

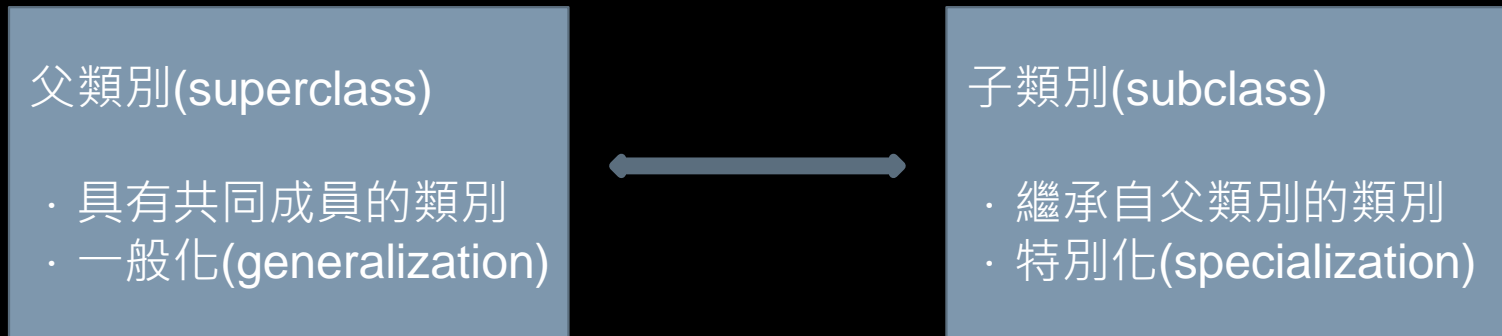
Module X

繼承與多型

1. 繼承(Inheritance)基本概念
 2. 繼承的優點與好處
 3. 繼承語法與注意事項
4. 覆寫(Overriding)目的與規定
5. 呼叫父類別方法 – super.
6. 呼叫父類別建構子 – super(...)
7. 多型 (Polymorphism)

繼承基本概念

- 一個類別可以繼承它的父類別所有狀態及行為，我們即把此稱之為**子類別**(subclass)延伸(extends)自**父類別**(superclass)



繼承優點與好處

- 繼承主要目的就是提高程式的重覆使用性
- 子類別將會繼承到父類別中所有可以存取的成員，包括變數與方法
 - **共同資料只要描述一次**
 - 如正職員工(FullTime)與打工仔(PartTime)都有共同資料，如姓名、電話、地址、性別等...
 - **處理共同資料的成員方法也只要描述一次**
 - 如每一個子類別可以使用定義在父類別的成員方法，如員工方法的 `getName(); getSalary();` 等
- 子類別繼承父類別功能之後：
 - 還可以再加入新方法 (method)
 - 也可以覆寫(override)從父類別而來的方法，建立一個專屬自己類別的運作邏輯

繼承語法與注意事項

- 語法：
 - **class** SubClassName **extends** SuperClassName
 - **class** FullTimeEmployee **extends** Employee {
 private double monthlySalary; //月薪
- 當B繼承自A，以「B **is a** A」表示
 - FullTimeEmployee **is a** Employee
 - FullTimeEmployee **has a** monthlySalary
- 注意：
 - Java不支援多重繼承，一個子類別只能**extends**一個父類別
 - 建構子(Constructor)無法被繼承
 - java.lang.**Object**類別為所有類別的共同父類別

覆寫目的與規定 (1/2)

- 目的：
 - 子類別繼承父類別後，不滿意父類別定義的方法，子類別可以在繼承後**重新改寫**，即為overriding
- 規定：
 - 子類別宣告覆寫(overriding)方法時，方法名稱、參數個數、參數型別與回傳值(註1)皆須跟父類別裡被覆寫的方法相同
 - 註1：JDK 1.5開始，如回傳型態是類別，則**可以是原方法回傳值型別的子類別**
 - 註2：**存取修飾子的等級不可以小於原方法**
(指可以一樣或是更加寬廣) (參考存取修飾子內容)
 - 註3：子類別覆寫父類別定義有throws的方法時，**不得**比父類別被覆寫方法的Exception還要高階 (參考例外處理章節)
 - 註4：後面提到 final 與 static 關鍵字時補充

