程式語言 (Programming Language)

Lec. 04 物件導向觀念

楊　吳　泉

義守大學資訊工程學系

web: http://elearning.isu.edu.tw

http://audtm.net/~wcyang

mail: wcyang@isu.edu.tw

綱要 (Outline)

- 類別與物件 (Class & Object)
- 欄位與方法 (Field & Method)
- 繼承 (Inheritance)
- 多載與覆寫 (Overload & Override)
- 例外處理 (Exceptions)
- 介面與實作 (Interface & Implement)

Class & Object (1/2)

- 定義
  - ✓ Class: 定義物件在執行時期之狀態(state)與行為(behavior)的規格(template)
  - ✓ Object: 類別的實體(instance)
  - ✓ The concepts of *class/object* provide a mechanism for capsulation

- 類別格式

  [修飾字] ***class*** [類別名稱] { [類別定義] }

  常見之類別修飾字: public, abstract, final

  Ex: public class Ex3_1 {

  . . .

  }

Class & Object (2/2)

➢ 類別之整理與引用

✓ Java **API** (Application Program Interface):可視為Java之函式庫,包含許多有用之類別,協助程式設計師以Java撰寫各類程式.

→ API中將各相關類別依照使用特性, 分門別類置於不同之目錄, 稱為Package.

→ 欲引用某一類別時須於類別定義之前, 以**import**指令引用.
(package java.lang是唯一會自動引入程式之類別)

→ 也可以package指令自行定義類別於指定的**package**中, 需注意package敘述需置於import敘述之前.

Field & Method (1/4)

- 基本之類別成員

類別欄位　**類別方法** → 使用 **static** 定義之欄位與方法

實體欄位　**實體方法** →未使用static定義之欄位與方法

↓　↓

資料存於欄位　程式碼組成方法

- 基本之類別成員
  - ✓ 差別
    - 屬於**整體共同**之特性或操作為類別欄位或方法
    - 屬於**物件個別**之特性或操作為實體欄位或方法
  - ✓ 使用方式
    - 類別欄位或方法以類別名稱呼叫
    - 實體欄位或方法以實體名稱呼叫

Field & Method (2/4)

- 宣告範例

Circle.java | TestCircle.java

```
1  public class Circle {
2      // 類別欄位與方法
3      public static final double PI=3.14159;
4      public static double radiansToDegrees(double rads)  {
5              return rads * 180 / PI;
6      }
7      //實體欄位與方法
8      public double r;
9      public double area() {
10         return PI * r * r;
11     }
12     public double circumference() {
13         return 2 * PI * r;
14     }
15 }
```

Field & Method (3/4)

➢ 使用範例

```
1  public class TestCircle {
2      public static void main(String[] args) {
3          Circle c=new Circle();
4          c.r=2.0;
5          Circle d=new Circle();
6          d.r=c.r * 2;
7          //靜態方法用class名稱呼叫
8          double deg=Circle.radiansToDegrees(Circle.PI/6);
9          char dc = 0xb0; //表示度的符號
10         System.out.println("PI/6 = "+deg+dc);
11         //一般方法用實體名稱呼叫
12         double cArea=c.area();
13         System.out.println("半徑2.0之圓面積 = "+cArea);
14     }
15 }
```

```
PI/6 = 30.0°
半徑2.0之圓面積 = 12.56636
```

Circle.java | TestCircle.java

```
1  public class Circle {
2      // 類別欄位與方法
3      public static final double PI=3.14159;
4      public static double radiansToDegrees(double rads)  {
5              return rads * 180 / PI;
6      }
7      //實體欄位與方法
8      public double r;
9      public double area() {
10         return PI * r * r;
11     }
12     public double circumference() {
13         return 2 * PI * r;
14     }
15 }
```

Field & Method (4/4)

- 特別之方法---建構子(Constructor)

Constructor: 建立物件時(在記憶體保留區域來儲存物件之相關資料及方法)所執行之方法,主要係用以建立物件之初始狀態.

→ Constructor 名稱與類別名稱相同.

→ Constructor 無須返回值.

→ Constructor 透過new來啟動.

