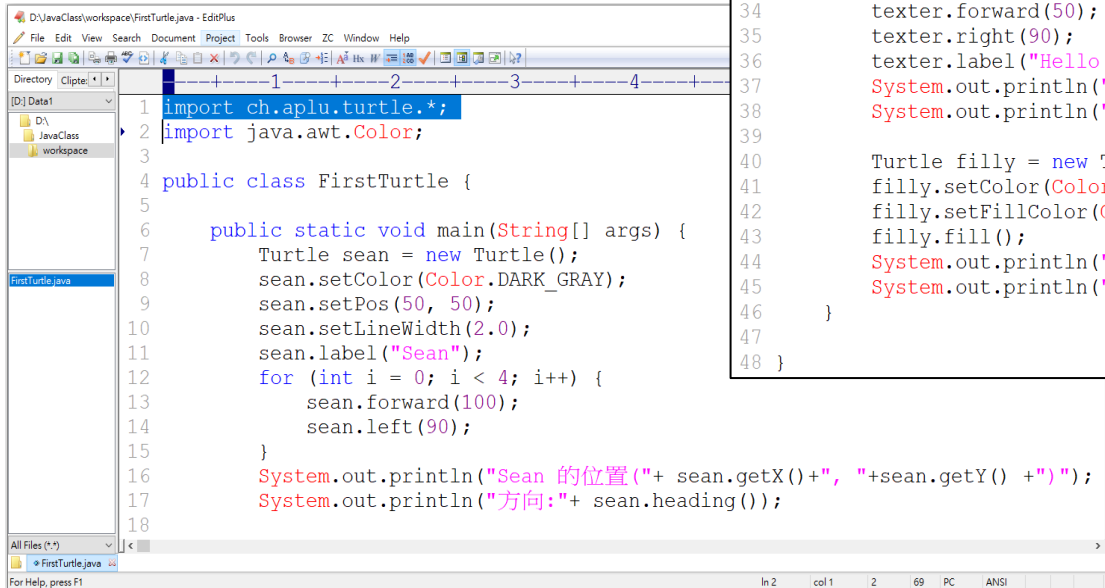
- 物件可提供的操作行為
- 需透過指定物件變數來呼叫
  - 不同物件的執行結果可能不相同
- 呼叫時需依方法定義傳入要求的參數
- 並宣告變數接收傳回值

變數 = 物件名稱.方法名稱(傳入參數...);

# Turtle 常用方法

方法	傳回值	說明
forward(double distance)	Turtle	向前方移動 distance 個像素距離
bk(double distance)	Turtle	向後方移動 distance 個像素距離
right(double degrees)	Turtle	向右順時針旋轉 degrees 度
left(double degrees)	Turtle	向左逆時針旋轉 degrees 度
setPos(double x, double y)	Turtle	移動到距中心點(x , y)的位置
home()	Turtle	移動到中心點 (0, 0) 位置
getX() / getY()	double	傳回目前位置 x軸或 y 軸的值
heading()	double	取得目前移動方向，0 向北，單位為角度
label(String text)	Turtle	在烏龜牌標示指定文字
distance(double x, double y)	double	計算目前位置距離指定(x,y)位置的距離
clean()	Turtle	清除所有繪製的線

# Turtle 範例



```
1 import ch.aplu.turtle.*;
2 import java.awt.Color;
3
4 public class FirstTurtle {
5
6     public static void main(String[] args) {
7         Turtle sean = new Turtle();
8         sean.setColor(Color.DARK_GRAY);
9         sean.setPos(50, 50);
10        sean.setLineWidth(2.0);
11        sean.label("Sean");
12        for (int i = 0; i < 4; i++) {
13            sean.forward(100);
14            sean.left(90);
15        }
16        System.out.println("Sean 的位置(" + sean.getX() + ", " + sean.getY() + ")");
17        System.out.println("方向:" + sean.heading());
18    }
19 }
```