覆寫目的與規定 (2/2)

- 子類別覆寫(override)父類別的方法時，其存取控制(access control)的範圍不可小於原方法(指的是可以相同或是開放等級更高)
- 例：

```
public class Father {  
    protected void doSomething() {...}  
}
```

```
public class Son extends Father {  
    private void doSomething() {...}    //失敗，private小於protected  
    void doSomething() {...}           //失敗，default小於protected  
    protected void doSomething() {...} //成功，與父類別相同存取等級  
    public void doSomething() {...}    //成功，大於父類別存取等級  
}
```

final修飾子

- 一個**類別**宣告為**final**，表示這個類別不能被繼承
 - public final class String {...}
 - public final class Math {...}
- 一個**方法**宣告為**final**，表示這個方法不能被覆寫(Override)
- 一個**變數**宣告為**final**，表示這個變數在初始值化後，不得再變更其值，也就是常數(Constant)
- 一個**物件參考變數**宣告為**final**，表示這個變數在初始值化後，不得再指向另一個物件

呼叫父類別的方法

- 子類別透過 **super.** 可以呼叫上一層類別的方法
 - 語法：**super.methodName()**;
 - 其中的methodName() 為上一層類別的方法
 - 無法越級呼叫


```
1 public class FullTimeEmployee extends Employee {
2     private double monthlySalary; //月薪
3
4     public void display() {
5         super.display();
6         System.out.println("月薪=" + monthlySalary);
7     }
8 }
```


呼叫父類別的建構子 (1/2)

- 子類別透過建構子，用 **super(...)** 將共同的建構子參數傳給父類別 (指共同的資料應使用父類別的建構子)
- 物件產生時，建構子呼叫的順序為先父類別再子類別，所以：
 - 建構子中**若有出現 super(...)**，一定要放在**第一個敘述位置**
 - 建構子中**若未出現 super(...)**，Java**預設**會有一個隱形的 **super()**
【呼叫父類別不帶參數的建構子】**預設自動放在第一個敘述的位置**


呼叫父類別的建構子 (2/2)

- public class **Employee**



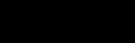
```
public Employee (int empno, String ename) {  
    this.empno = empno;  
    this.ename = ename;  
}
```

- public class **FullTimeEmployee** extends Employee
private double monthlySalary; //月薪



```
public FullTimeEmployee (int empno, String ename, double monthlySalary) {  
    super(empno, ename);  
    this.monthlySalary = monthlySalary;  
}
```

- public class **Manager** extends FullTimeEmployee
private double bonus; //獎金



```
public Manager (int empno, String ename, double monthlySalary, double bonus) {  
    super(empno, ename, monthlySalary);  
    this.bonus = bonus;  
}
```

範例：

Employee.java +
FullTimeEmployee.java +
Manager.java +
TestFullTimeEmployee.java +
TestManager.java

課堂練習

- 產生一個class，名為Elephant.java延伸自類別Animal
- 此類別有一個成員變數為name(種類名稱 – 型別String)
- 有一覆寫成員方法名為speak()，用以列印父類別的兩個成員變數和自己的成員變數
- 在main()裡透過建構子產生兩個Animal
 - 其一為Animal，其年紀和體重分別為3歲、8.0公斤
 - 另一為Elephant，其年紀、體重和種類名稱分別為8歲、1200.0公斤、大象
- 列印上述兩種Animal的值

多型 (1/6)

