Java 程式設計進階
建構子(Constructor)與封裝(Encapsulation)

鄭安翔
ansel_cheng@hotmail.com

課程大綱

1) 建構子
   - 物件屬性欄位初始化
   - 建構子
   - 預設建構子
   - 建構子多載
2) 封裝

Java 建構子

- Java 建構子
  - 定義類別時，可以使用建構式定義物件建立的初始化流程
  - 物件建構時，即設定適當的初始值
    - 依不同參數建構不同內容的物件
    - 屬性值不需逐一設定
    - 避免遺漏

```
public class Shirt {
    public int shirtID;
    public char colorCode= 'G';
    public double price = 299.0;
    public String description;
    public void displayInformation( ) {….}
}
```

```
public class TestShirt {

    public static void main(String[ ] args) {
        Shirt myShirt = new Shirt();
        myShirt.shirtID = 101;
        myShirt.colorCode = 'R';
        myShirt.price = 199.0;
        myShirt.description = "Polo Shirt";

        myShirt.displayInformation( );

    }
}
```

Java 建構子

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

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

建構子 constructor 語法

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

Accessibility

Same as class_name

Arguments list

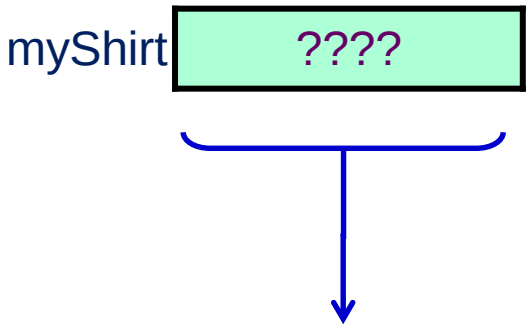
- 與類別名稱一樣
- 沒有回傳型態
- 預設建構子
- 可以多載(Overloading)

物件建構流程 – 宣告

```
public class Shirt {
    private int shirtID = 101;
    private char colorCode = 'R';
    private double price = 299.0;
    private String description = "Polo Shirt";

    public Shirt(char c, double p, String d){
        colorCode = c;
        price = p;
        description = d;
    }

}
```

Shirt myShirt;

myShirt = new Shirt('G', 199.0 ,
"T-Shirt");

