Module XI
介面與多型

1. abstract修飾子(抽象類別)
2. 介面(interface)
3. 介面設計模式

抽象類別 abstract class

- 抽象方法沒有方法主體，且必須加上abstract修飾子

  - public **abstract** void myMethod();

- 抽象類別不一定要有抽象方法，但具有抽象方法的類別，一定要宣告為抽象類別

  - public **abstract** class MyClass {…}

- 一個類別只要加上abstract修飾子(即使它裡面不含任何abstract方法)，它就無法產生實體，只能透過繼承來建立延伸子類別

- 一個(子)類別若繼承了抽象父類別，除非它實作了抽象父類別當中的所有抽象方法，否則它仍然只是個抽象類別

- 使用時機 (上課詳述)

  在建立類別時，若有方法尚未決定如何設計內容主體時，就可將此方法加上abstract修飾子成為抽象方法，之後再由繼承的子類別來實做

範例：TestAbstract.java

介面 (interface) (1/2)

- **Java使用介面(interface)的主要目的如下：**
- **多重繼承**

  - Java只能單一繼承，而介面可以實現物件導向中的多重繼承(替代C++中的多重繼承)

  - class 子類別 **extends** 父類別 **implements** 介面1, 介面2, … {…}

  - class 子類別 **implements** 介面1, 介面2, … {…}

- 預先**定義規格**給實作此介面的所有子類別

  - 介面可說是一種所有方法皆為抽象方法的抽象類別，所以子類別必須實作介面的所有抽象方法

  - 而介面跟介面之間是可以再繼承(**extends**)的

介面 (interface) (2/2)

- **Java使用介面(interface)的主要目的如下(續)：**
- **貼標籤、型態轉換、降低相依性**

  - 介面也是一種參考型態，也就是說介面提供了另一種彈性，使子類別在繼承原父類別的特性之外，也能具有其他型態的特性
  - 因為一個物件可以實作多個介面，所以每一個父介面都可以當作此物件的(父)多型之一，也因此用介面來幫物件貼標籤、作型態轉換將是一件容易的事情
  - 降低相依性的觀念範例，見比較性範例：

    · 低凝聚性 - 高相依性：Pencil.java, InkBrush.java, WorkWithPens.java, WriteBusinessTest.java

    · **高凝聚性 - 低相依性**：IWritable.java, Pencil2.java, InkBrusch2.java, WOrkWithPens2.java, WriteBusinessTest2.java

範例：TestInterface.java

空介面

- 沒有定義任何方法的介面叫做空介面
- 一個類別可以implements某個空介面，好消息是不需實作任何方法，但該類別的任何實體即已經成為該介面的一個合法實體
- java.lang.**Cloneable** 和 java.io.**Serializable**是比較有名的兩個空介面
  - 一個類別implements前者java.lang.Cloneable空介面時，該類別的物件才可以做**物件的複製**
  - 一個類別implements後者java.io.Serializable空介面時，該類別的物件才可以做**物件的序列化**(將物件永久儲存(persistence)，稱做序列化)

修飾子適用的場合 (補充)

修飾子	類別	實體變數	方法	建構子	程式區塊
public	✔	✔	✔	✔	-
protected	-	✔	✔	✔	-
default	✔	✔	✔	✔	-
private	-	✔	✔	✔	-
final	✔	✔	✔	-	-
static	-	✔	✔	-	✔
abstract	✔	-	✔	-	-

章節整理

- 抽象類別(abstract class)內無如實做所有抽象方法，則繼承的子類別也得宣告成為抽象類別
- Java因介面而偉大，實現了多重繼承、預設規格，更能做到型態轉換操作(多型)
- 介面內只能有抽象方法，如有變數則必須給予初值
- 瞭解抽象類別與介面的使用時機