**5/5　認識資料結構**

**sheep programming class**

**資訊學院資傳學習家族**

**［１］認識ADT**

**資料型態 Data Type**

**→　能以量表示的型態**

**抽象資料型態 Abstract Data Type**

**→　無法以量表示的型態，內部放的資料通稱為元素（Element）**

**［２］常見的ADT**

**串列 List**

**→　一個一個元素串接在一起**

**堆疊 Stack**

**→　每次只能動用最後一個元素的增加或刪除**

**特色：先進後出（First In Last Out）　後進先出（Last In First Out）**

**佇列 Queue**

**→　元素只能從後面增加，刪除只能從頭開始刪**

**特色：先進先出（First In First Out） 後進後出（Last In Last Out）**

**［３］認識泛型**

**sheep programming class**

**資訊學院資傳學習家族**

**泛型 Generic Type**

**泛型是在Java5後新增的語法，使設計API時，讓程式碼更簡潔。**

**泛型的使用範例：**

**Stack<Integer> st = new Stack<Integer>();**

**可使用push與pop**

**泛型的實作範例：**

* **GenericFoo.java**

**public class GenericFoo<T> {  
 private T foo;  
   
 public void setFoo(T foo) {  
 this.foo = foo;  
 }  
   
 public T getFoo() {  
 return foo;  
 }  
}**

**GenericFoo<Boolean> foo1 = new GenericFoo<Boolean>();  
GenericFoo<Integer> foo2 = new GenericFoo<Integer>();  
   
foo1.setFoo(new Boolean(true));  
Boolean b = foo1.getFoo();  
   
foo2.setFoo(new Integer(10));  
Integer i = foo2.getFoo();**

**foo1與foo2「相當於」下面兩個類型（只是個比喻）：**

**sheep programming class**

**資訊學院資傳學習家族**

**public class GenericFooBoolean {  
    private Boolean foo;  
   
    public void setFoo(Boolean foo) {  
        this.foo = foo;   
    }  
   
    public Boolean getFoo() {  
        return foo;  
    }  
}**

**以及：**

**public class GenericFooInteger {  
    private Integer foo;  
   
    public void setFoo(Integer foo) {  
        this.foo = foo;  
    }  
   
    public Integer getFoo() {  
        return foo;  
    }  
}**

**［４］介面與類別的關係**

**介面 Interface**

**通常類別不予許多重繼承的情況下發生，也就是每個類別最多只能有一個父類別，而Java提供的Interface允許多重繼承，但是內部都只是類似抽象的型態，並沒有被實作。**

**一般類別的繼承方式：**

**sheep programming class**

**資訊學院資傳學習家族**

**class Super{…}**

**class Sub extends Super{……}**

**介面的繼承方式：**

**interface Indoorsy{**

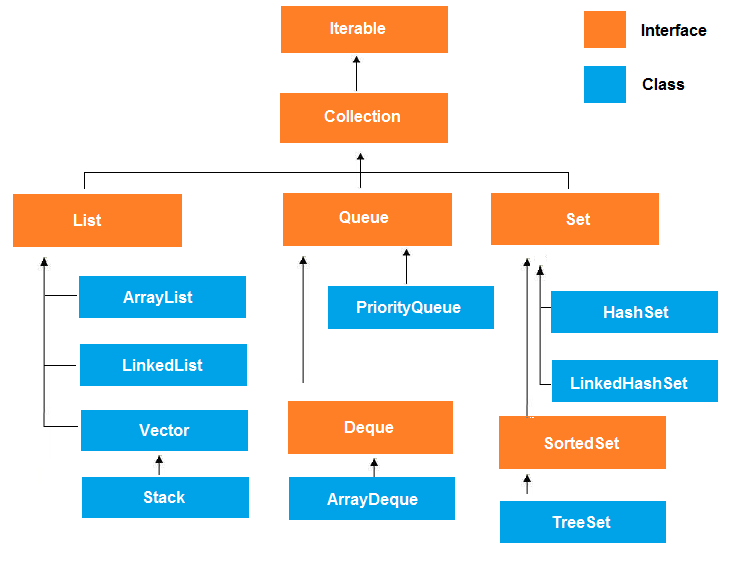
**void eat(String food);**

**void poo();**

**}**

**class Man implements Indoorsy{……}**

**［５］Java Collections Framework**

****