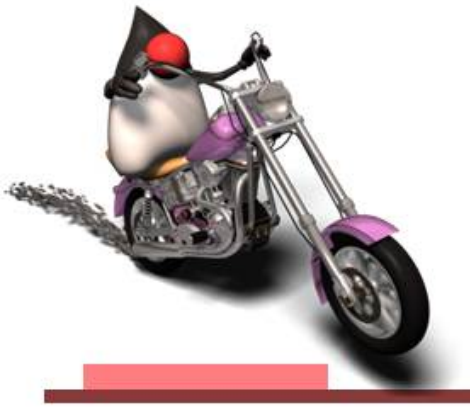


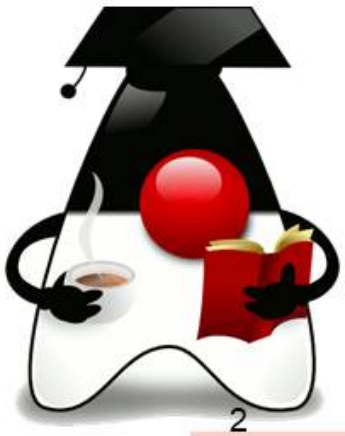
J01-11

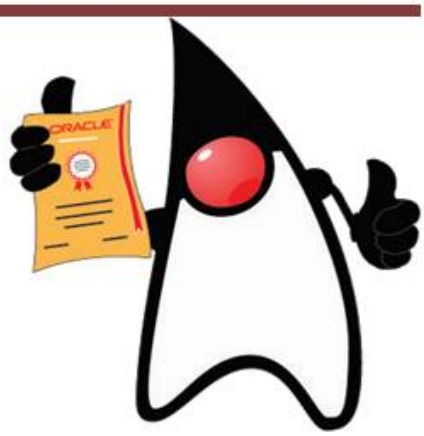
使用 封裝 (Encapsulation) 和 建構子 (Constructors)

曾瑞君 (Jim_Tzeng)

學習目標

1. 封裝 (**Encapsulation**) 的觀念與做法
2. 使用建構子 (**Constructors**)





1/2

封裝 (Encapsulation) 的觀念與做法

封裝的目的

- 封裝是物件導向(Object Oriented)程式設計的重要一環。
- 封裝可以藉由將物件的屬性/欄位設定為private，來達到隱藏的效果，使別的物件無法直接存取該物件屬性。此時可以搭配
  - getter、setter方法
  - 在setter方法裡，可以檢查設定值是否合理。

為什麼要封裝 field?

- public field 可能遇到這種問題：

```
public class Person0 {
    public int age;
}
```

```
public class PersonTest {
    public static void testPerson0() {
        Person0 p = new Person0();
        p.age = 200;
        System.out.println(p.age);
    }
    public static void main(String[] args) {
        testPerson0();
    }
}
```

封裝 field後

```
public class Person1 {
    private int age;
    public void setAge(int age) {
        if (age>1 && age < 120)
            this.age = age;
    }
    public int getAge() {
        return this.age;
    }
}
```

Setter

Getter

```
public class PersonTest {
    public static void testPerson1() {
        Person1 p = new Person1();
        // p.age = 200; //compile error
        p.setAge(200);
        System.out.println(p.getAge());
        // System.out.println(p.age); //compile error
    }
    public static void main(String[] args) {
        testPerson1();
    }
}
```

Method 需要封裝嗎?

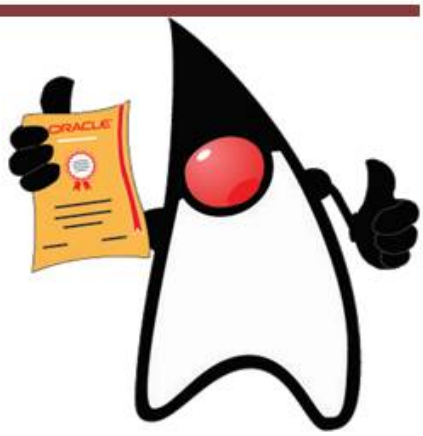
```
public class Elevator {
        //... < code omitted > ...
? private void setFloor(int desiredFloor ) {
        while ( currentFloor != desiredFloor ){
            if (currentFloor < desiredFloor) {
                goUp();
            } else {
                goDown();
            }
        }
    }
}
```