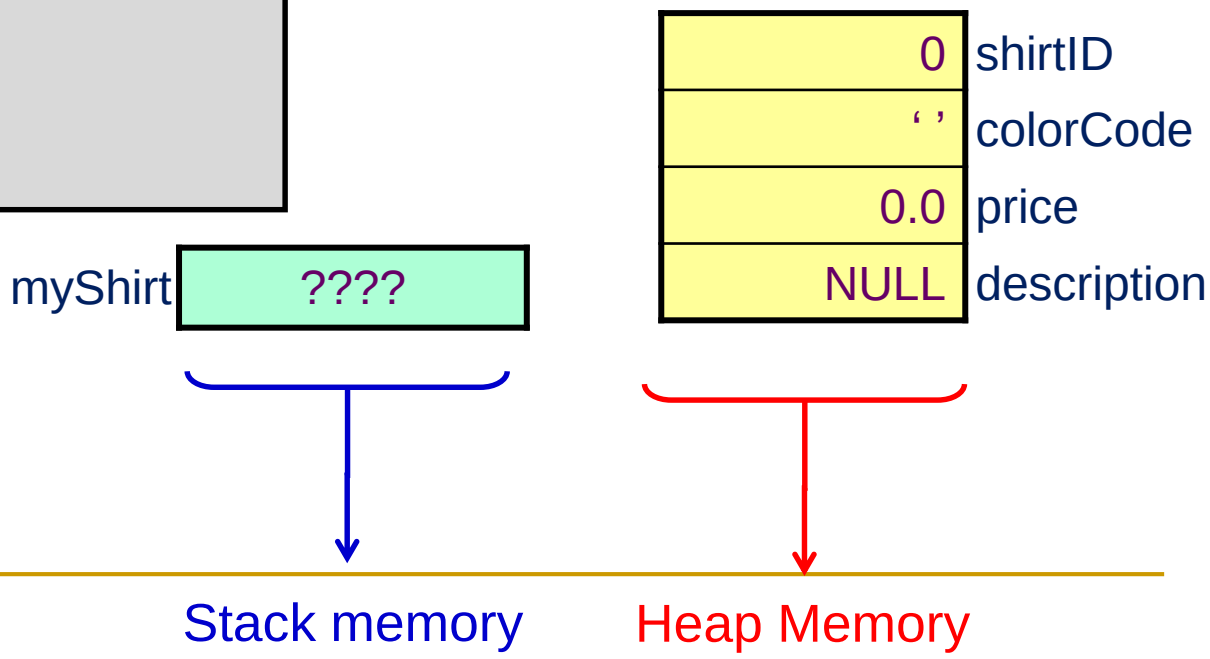


物件建構流程 – 實體化

記憶體配置

```
public class Shirt {
    private int shirtID = 101;
    private char colorCode = 'R';
    private double price = 299.0;
    private String description = "Polo Shirt";

    public Shirt(char c, double p, String d){
        colorCode = c;
        price = p;
        description = d;
    }

}
```

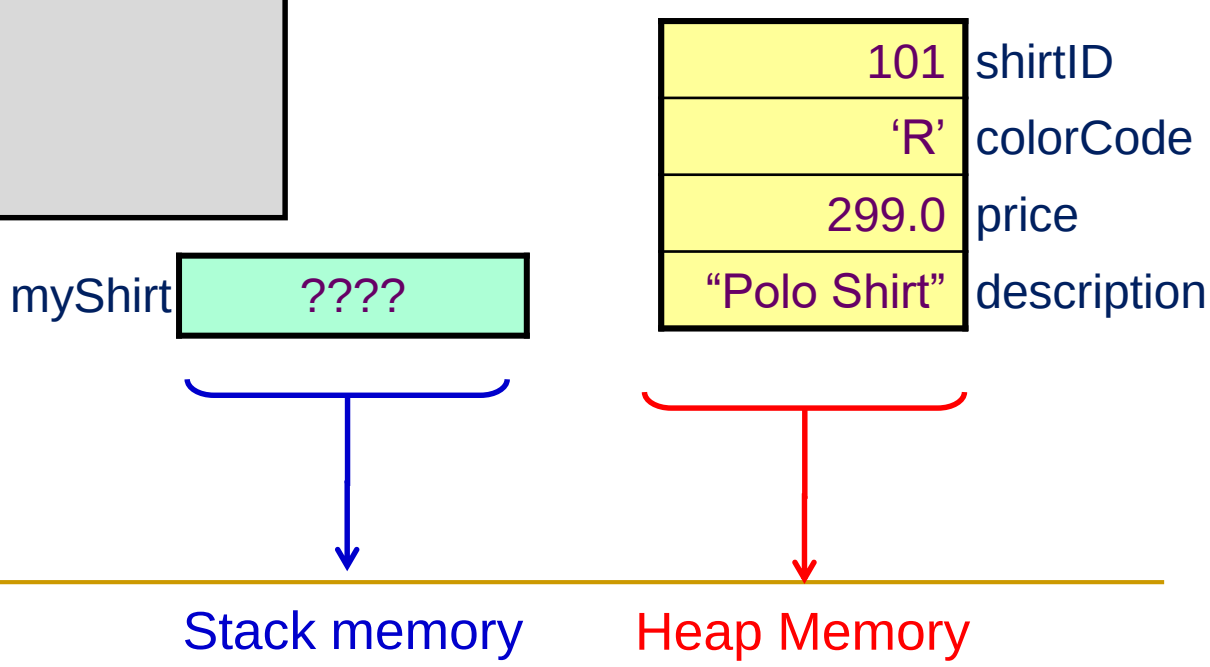
```
Shirt myShirt;
myShirt = new Shirt ('G', 199.0 ,
                     "T-Shirt" );
```



物件建構流程 –初始化
初始值賦值

```
public class Shirt {
    private int shirtID = 101;
    private char colorCode = 'R';
    private double price = 299.0;
    private String description = "Polo Shirt";

    public Shirt(char c, double p, String d){
        colorCode = c;
        price = p;
        description = d;
    }

}
```

```
Shirt myShirt;
myShirt = new Shirt ('G', 199.0 ,
                     "T-Shirt" );
```