Ex:
```
public Circle(double r) {
    this.r=r;
}
```

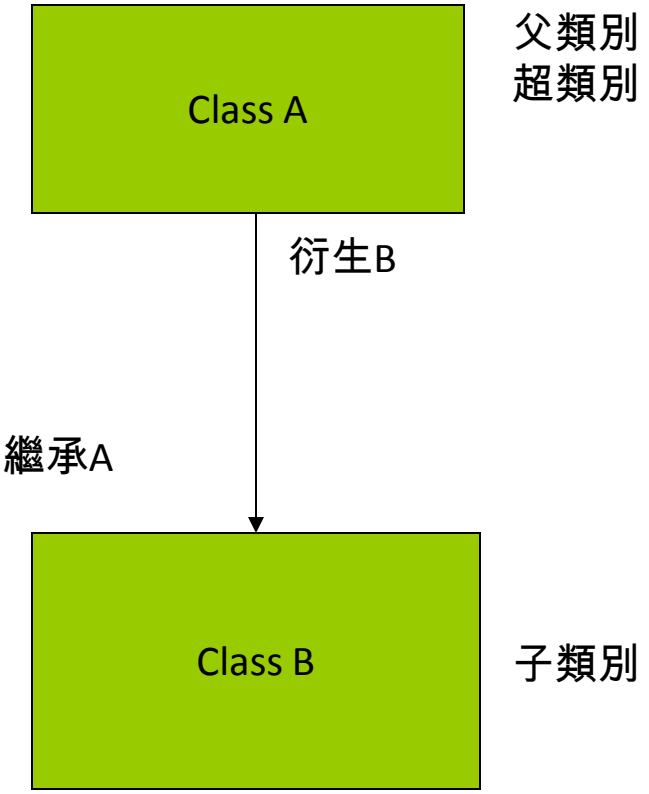
Ex:
```
public Circle() {}
```

Ex:
```
Circle c2=new Circle(2.5);
```

```
1  public class Circle {
2      //建構子
3      public Circle() { }
4      public Circle(double r) {
5          this.r=r;
6      }
7      // 類別欄位與方法
8      public static final double PI=3.14159;
9      public static double radiansToDegrees(double rads)  {
10             return rads * 180 / PI;
11     }
12     //實體欄位與方法
13     public double r;
14     public double area() {
15         return PI * r * r;
16     }
17     public double circumference() {
18         return 2 * PI * r;
19     }
20 }
```

Inheritance (1/2)

- 衍生 (或稱繼承, Inheritance)
  - ✓ 如右圖類別B繼承類別A
    - B可使用所有A之欄位與方法
    - B可增加新的欄位與方法
    - B可改寫A之欄位與方法 (override)



Inheritance (2/2)

- 關鍵字this:表示目前所在之類別.
- 關鍵字super:表示目前所在類別之父類別.

Test.java

```
1 public class Test {
2   protected int var = 10;
3   public static void main(String arg[]) {
4       TestChild ch=new TestChild();
5       ch.show();
6   }
7 }
```

TestChild.java

```
1 class TestChild extends Test {
2   private int var;
3   TestChild() {
4       var = 20;
5   }
6   public void show() {
7       int var = 30;
8       System.out.println(var);
9       System.out.println(this.var);
10      System.out.println(super.var);
11  }
12 }
```

```
30
20
10
```

Overload & Override I

- Overload
  - ✓ 相同名稱之方法, 但使用不同之參數稱之
- Override
  - ✓ 子類別改寫繼承自父類別的方法, 方法之名稱使用之參數均相同, 但有不同之方法內容

```
Ex: class ABC {
      method1(int k) {…}
    }

    class DEF extends ABC {
      //與之上method1內容不同, override
      method1(int k) {…}

      //與之上第二個method1之參數及方法內容不同, overload
      method1(int k, int l){…}
    }
```

Overload & Override II

```
1  public class TestOverMethod {
2      public static void main(String[] args) {
3          ABC a = new ABC();
4          DEF d = new DEF();
5          System.out.println("In ABC a setInt( ): "+a.setInt());
6          System.out.println("In DEF d setInt( ): "+d.setInt());
7          System.out.println("In DEF d setInt(a): "+d.setInt(3));
8      }
9  }
10
11 class ABC {
12     int setInt() {
13         return 1;
14     }
15 }
16
17 class DEF extends ABC {
18     //與第一個setInt()參數相同、內容不同為上述方法之overriding方法
19     int setInt() {
20         return 2;
21     }
22     //與第二個setInt()參數不同為上述方法之overloading方法
23     int setInt(int a) {
24         return a;
25     }
26 }
```

```
In ABC a setInt( ): 1
In DEF d setInt( ): 2
In DEF d setInt(a): 3
```

Exception (參考)

- 當JVM在執行時期遇到以下原因會丟出例外
  (1) 不正常之執行, 如: 整數除以零, 陣列索引超出範圍,...
  (2) Java程式碼執行throw敘述
  (3) 發生非同步例外

- 編譯時期之例外檢查
  (1) Java語言藉由分析方法或建構子會引發的檢查例外來判斷程式中所包含之例外處理器.
  (2)方法或建構子的throws需包含需檢查例外之類別或其子類別
  (3) RuntimeException及其子類別, 類別Error及其子類別不需受編譯時期之例外檢查.

Interface & Implement (參考)

- 類別無多重繼承
  - ✓ Java不允許類別之多重繼承(D extend A,B,C)
  - ✓ 若有需求時, 需以interface為之
- 介面(Interface)
  - ✓ 可多重繼承
  - ✓ 只能做資料及方法之宣告, 不能定義方法之內容
  - ✓ 宣告格式
    - public interface [介面名稱] extends [其他介面]
- 介面實作
  - ✓ 某類別欲實現一介面時以下列格式為之
    - [修飾字] class [類別名稱] (extends [父類別名稱])
      implements [介面名稱] {
      ……
      }

範例練習

➢ 10進位轉16進位

✓ 讀入10進位整數，將其轉換為16進位，a-f的顯示可以大寫顯示

HexPrint1.java

```
1  import java.util.Scanner;
2  public class HexPrint1 {
3      public static void main(String[] args) {
4          Scanner cin = new Scanner(System.in);
5          System.out.print("Input an Integer: ");
6          int a = cin.nextInt();
7          System.out.println("a(DEC) = "+a);
8          System.out.println("a(hex) = "+Integer.toString(a, 16));
9          System.out.println("a(HEX) = "+Integer.toString(a, 16).toUpperCase());
10         cin.close();
11     }
12 }
```

```
Input an Integer: 123456789
a(DEC) = 123456789
a(hex) = 75bcd15
a(HEX) = 75BCD15
```

✓ 變化: 輸入之整數很大時，如a = 123456789012345678901

- 提示: 尋找大整數類別