Method 封裝後

```
public class Elevator {
        //... < code omitted > ...
    private void setFloor(int desiredFloor ) {
        while ( currentFloor != desiredFloor ){
            if (currentFloor < desiredFloor) {
                goUp();
            } else {
                goDown();
            }
        }
    }
    public void service (int desiredFloor, Boolean x) {
        //... < contains code to add requested floor to a queue > ...
    }
}
```

Method 封裝做法

- 定義公開方法，未來考慮進一步和 interface(介面) 整合。
- 物件的商業邏輯實作細節，盡可能private；供 interface 上的public method內部呼叫。
- 未來程式擴充，僅需修改private method的商業邏輯實作內容，毋須異動interface；使用者 (caller) 也就不需一併更動。
- 【封裝】有助於【多型】的實現!



2/2

使用建構子 (Constructors)

這樣做OK...但...

- 物件內容主是要data，設定value的作法：

```
public class ShirtTest {
    public static void main (String args[]) {
        Shirt s = new Shirt();
        // set values
        s.setColorCode('R');
        s.setDescription("Outdoors Function");
        s.setPrice(45.12);
        s.setSize(20);
        s.show();
    }
}
```

使用Constructor的時機

- 看到2個問題：
  - 初始化 object 的 field 可使用setter method；但若 field 多時，有點麻煩。
  - 那些field對完成物件初始化是必要的?依賴setter逐field設定，可能有遺漏的問題。

何謂Constructor

- 和宣告method的方式很相似：
  - 名字必須和class name一樣
  - 通常使用來初始化object的field
  - 可以是 overloaded
  - 沒有return type，也不是 void
- 語法

```
[modifiers] class ClassName {
    [modifiers] ClassName ([arguments]){
        code_block
    }
}
```

```
public class Shirt {

    private char colorCode;
    private int size;
    private double price;
    private String description;

    public void setColorCode(char colorCode) {this.colorCode = colorCode;}
    public void setSize(int size) {this.size = size;}
    public void setPrice(double price) {this.price = price;}
    public void setDescription(String description) {this.description = description;}

    public Shirt( int size, double price ) {
        this.setSize(size);
        this.setPrice(price);
    }

    public void showToCustomer() {
        System.out.println(this.price + ", " + this.size);
    }

}
```

Edited by Ruei-Jiun Tzeng

使用Constructor建構新物件

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

    //Shirt s = new Shirt();

    Shirt1 s1 = new Shirt1(20, 45.12) ;
    s1.show();

    s1.setColorCode('R');
    s1.setDescription("Outdoors Function");
    //s.setPrice(50);
    //s.setSize(19);
}
```

How about this?

Call Constructor

建構子使用原則 1

1. 若程式開發者未在類別中建立建構子，則Java將自動提供「預設建構子(Default Constructor)」，該建構子不帶任何參數，而且我們看不到。所以先前建立物件時，雖然類別裡沒有建構子的設計，還是可以直接使用：

   ```
   Shirt0 s0 = new Shirt0();
   ```

2. 反之，若開發者已建立類別的新建構子，則Java將不再提供未帶參數的預設建構子。所以以下呼叫方式就會編譯失敗：

   ```
   Shirt1 s1 = new Shirt1();
   ```

建構子使用原則 2

3. 若已經建立其他建構子，但仍需要使用未帶參數的建構子，此時就必須自己建立，稱為「無參數建構子 (No-args Constructor)」。

4. 建構子允許多個，因此可以藉由多載(overloading)建立其他參數不同的建構子。建構子之間，可以使用「this(args)」互相呼叫。因為建構子串聯呼叫，像鍊子(chain)般連結，可以稱為「Chaining Constructors」。

No-args Constructor

```
public class Shirt2 {

    public Shirt2() {}

    public Shirt2(int size, double price) {
        this.setSize(size);
        this.setPrice(price);
    }

    public Shirt2( int size, double price, char colorCode ) {

        this(size, price);

        //Shirt2(size, price);          //Error!! Java will try to find method "Shirt2()"

        this.setColorCode(colorCode);
    }
}
```

No-args Constructor

Chaining Constructors

Edited by Ruei-Jiun Tzeng

END ~~

Thank you!!