- 所謂多型(Polymorphism)是運用類別間繼承的關係，使父類別(superclass)可以當成子類別(subclass)的通用型態
- 子類別可以自動升級成父類別
 - Employee **e1** = new FullTimeEmployee(); // OK
 - Employee e2 = new Manager(); // OK
 - Employee e3 = new PartTimeEmployee(); // OK
- 父類別若是要轉型成子類別，則需要靠強迫轉型(**Casting**)，但是會在執行時期檢查是否能夠轉回適當的子類別
 - FullTimeEmployee f = (FullTimeEmployee)**e1**; // 轉型
 - Manager m = (Manager)**e1**; //執行發生java.lang.ClassCastException

多型 (2/6)

- **instanceof** 運算子常被用來判斷父類別參考真正指向何種子類別的實體
 - 語法：物件參考變數 instanceof 類別名稱
 - 說明：檢查左邊的物件是否可以轉型為右邊的類別型態，如果可以回傳true，否則為false
- Employee **e1** = new FullTimeEmployee();
 - System.out.println(**e1** instanceof FullTimeEmployee); // true
 - System.out.println(**e1** instanceof Manager); // false
 - System.out.println(**e1** instanceof PartTimeEmployee); // false

多型 (3/6)

- 用父類別的型別(參考)，指向子類別的物件，並對應到子類別 **overriding** 的方法
 - 父類別會先判斷實際的子類別物件是哪一個，再呼叫此子類別裡對應的 **overriding** 方法

```
Manager m = new Manager(7003, "David", 50000.0, 10000,0);  
double salary = m.getSalary();
```

```
Employee e = new Manager(7003, "David", 50000.0, 10000.0);  
double salary = e.getSalary();
```

- 以上 **getSalary()** 在最後執行時，都是子類別 **Manager** 的方法
但父類別的 **getSalary()** 還是不可省去，否則無法進行對應造成錯誤

多型 (4/6)

- public class **Employee**
private int empno;
private String ename;
→ public double getSalary() { return 0; }
- public class **FullTimeEmployee** extends Employee
private double monthlySalary; //月薪
→ public double getSalary() { return monthlySalary; }
- public class **Manager** extends FullTimeEmployee
private double bonus; //獎金
→ public double getSalary() {
double monthlySalary = super.getSalary();
return monthlySalary + bonus;
}

多型 (5/6)

- public class **Employee**
private int empno;
private String ename;
→ public double getSalary() { return 0; }
- public class **PartTimeEmployee** extends Employee
private double hourlyPay; //時薪
private int workHour; //工時
→ public double getSalary() {
return hourlyPay * workHour;
}

多型 (6/6)

- 多型測試

```
public class TestPolymorphism2 {  
    public static void main(String[] args) {  
        Employee[] e = new Employee[3];  
        e[0] = new FullTimeEmployee(7002 , "peter", 40000.0 );  
        e[1] = new Manager(7003 , "merry", 50000.0 , 10000.0);  
        e[2] = new PartTimeEmployee(7004 , "John" , 1000.0, 8);  
        for (int i = 0; i < e.length; i++)  
            System.out.println(e[i].getSalary());  
    }  
}
```

範例：

EmployeePoly.java +
FullTimeEmployeePoly.java +
ManagerPoly.java +
PartTimeEmployeePoly.java +
TestPolymorphism2.java

課堂練習

- 產生一個class，名為PolyAnimal.java
- 程式同上一個課堂練習
- 在main()裡透過多型來製作

章節整理

- 繼承最大優點是提昇程式碼的使用，避免重覆撰寫
- Java使用**extends**關鍵字作為繼承父類別用
- Java不允許多重繼承，一個類別只能繼承一個父類別
- 我們可以使用**super**呼叫父類別的屬性與方法
- 多型運用了類別之間繼承關係，讓父類別可當成子類別的通用型態
- 要使用對應的方法，子類別必須改寫父類別的方法
- 注意改寫(Override)的相關規定
- 判斷資料型態是否可以轉型，我們可以使用**instanceof**運算子進行判斷，回傳true/false