Method

方法

Java Fundamental

類別與方法

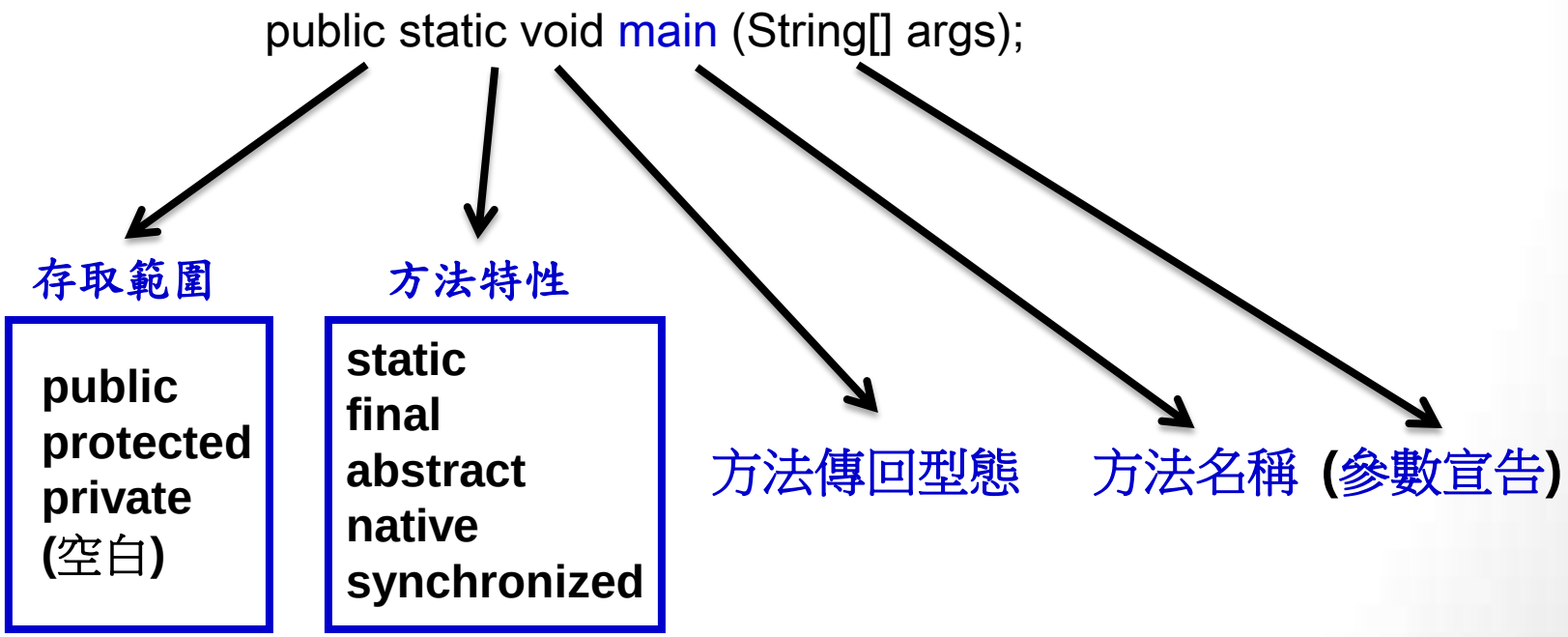
Shirt
+shirtID: int +colorCode: char +size: String +price: double +description : String
+Shirt (color: char, size: String, price: double, description: String)
+displayInformation ( ) +setPrice(p: double) +getPrice ( ) : double

```
01  public class Shirt {
02
03     public int shirtID = 0;
04     public char colorCode = 'R';
05     public String size = "XL" ;
06     public double price = 299.00;
07     public String description = "Polo Shirt";
08
09     public Shirt(char color, String size,
10                  double price, String desc) {
11        this.colorCode = color;
12        this.size = size;
13        this.price = price;
14        this.description = desc;
15     }
16
17     public void displayInformation() {
18        System.out.println("Shirt ID:" + shirtID);
19        System.out.println("Color:" + colorCode);
20        System.out.println("Size:" + size);
21        System.out.println("Price:" + price);
22     }
23     public void setPrice(double p) {
24        price = p;
25     }
26     public double getPrice( ) {
27        return price;
28     }
29
30  }
```

物件方法
(操作)

方法 (Method)

- 定義
  - 可重複使用的程式碼片段
- 格式



Method

- Method 命名習慣
  - 應該採用英文動詞，可以串接多個單字：整個字串的第一個字母小寫，其他串接單字的第一個字母大寫
    - 如：toString(), compareTo(), setX(), getX()…
  - Method 名稱不能和 class 名稱一樣，因 class 名稱保留給建構式使用
- 將資料傳入method – 參數
  - 只要是合法的 Java 資料型態，都可以傳入 method 中，例如：double, float 與 int，或是物件
  - Java 中不允許將 methods 當成參數，傳入另一個 method 中