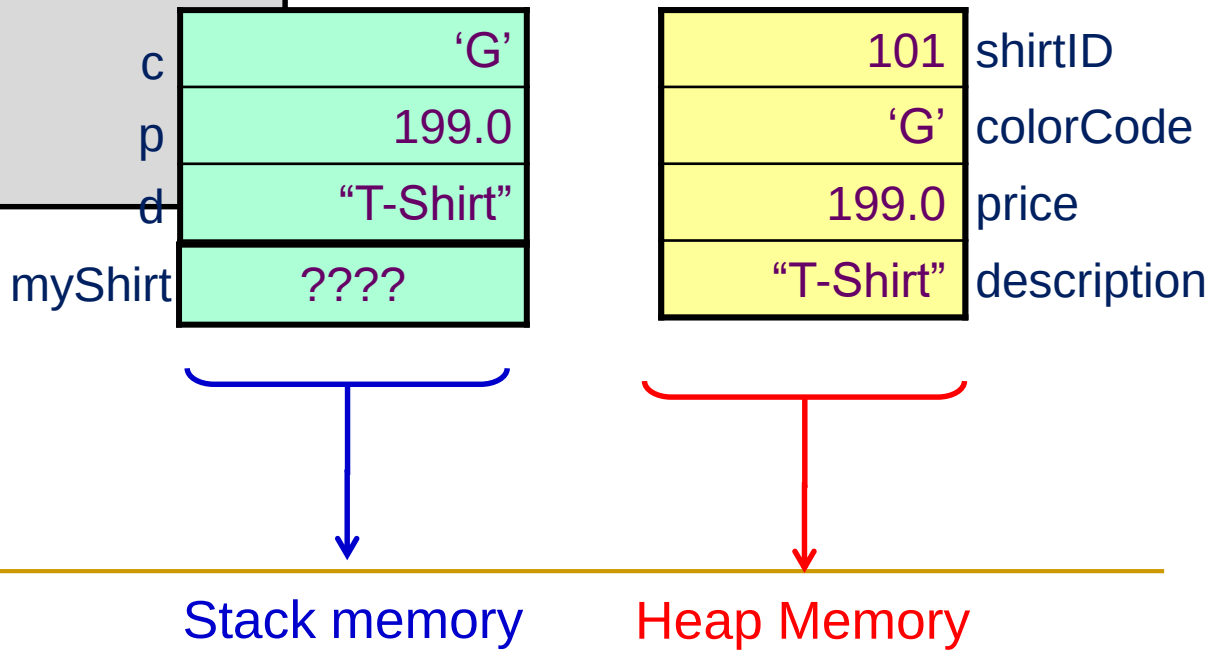


物件建構流程 –初始化

執行建構式

```
public class Shirt {
    private int shirtID = 101;
    private char colorCode = 'R';
    private double price = 299.0;
    private String description = "Polo Shirt";

    public Shirt(char c, double p, String d){
        colorCode = c;
        price = p;
        description = d;
    }

}
```

```
Shirt myShirt;
myShirt = new Shirt ('G', 199.0 ,
                     "T-Shirt" );
```