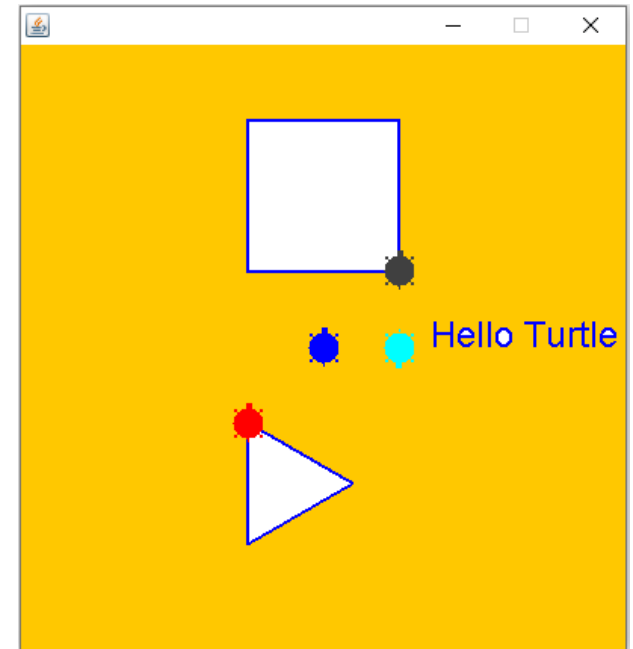
```
19 Turtle amy = new Turtle(sean);
20 amy.setColor(Color.RED);
21 amy.setPos(-50, -50);
22 amy.speed(50);
23 amy.label("Amy");
24 for(int i=0; i<3; i++){
25     amy.right(120);
26     amy.back(80);
27 }
28 System.out.println("Amy 的位置(" + amy.getX() + ", " + amy.getY() + ")");
29 System.out.println("方向:" + amy.heading());
30
31 Turtle texter = new Turtle(sean);
32 texter.heading(90);
33 texter.penUp();
34 texter.forward(50);
35 texter.right(90);
36 texter.label("Hello Turtle");
37 System.out.println("Texter 的位置(" + texter.getX() + ", " + texter.getY() + ")");
38 System.out.println("方向:" + texter.heading());
39
40 Turtle filly = new Turtle(sean);
41 filly.setColor(Color.BLUE);
42 filly.setFillColor(Color.ORANGE);
43 filly.fill();
44 System.out.println("Filly 的位置(" + filly.getX() + ", " + filly.getY() + ")");
45 System.out.println("方向:" + filly.heading());
46 }
47
48 }
```

# Turtle 範例

```
命令提示字元 - java FirstTurtle

D:\JavaClass\workspace>javac FirstTurtle.java

D:\JavaClass\workspace>java FirstTurtle
Sean 的位置(49.999999999999986, 49.99999999999998)
方向:-360.0
Amy 的位置(-49.99999999999999, -49.99999999999998)
方向:360.0
Texer 的位置(50.0, 3.061616997868383E-15)
方向:180.0
Filly 的位置(0.0, 0.0)
方向:0.0
```





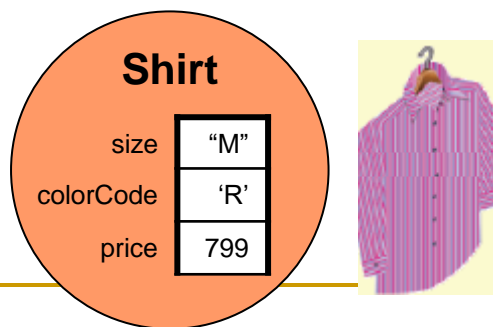
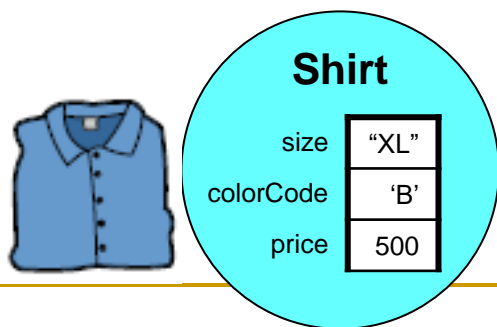
# 類別 vs. 物件