Method

- 在 Java 中，方法只能在類別中被創造
- 方法分成2個部分：
  - 方法簽章
  - 方法內容

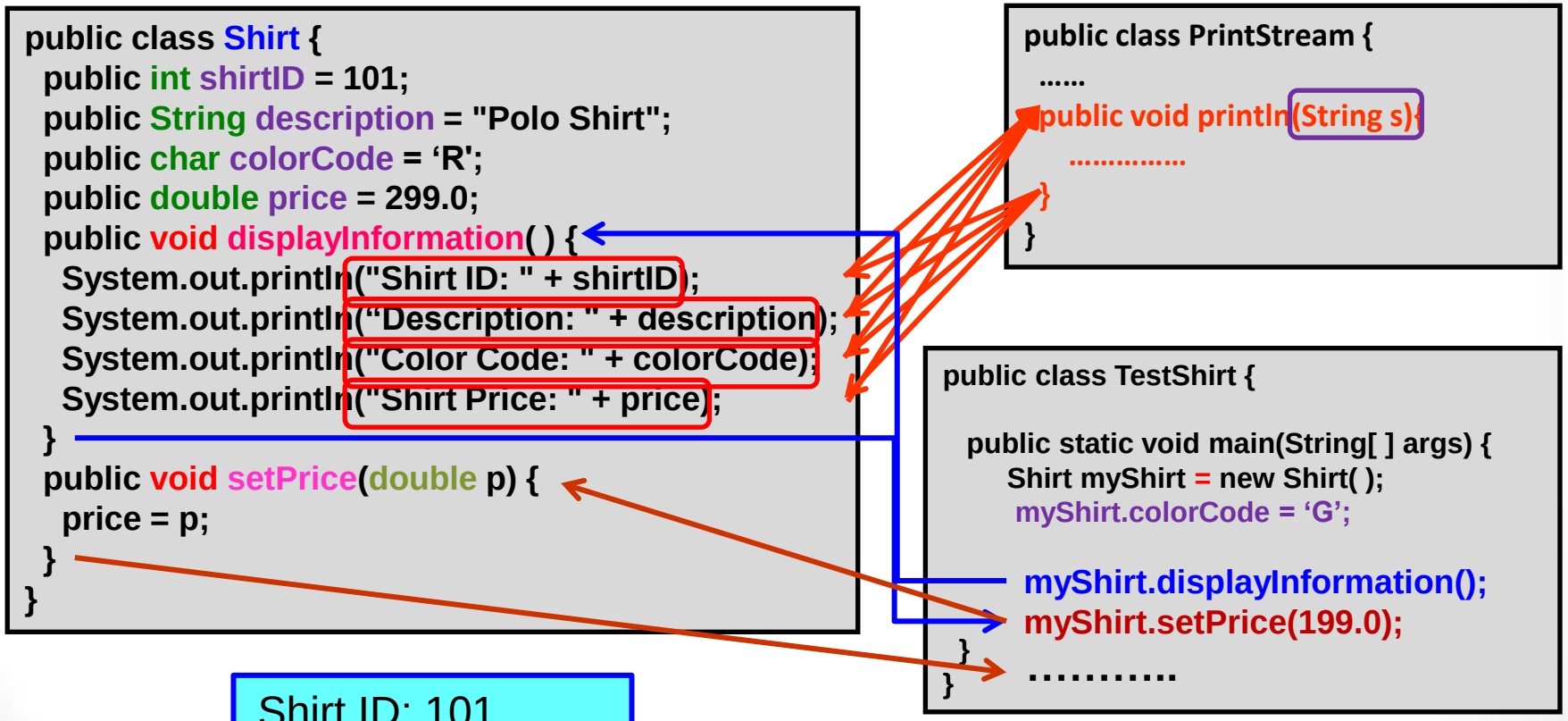
```
void setPrice(double d) //方法簽章
{d *= 0.9;} //方法內容
```

Method

- 方法的呼叫
  - 透過物件呼叫(instance methods)
    - 必須使用正確的方法名稱、參數個數與型態，例如：str.substring()
  - 透過類別呼叫(static methods or class methods)
    - 如：Math.power()
- 呼叫方法時會比對方法簽章的3個部分：
  - 名稱
  - 參數個數
  - 參數資料類型

```
void setPrice(double d) //方法簽章
{d *= 0.9;} //方法內容
```