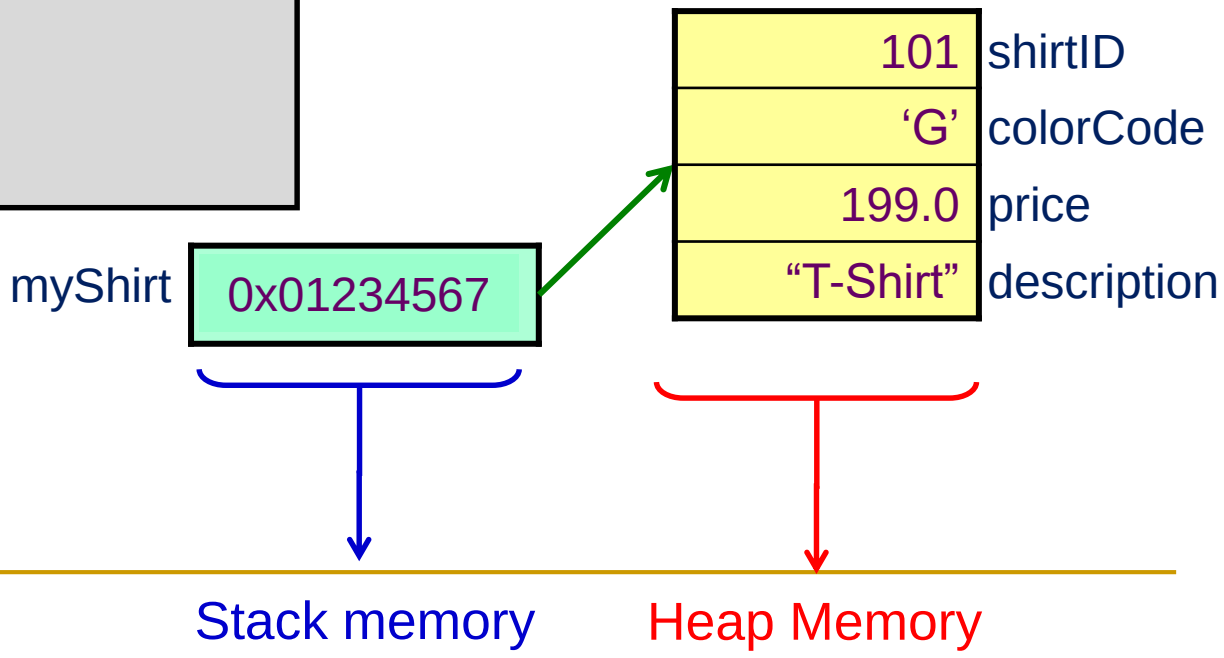


物件建構流程 –儲存物件參考

```
public class Shirt {
    private int shirtID = 101;
    private char colorCode = 'R';
    private double price = 299.0;
    private String description = "Polo Shirt";

    public Shirt(char c, double p, String d){
        colorCode = c;
        price = p;
        description = d;
    }

}
```

Shirt myShirt;

myShirt = new Shirt('G', 199.0 , "T-Shirt");



預設建構子 Default Constructor

- 物件裡面一定要有建構子，所以在撰寫class時必須定義該物件的建構子
- 程式中若沒有定義建構子，在編譯時期會自動加入，所加入的就稱之為預設構構子；
  - 預設建構子沒有參數列(no arguments)；
  - 除了初始物件變數或繼承時super()的定義外，預設建構子沒有其他的程式敘述(no body statement)。
  - 自行建立建構後子即無預設建構子

```
01  public class Shirt {
02      private int shirtID = 101;
03      private char colorCode = 'R';
04      private double price = 299.0;
05      private String description = "Polo Shirt";
06       public Shirt(){ }
07
08  }
```

```
01  public class TestShirt {
02      public static void main(String[] args) {
03          Shirt s = new Shirt();
04      }
05  }
```