## ■ 類別

- 程式設計師以類別來定義同類型物件的共同藍圖
- 在**Java**中類別也可以是一種型別定義

## ■ 物件

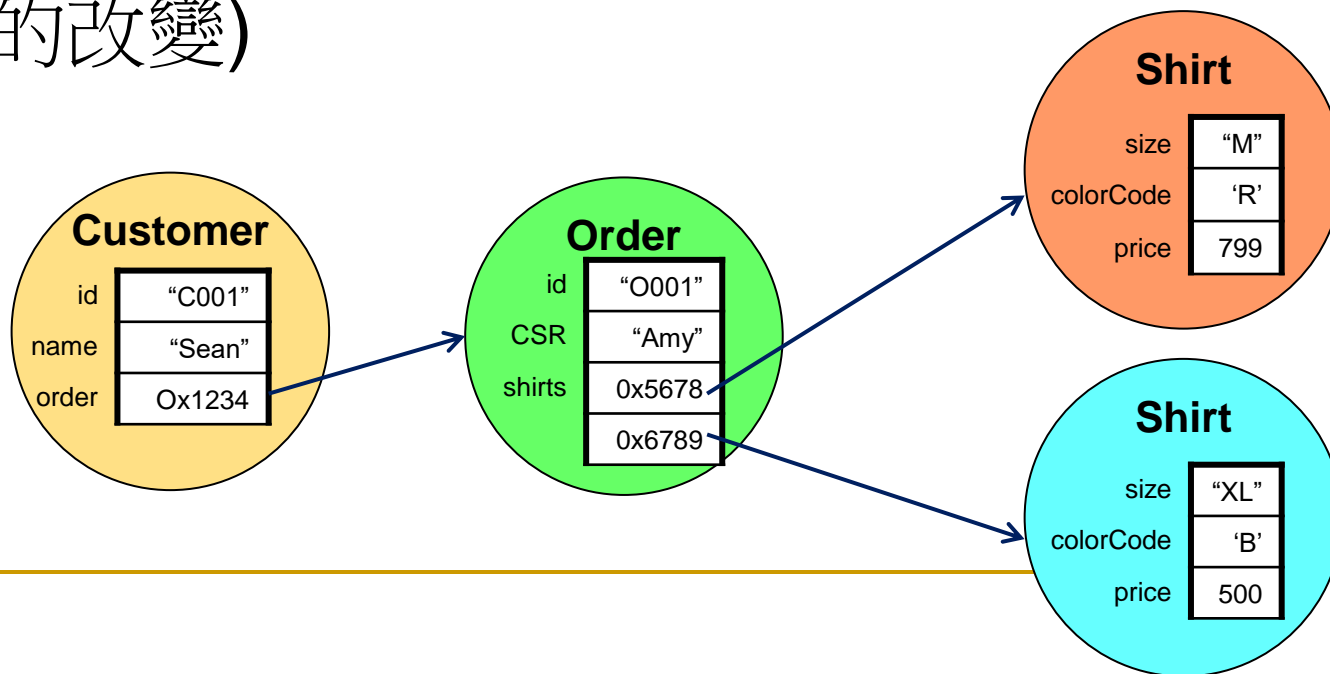
- 物件是類別的一個實體
- 兩件衣服是同一個類別的不同實體



Shirt
+shirtID: int +colorCode: char +size: String +price: double
+Shirt (color: char, price: double, description: String)
+calculateShirtID() : int +displayInformation()

# 類別 vs. 物件

- **Java programmer** 設計類別及類別之間的互動關係.
- **Java** 應用程式執行時，**JVM** 根據類別定義建立物件(系統中配置記憶體)，並處理物件之間互動產生之的狀態變化(記憶體中的儲存值的改變)



# 課程大綱

1) 物件導向程式設計

2) **Java**程式結構

- 類別 **Class**
- 屬性 **Attributes**
- 方法 **Methods**
- 撰寫註解及空白
- 測試及執行**Java**程式

---

# Java 程式結構

- 類別 Class
  - 屬性 Attributes
  - 方法 Methods
  - 建構子 Constructor
  - 註解 Comments
  - 程式進入點 `main()` 方法
-