方法呼叫堆疊 Call Stack



Shirt ID: 101
Description: Polo
Color Code: G
Shirt Price: 299.0

Method

- 參數傳遞(Passing Argument)
  - 參數名稱
    - 參數名稱在 method 中用來指所傳入的變數
    - 參數名稱可與類別中的成員變數名稱相同，此時成員變數將被隱藏(hide)，也就是說，在 method 中所指稱的乃所傳入之變數，此時可用 this 來指定成員變數
  - 參數傳遞方式
    - 傳值 Pass-by-Value
      - 傳入參數為 primitive 型態，**在 method 內對參數的改變將不影響原來method 外變數的值**
    - 傳址 Pass-by-Reference
      - 傳入參數為 reference 型態，無法更動 reference 對象，但可使用物件的 methods 與變數

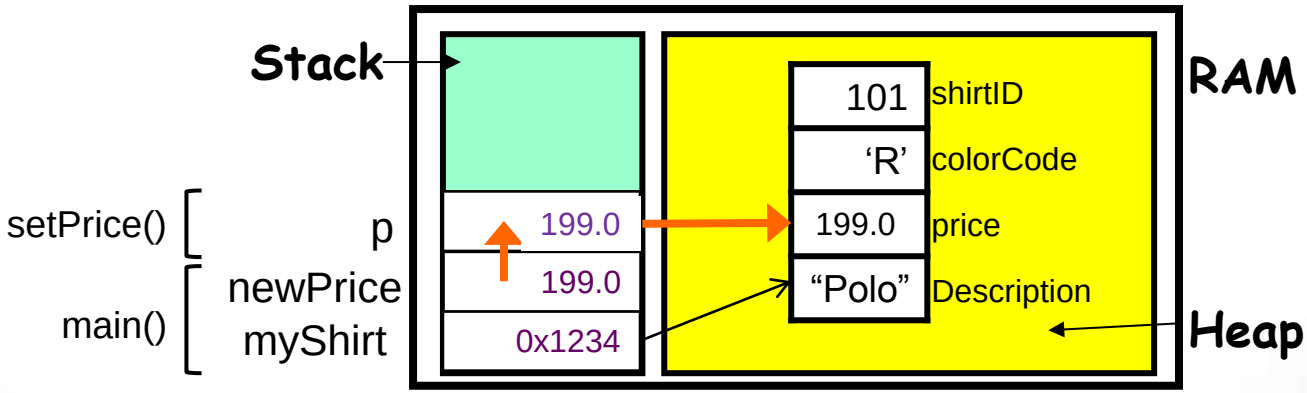
Pass by Value (傳值)

- Java 中參數的指派是傳遞目前 stack 中的內容
  - primitive type - 內容值

```
public class Shirt {
  public int shirtID = 101;
  public char colorCode = 'R';
  public double price = 299.0;
  public String description = "Polo Shirt";
  public void setPrice(double p) {
    price = p;
  }
}
```

```
public class TestShirt {

  public static void main(String[ ] args) {
      Shirt myShirt = new Shirt( );
      double newPrice = 199.0;
      myShirt.setPrice(newPrice);
      ..........
  }
}
```



Method

- Example: PassArgDemo.java

```
static void myMethod1 (int i)
{
    i += 1;
    System.out.println("In MyMethod(): " + i);
}

public static void main(String[] args)
{
    int i = 10;
    // 參數以 primitive 的型態傳入
    myMethod1(i);
    // 不會改變在 main 中的 i 值
    System.out.println("In main(): " + i);
}
```

Pass by Value (傳址)