javac Shirt.java

建構子多載 Constructors overloading

- 提供多組建構子為物件設定初值
  - 傳入參數數量或型態不同

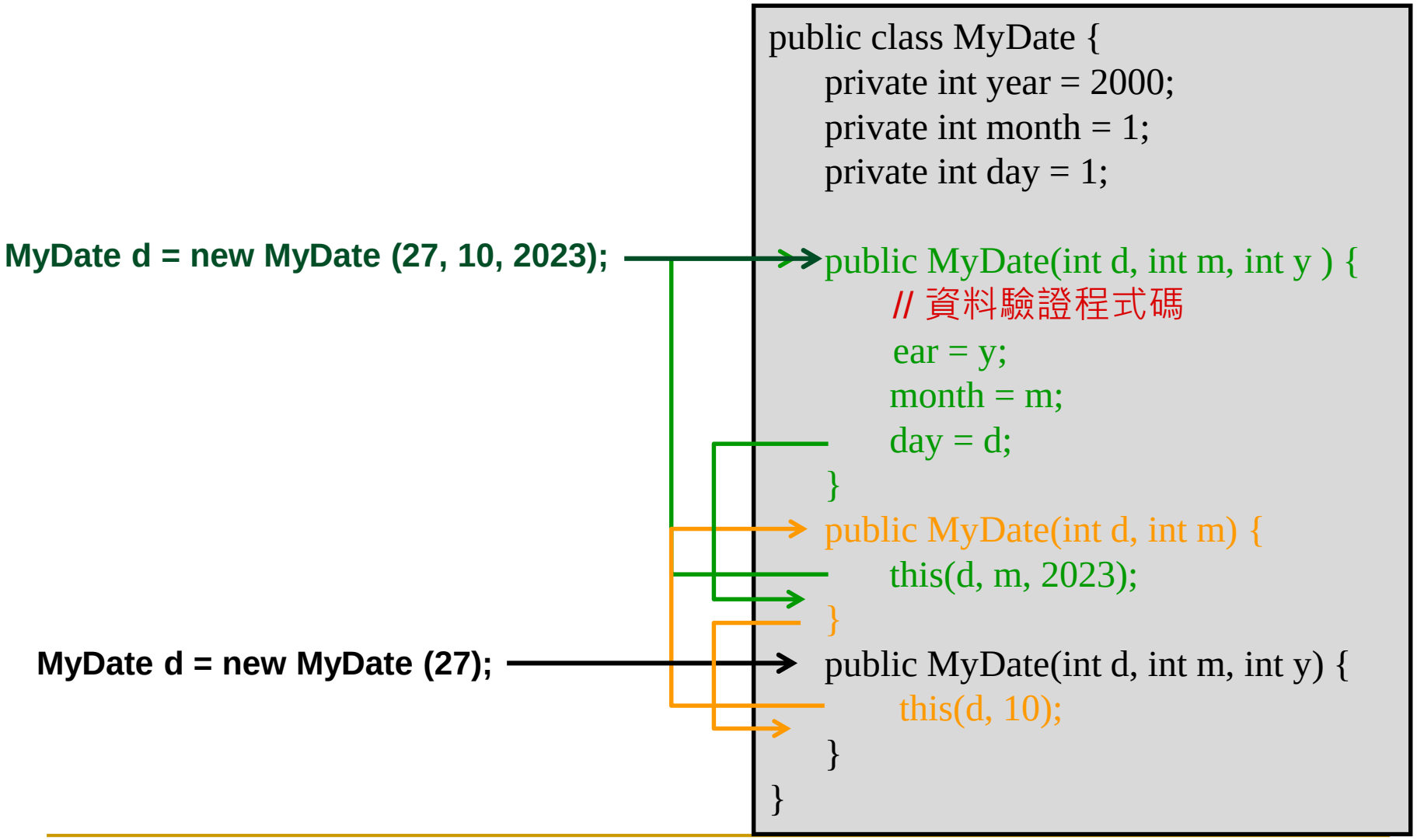
```
01  public class Shirt {
02      private int shirtID = 101;
03      private char colorCode = 'R';
04      private double price = 299.0;
05      private String description = "Polo Shirt";
06
07      public Shirt(int id) {
08          shirtID = id;
09      }
10      public Shirt(char color, double newPrice) {
11          colorCode = color;
12          price = newPrice;
13      }
14      public Shirt(char color, double newPrice,
15                   String desc) {
16          colorCode = color;
17          price = newPrice;
18          description = desc;
19      }
20  }
```

```
01  public class TestShirt {
02      public static void main(String[] args) {
03
04          Shirt s1 = new Shirt();
05
06          Shirt s2 = new Shirt(101);
07
08          Shirt s3 = new Shirt('G', 600.0);
09
10          Shirt s4 = new Shirt('Y', 199.0,
11                               "T-Shirt");
12
13      }
14  }
```

建構子串聯 Chaining Constructors

- 建構子串聯呼叫 Chaining Constructors
  - 在建構子中呼叫其他建構子
    - 減少重複的參數檢查程式碼
  - this(參數列);
    - this 關鍵字：執行時期自動產生，代表本身物件的參考 (Reference)
  - 必須寫在建構子中第一行

建構子串聯 Chaining Constructors

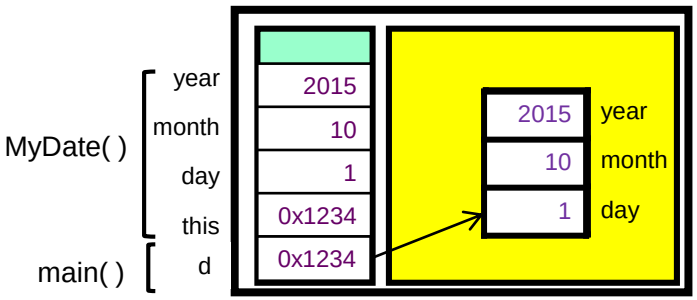


存取被遮蔽的屬性 - this

- **this** 關鍵字
  - 代表本身物件的參考 (Reference)
  - 建構子或方法中，存取被遮蔽的物件屬性
    - this.屬性

```
public class MyDate {
    public int year = 2000;
    public int month = 1;
    public int day = 1;

    public MyDate(  int day, int month, int year    ) {
        this.day   = day;
        this.month = month;
        this.year  = year;
    }
}
```

MyDate d = new MyDate(1, 10, 2015);