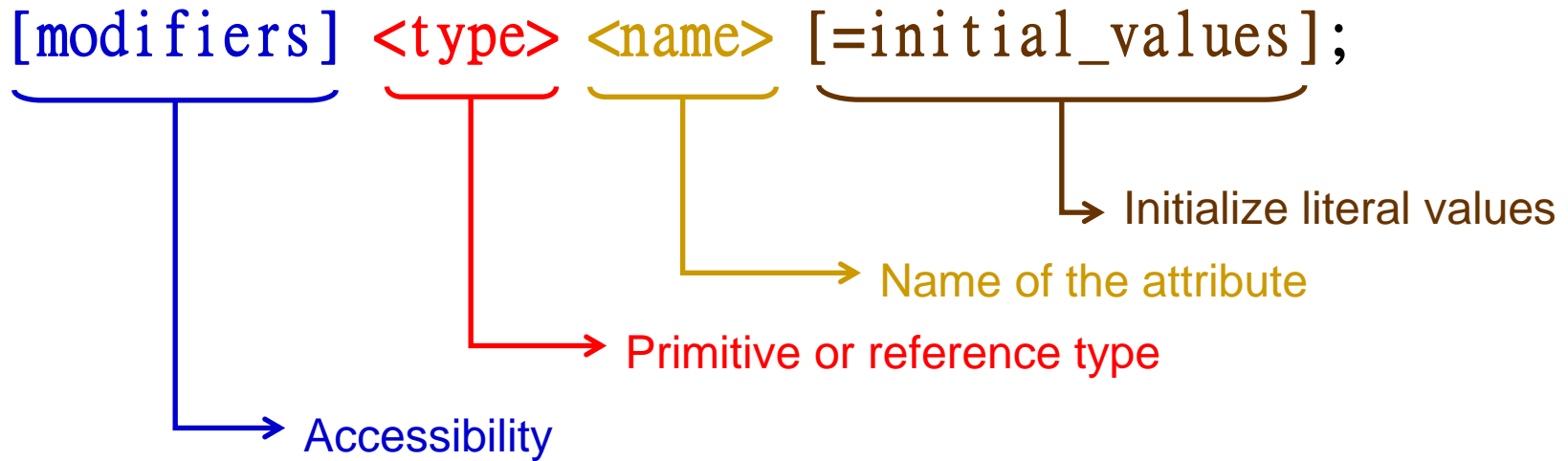
# Class Declaration 類別宣告

`[modifiers] class <class_name> {`  
} `Accessability`  
`Keyword`  
`Name of the class`

The diagram illustrates the syntax of a Java class declaration. The text `[modifiers] class <class_name> {` is shown with three colored brackets underneath: a blue bracket under `[modifiers]`, a red bracket under `class`, and a yellow bracket under `<class_name>`. Arrows point from these brackets to labels: a blue arrow from the blue bracket points to the label `Accessability`, a red arrow from the red bracket points to the label `Keyword`, and a yellow arrow from the yellow bracket points to the label `Name of the class`. The closing brace `}` is shown on the line below the opening brace.

- Modifier為public的類別,需存在class\_name.java的檔案中
- 一個Java檔案,可包含一個以上的類別宣告
- 多個類別宣告在一個Java檔案中,只能有一個public的類別

# Attribute Declaration 屬性宣告

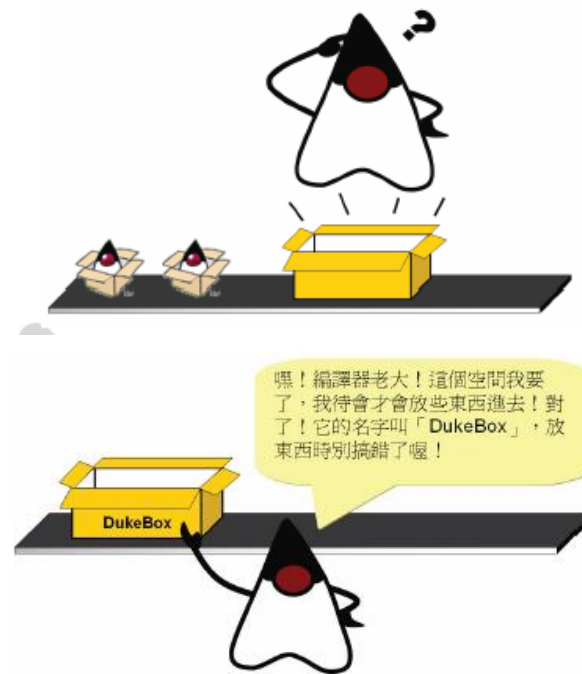


- 程式中用變數來存放需使用到的資料
- 變數定義在class body內用來表示屬性
- 存取權限modifier有public, protected, (default), private