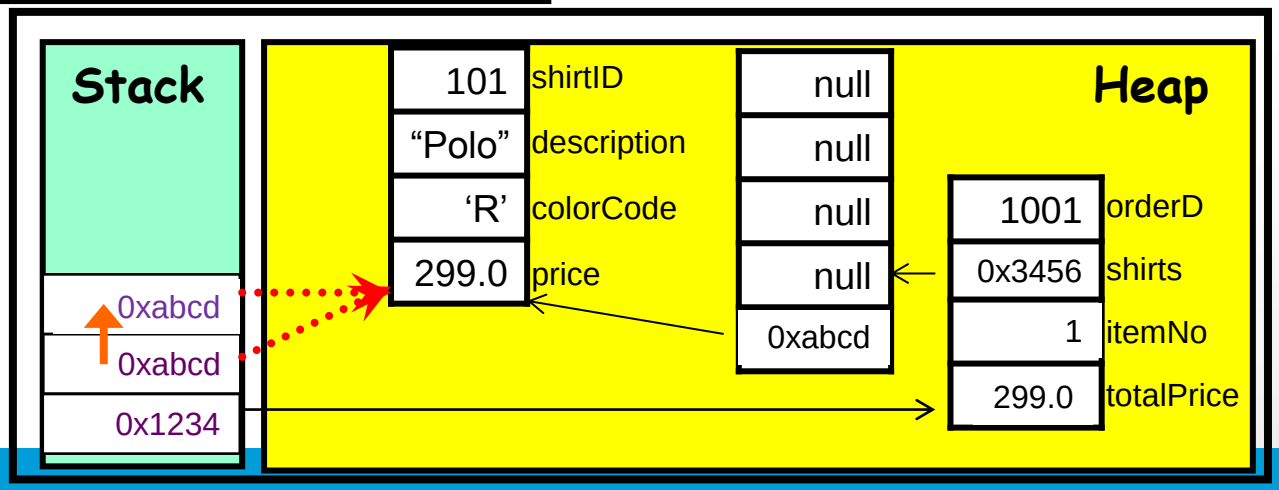
- Java 中參數的指派是傳遞目前 stack 中的內容
  - reference type - 參考值(位址)

```
public class Order {
   public int orderID = 1001;
   public Shirt[ ] shirts = new Shirt[5];
   public int itemNo = 0;
   public double totalPrice = 0.0;
   public void addShirt(Shirt s) {
      shirt[itemNo++] = s;
      totalPrice += s.price;
   }
}
```

```
public class TestOrder {

  public static void main(String[ ] args) {
      Order myOrder = new Order( );
      Shirt myShirt = new Shirt();

      myOrder.addShirt(myShirt);
      ……
  }
}
```



Method

- Example: PassArgDemo.java

```
static void myMethod2 (int[] i)
{
    i[0] = i[0] +1;
    System.out.println("In MyFunc(): " + i[0]);
}

public static void main(String[] args)
{
     int[] i = {10};
     // 參數以 reference 的型態傳入
     myMethod2(i);
     // 會改變在 main 中的 i 值
     System.out.println("In main(): " + i[0]);
}
```

Method

- return
  - 宣告method時可設定回傳型態，在method主體中，必須使用 return 敘述，回傳適當的數值
  - 宣告為 void 的method，不能包含任何return敘述
  - 回傳的數值的資料型態，必須要符合method宣告設定的資料型態一致

Method

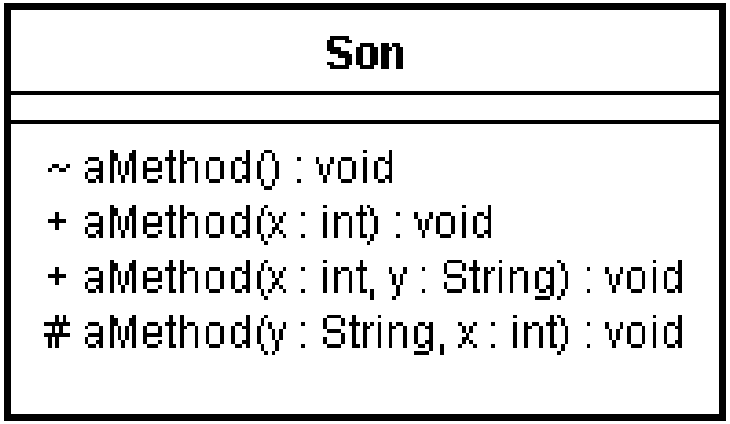
- Example: PassArgDemo.java

```
static int myMethod3 (int i)
{
    i *= 10;
    return i;
}

public static void main(String[] args)
{
    int i = 10;
    int j = myMethod3(i);
    System.out.println("The return value: " + j);
}
```

方法多載 (method overloading)

◆**同名方法名稱**根據其**不同的參數型別**以對應執行到不同的實作(內容)。




```
01  public class Son {
02      void aMethod() { }
03      public void aMethod(int x) { }
04      public void aMethod(int x, String y) { }
05      protected void aMethod(String y, int x) { }
06  }
```

方法多載(method overloading)

```
public class Calculator {
   public int sum(int numberOne, int numberTwo){
     System.out.println("Method One");
     return numberOne + numberTwo;
   }

  public float sum(float numberOne, float numberTwo) {
     System.out.println("Method Two");
     return numberOne + numberTwo;
  }

  public float sum(int numberOne, float numberTwo) {
    System.out.println("Method Three");
    return numberOne + numberTwo;
  }
}
```

```
public class CalculatorTest {

  public static void main(String [] args) {

    Calculator myCalculator = new Calculator();

    int totalOne = myCalculator.sum(2,3);
    System.out.println(totalOne);

    float totalTwo = myCalculator.sum(15.99F, 12.85F);
    System.out.println(totalTwo);

    float totalThree = myCalculator.sum(2, 12.85F);
    System.out.println(totalThree);
  }
}
```

Q & A