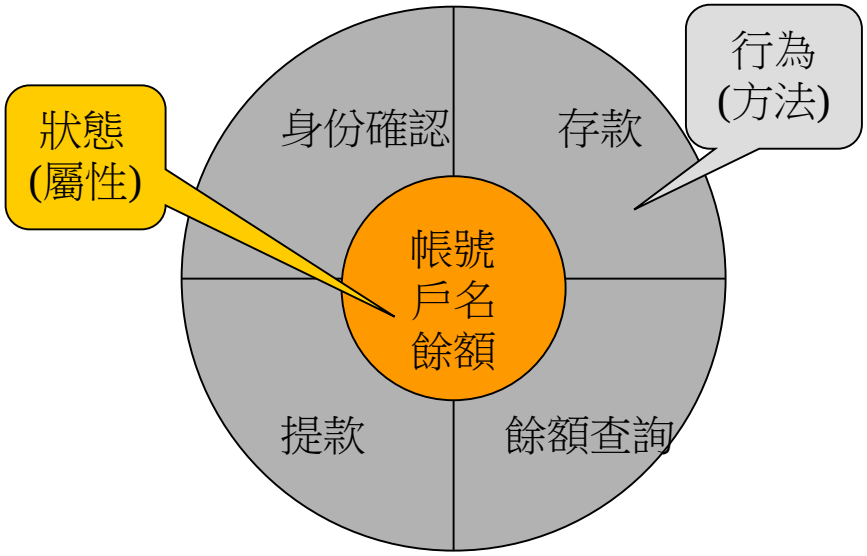


課程大綱

1) 建構子
2) **封裝**
   - **封裝的概念**
   - **Java語言實作封裝**
   - **存取權限修飾字**
   - **封裝與建構子**

封裝 Encapsulation

- 封裝 encapsulation
  - 保護類別中的資料,不讓資料被誤用或破壞
  - 隱藏實作的細節,增加應用程式可維護性



封裝的方法

- 封裝的方法
  1. 更改屬性為private
  2. 提供存取屬性的方法
     - Accessor (getter & setter)
     - Setter提供保護資料的邏輯
  3. 存取此類別之資料,需使用類別所提供的方法來存取

Java 語言實作封裝

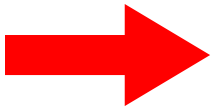
```
public class MyDate {
   public int  day;
   public int month;
   public int year;
}
```

```
public class TestMyDate {
  public static void main(String args[]) {
     MyDate d = new MyDate();

     d.day = 30;
     d.month = 2;
     d.year = 2003;

     System.out.println(d.day +"/"+ d.month +
                                "/"+    d.year);
  }
}
```

Invalid date

Java 語言實作封裝

```
public class MyDate {
   public int  day;
   public int month;
   public int year;
}
```



```
public class MyDate {
  private int  day;
  private int month;
  private int year;

  public void setDate(int d, int m, int y) {
        ……
        ……
  }

  public String getDate() {
        return day +"/"+ month +"/"+ year;
  }
}
```

setter method
setXXX( )

getter method
getXXX( )

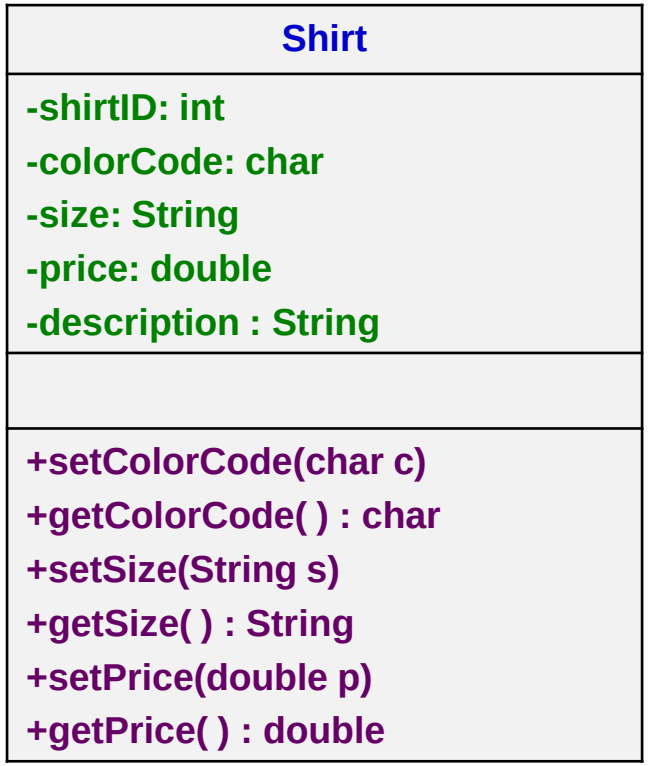
Java 語言實作封裝

```
public class MyDate {
  private int  day;
  private int month;
  private int year;

  public void setDate(int d, int m, int y) {
    ……..
    ……
  }

  public String getDate() {
    return day +"/"+ month +"/"+ year;
  }
}
```

```
public class TestMyDate {
  public static void main(String args[]) {
    MyDate d = new MyDate();

    d.day = 30;
    d.month = 2;
    d.year = 2003;

    System.out.println(d.day +"/"+ d.month +
                       "/"+    d.year);

    d.setDate(28,2 2003);

    System.out.println("Date: " + d.getDate());
  }
}
```

compile error!