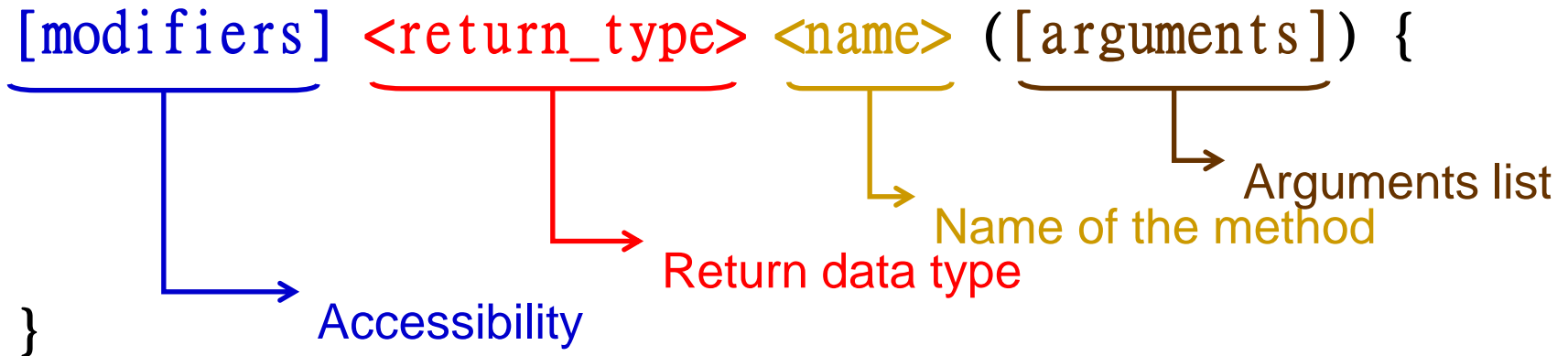
# 變數宣告 Variable Declaration

**<type>** **<identifier>** [=initial\_values];

- 資料型態
  - 適當記憶體儲存空間
- 變數名稱
  - 以該名稱來取得儲存值
- = 指定運算子
  - 右邊的值存到左邊的記憶體



# Method Declaration 方法宣告



- 存取權限有public, protected, (default), private
- 加上**static**的稱為「類別方法」,反之稱為「物件方法」
- 傳回值型態需與方法區段內**return**的資料型態相符
  - 方法如沒有傳回值,傳回值型態為void
- 傳入參數列(Arguments List)
  - 格式為Type Name
  - 可有0~N個,超過一個,用 , 隔開



# Constructor 建構子宣告

```
[modifiers] class <class_name> {  
    [modifiers] constructor_name ([arguments]) {  
        code_blocks  
    }  
}
```

Diagram illustrating the components of a constructor declaration:

- [modifiers]** (blue): Accessibility
- constructor\_name** (red): Same as class\_name
- [arguments]** (green): Arguments list

- 與類別名稱一樣
- 沒有回傳型態
- 預設建構子
- 可以多載(Overloading)

# Shirt類別

## ■ 屬性 (欄位)

屬性名稱	資料型態	說明	欄位宣告
shirtID	int	唯一的衣服編號	<b><i>public int shirtID;</i></b>
colorCode	char	衣服顏色代碼	<b><i>public char colorCode;</i></b>
size	String	衣服尺寸	<b><i>public String size;</i></b>
price	double	衣服價格	<b><i>public double price;</i></b>
description	String	衣服描述	<b><i>public String description;</i></b>

## ■ 建構子

建構子宣告	說明
<b><i>public Shirt(char c, String s, double p, String d) {...}</i></b>	建立衣服物件

## ■ 方法 (操作)

方法宣告	說明
<b><i>public void setPrice(double p) {...}</i></b>	設定衣服的價格
<b><i>public double getPrice( ) {...}</i></b>	取得衣服的價格
<b><i>public void displayInformation() {...}</i></b>	顯示衣服的資訊