封裝 Shirt 類別

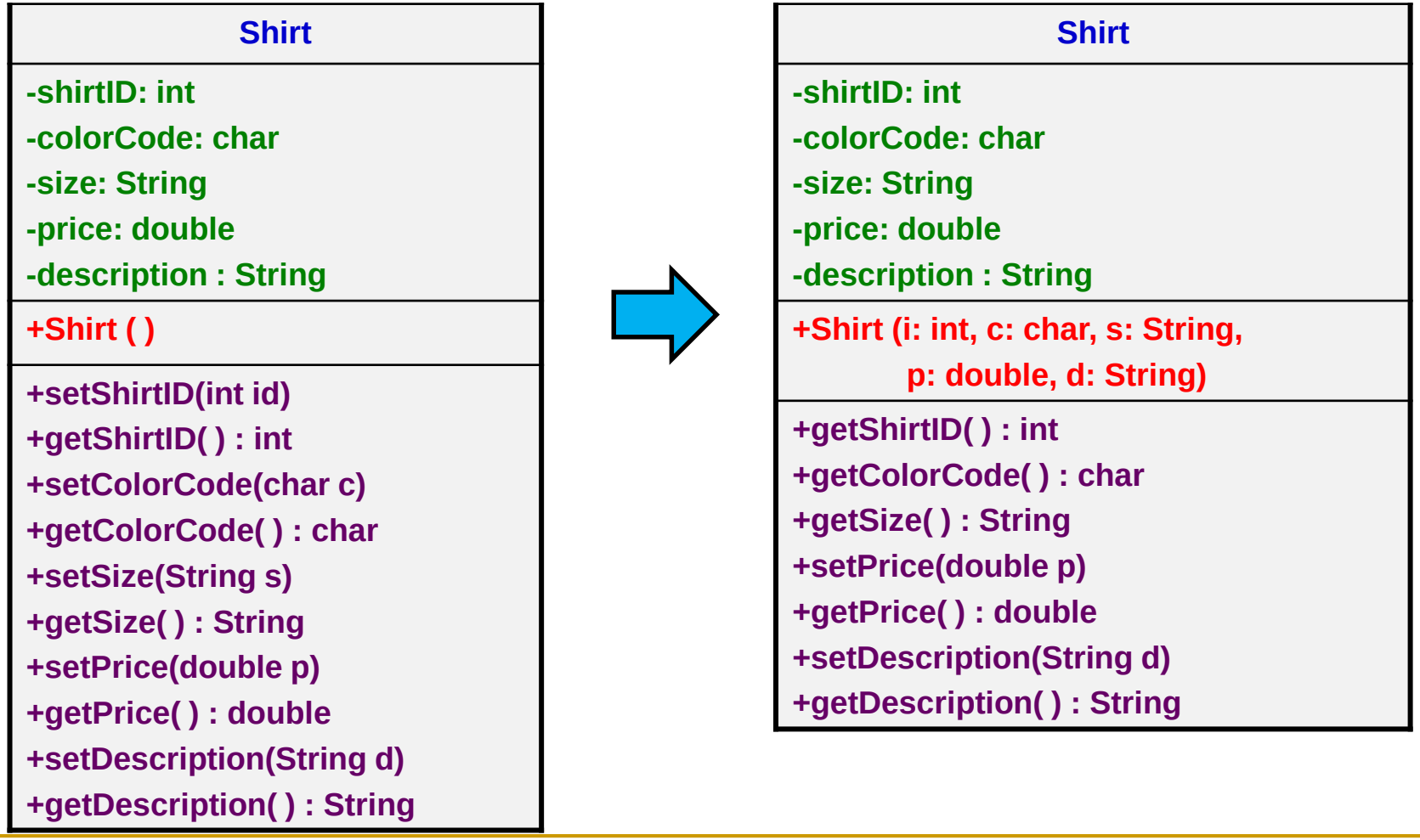



```
01  public class Shirt {
02      private int shirtID = 0;
03      private char colorCode = 'G';
04      private String size = "XL" ;
05      private double price = 299.00;
06
07      public void setColorCode(char c) {
08          if(c=='R' || c=='G' || c=='Y')
09              colorCode = c;
10      }
11      public char getColorCode ( ) {
12          return colorCode;
13      }
14      public void setSize(String s) {
15          if(s.equals("S") || s.equals("M") ||
16            s.equals("L") || s.equals("XL") )
17              size = s;
18      }
19      public String getSize( ) {
20          return size;
21      }
22      public void setPrice(double p) {
23          if(p>=0.0)
24              price = p;
25      }
26      public double getPrice( ) {
27          return price;
28      }
29
30  }
```

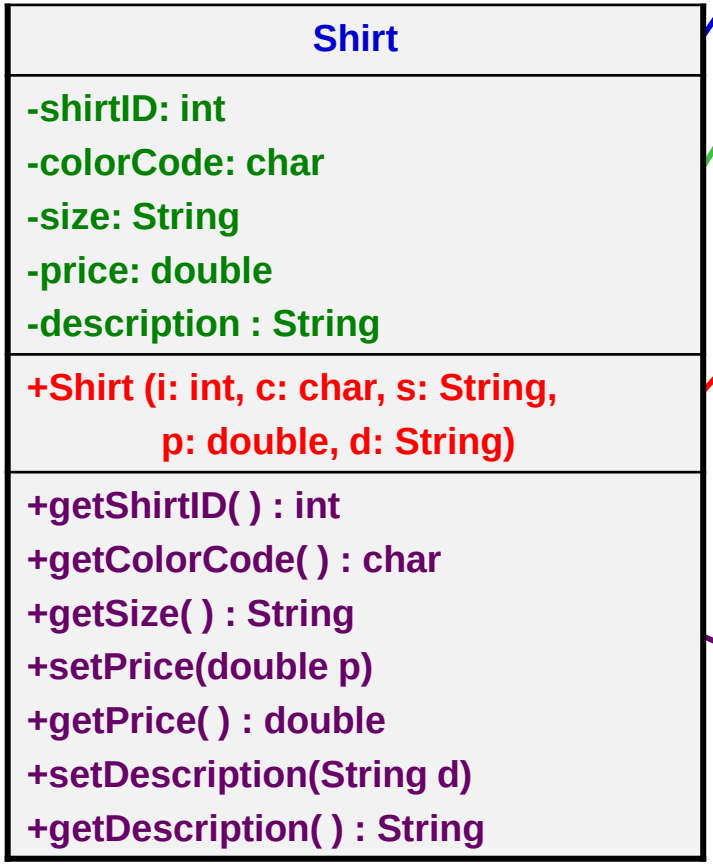
封裝的設計準則 Best Practice

- 設計準則 Best Practice
  - As Immutable as possible
  - 盡可能隱藏屬性實作細節並使其不可變更
  - 只公開必需的Setter方法
  - 以建構式設定不可變更屬性的初始值

封裝實作


Shirt
-shirtID: int
-colorCode: char
-size: String
-price: double
-description : String
+Shirt ( )
+setShirtID(int id)
+getShirtID( ) : int
+setColorCode(char c)
+getColorCode( ) : char
+setSize(String s)
+getSize( ) : String
+setPrice(double p)
+getPrice( ) : double
+setDescription(String d)
+getDescription( ) : String
Shirt
-shirtID: int
-colorCode: char
-size: String
-price: double
-description : String
+Shirt (i: int, c: char, s: String,
p: double, d: String)
+getShirtID( ) : int
+getColorCode( ) : char
+getSize( ) : String
+setPrice(double p)
+getPrice( ) : double
+setDescription(String d)
+getDescription( ) : String

封裝實作




```
01  public class Shirt {
02      private int shirtID = 0;
03      private char colorCode = 'G';
04      private String size = "XL" ;
05      private double price = 299.00;
06      private String descriptiom = "Polo Shirt";
07
08      public Shirt(int i, char c, String s,
09                   double d, String d){
10          shirtID = I;     colorCode = c;
11          size = s;      price = p;
12          description = d;
13      }
14
15      public int getShirtID ( ) {
16          return shirtID;
17      }
18      public char getColorCode ( ) {
19          return colorCode;
20      }
21      .......
22      public void setPrice(double p) {
23          if(p>=0.0)
24              price = p;
25      }
26      public double getPrice( ) {
27          return price;
28      }
29      .......
30  }
```

物件屬性
(欄位)

建構子

物件方法
(操作)