# Java 程式結構

Shirt
+shirtID: int +colorCode: char +size: String +price: double +description : String
+Shirt (c: char, s: String, p: double, d: String)
+setPrice(double p) +getPrice ( ) : double +displayInformation ( )

01  
02  
03  
04  
05  
06  
07  
08  
09  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30

```
public class Shirt {
```

```
    public int shirtID = 0;
    public char colorCode = 'G';
    public String size = "XL" ;
    public double price = 299.00;
    public String description = "Polo Shirt";
```

物件屬性  
(欄位)

```
    public Shirt(char color, String size,
        double price, String desc) {
        colorCode = c;
        size = s;
        price = p;
        description = d;
    }
```

建構子

```
    public void setPrice(double p) {
        price = p;
    }
    public double getPrice ( ) {
        return price;
    }
    public void displayInformation() {
        System.out.println("Shirt ID:" + shirtID);
        System.out.println("Color:" + colorCode);
        System.out.println("Size:" + size);
        System.out.println("Price:" + price);
    }
```

物件方法  
(操作)

```
}
```

# 單行註解

## ■ 單行註解

```
int shirtID = 0; //Default ID for the shirt  
// The color codes are R= Red, B= Blue  
char colorCode = 'R';
```

```
01 public class SingleCommentDemo {  
02     public static void main(String[] args) {  
03         //顯示 Hello World!  
04         System.out.println("Hello World!");  
05     }  
06 }
```

# 多行註解

## ■ 多行註解

```
/*  
    this is a multi-line comment  
*/
```

01	/*
02	名稱:第一個java程式練習
03	目的:在螢幕上顯示Hello World!
04	*/
05	public class TraditionalCommentDemo {
06	public static void main(String[] args) {
07	System.out.println("Hello World!");
08	}
09	}

# 註解的用途及風格

## ■ 註解的用途

- 輔助程式設計人員閱讀程式

## ■ 常見註解風格

- 變數宣告時使用註解,表明變數的作用

```
int numberOfStudent; //學生的學號
```

- 類別和方法的最後一行加上單行註解,使區塊範圍明顯

```
public class HelloWorld {  
    public static void main(String[] args){  
        ...  
    } //main 結束  
} // HelloWorld 結束
```

# 常見註解風格

- 遇到暫時不想執行的陳述句,可用註解將其註銷,編譯器就不會去處理該指令

```
public class HelloWorld {  
    public static void main(String[] args){  
        //System.out.println(" Hello World !!! ");  
        System.out.println(" 你好 !!! ");  
    }  
}
```

- 程式的開頭以標題的方式說明程式的名稱、目的

```
/*  
 * 變數宣告範圍  
 * Variable Declaration Section  
 */
```

# JavaDoc 文件註解

## ■ JavaDoc 註解

### □ 語法：

```
/**
```

The following comment will be show in JavaDoc

```
*/
```

### □ 提供一個方便的工具,讓程式與文件內容保持同步

### □ JavaDoc 工具說明：[docs/technotes/guides/javadoc/index.html](https://docs.oracle.com/javase/8/docs/technotes/guides/javadoc/index.html)

```
01  /**
02     這是我的第一支Java程式,一個有禮貌的男孩
03     @version 1.0
04     @see HelloBoy
05     @author Sean Cheng
06  */
07  public class Boy {
08      /**男孩的姓名屬性*/
09      public String name = "大雄";
10      /**男孩的建構子*/
11      public Boy(){
12      }
13      /**男孩向人問好,並介紹自己*/
14      public void greet(){
15          System.out.println("你好!!!我是" + name);
16      }
17  }
```



javaDoc.exe

Package Class Tree Deprecated Index Help

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

### Class Boy

java.lang.Object  
Boy

---

```
public class Boy
extends java.lang.Object
```

這是我的第一支Java程式,一個有禮貌的男孩

See Also:

HelloBoy

---

### Field Summary

Fields

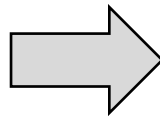
Modifier and Type	Field and Description
java.lang.String	name 男孩的姓名屬性



# 空白 White Space

- 程式設計師可在程式中加入任意數量的空白，包括 **tabs**、**spaces**、換行，以增加程式的可讀性

```
{int x; x=23*54;}
```



```
{  
    int x;  
    x = 23 * 54;  
}
```