12. Inheritance in Java

Java 的類別同樣有繼承的功能，衍生類別同樣會繼承基本類別所有的資料成員與函式成員，當繼承而來的函式成員不適於衍生類別時，同樣可以在衍生類別之定義內將其重新定義(override)，並可進一步加入新的成員(資料或函式)，以符合衍生類別應有的屬性與行為。

一基本類別(父類別)可以有需多繼承它的衍生類別(子類別)，與 C++不同的是，一子類別只能繼承一個父類別，多重繼承於 Java 中不允許。

Java 只有一種繼承模式，並不像 C++，有 public、protected、與 private 三種繼承模式。

在多型方面，Java 沒有非虛擬函式，Java 的每一個函式成員，不需明確宣告，均即是虛擬函式，Java 所有物件均必定透過參考來操作，表現出多型的行為。

12.1 Extending Classes in Java

Java 定義子類別時，是使用 extends 關鍵字，來繼承父類別。

Java 之子類別隱含著一個稱為 super 的參考，可用其來引用父類別之建構函式(不會繼承給衍生類別)，於函式重新定義(override)時呼叫父類別之函式成員，亦可用來明確地對繼承自父類別之公開資料成員做存取。

例：

```
class Employee
{
    private String name;
    private int deptID;
    ……
    public Employee(String n, int d)
    {
        name = n;
        deptID = d;
    }
```

```
        public void print( )
        {
            System.out.println(name);
            …….
        }
        …….
}

class Manager extends Employee
{
        private int rank;
        …….
        public Manager(String n, int d, int r)
        {
            super(n, d);            // initialize base members
            rank = r;               // initialize local members
        }

        public void print( )                        // override “print”
        {
            super.print( );         //calling print() of base class
            System.out.println(rank);
            …….
        }
        …….
}
```

程式範例: oop_ex109.java

注意要點:

1. Manager 類別繼承了 Employee 類別，這項指定出現於 circle 類別定義的第一行，extends 代表了繼承。
2. 衍生類別會繼承所有基本類別的成員，但於衍生類別的建構函式中，無法直接存取基本類別的私有成員(x, y)，為了對繼承自基本類別的私有成員初始化，必須要使用基本類別的建構函式，呼叫方式是使用 super 關鍵字。
3. 如需要將繼承自基本類別的函式成員，依衍生類別的需要修改或重新定義

(例如 print)，則我們須於衍生類別中重新宣告此一函式成員，並提供其新的定義，修改或重新定義後的函式將於衍生類別中取代繼承自基本類別的定義。

4. 如果對某一繼承自基本類別的函式成員，我們不需完全更改其原本的定義，只需要加入新的功能(例如 print)，我們可以不必重寫已定義於原基本類別之原函式的部分，而只要呼叫此一原函式，呼叫方式同樣是使用 super 關鍵字。
5. super 關鍵字可用來明確地對父類別之公開資料成員做存取(vs. this 關鍵字)。
6. Java 的所有函式，不需明確宣告，即是虛擬函式，表現出多型的行為。

12.2 Final Keyword

Java 中，宣告為 final 之資料成員，其值不可被修改，而宣告為 final 之類別，其不可被繼承。

程式範例: oop_ex110.java

注意要點:

Java 之 final 類別與 C++之 const 類別意義大不相同。

12.3 Abstract Methods and Abstract Classes in Java

Java 的類別同樣可宣告為無法產生實體的抽象類別(Abstract Classes)，為其子類別定義共同應實做的介面。C++中只定義函式的呼叫方法，而不提供實作內容的純虛擬函式，Java 中稱為抽象方法(Abstract Methods)。

Java 的抽象類別與抽象方法，需使用 abstract 關鍵字明確宣告。

例：

```
abstract class shape
{
        public abstract void move( ) ;
        public abstract void draw( ) ;
```

```
};
```

程式範例: oop_ex111.java

注意要點:

1. 抽象方法不能有{}(method body)。
2. 具有抽象方法之類別，需明確以 abstract 關鍵字宣告為抽象類別。
3. 抽象類別，無法產生實體，但可產生其參考。
4. 衍生類別繼承了於上層中尚未定義任何實作內容的抽象方法，如此衍生類別沒有對其提供實作加以定義，則此函式仍為抽象方法，此衍生類別亦為抽象類別。

12.4 Interface in Java

Java 之類別只能繼承(extends)一個類別，但可以實作(implements)一個以上的介面(Interface)。

介面的定義與類別很類似，但介面之成員只能是常數與抽象方法，故介面可視為沒有資料成員，且其方法均為虛擬方法的虛擬類別。

介面的作用，是定義了繼承其之類別均需要有(實作)的共同能力，但不提供任何實作內容，也不提供任何資料成員，給子類別繼承。

Java 以介面之設計，來避免 C++之多重繼承所可能造成的重複繼承問題。

相對於 extends，類別對於介面的實作，是使用 implements 關鍵字。

例：

```
interface printable
{
        public static final int count = 0 ;
        public abstract void print( ) ;
        public abstract void printToFile(String filename) ;
};
```

```
class circle extends shape implements printable, scaleable
{
    ……
    public void print( ){System.out.println("do print circle");}
    public void printToFile(String filename){System.out.println("do print circle on"+filename);}
    ……..
}
```

程式範例: oop_ex112.java

注意要點:

1. 介面之成員只能是常數與抽象方法。
2. 介面所定義之抽象方法均無實作內容，故繼承(implements)介面的類別需對繼承的每一個抽象方法提供實作，例如 triangle、circle、square，否則需宣告為抽象類別例如 shape。

12.5 Interface Inheritance

介面與類別一樣，可以繼承，同樣可使用 extends 關鍵字產生子介面，子介面繼承父介面所有的成員(抽象函式)，並可新增屬於自己的抽象函式或常數。

程式範例: oop_ex113.java

1. 繼承而產生仍為介面，故新增成員只能是常數與抽象方法。
2. 與類別不同，子介面可繼承多個介面。

12.6 instanceof operator in Java

Java 提供了 instanceof 運算子，用來檢查「某個物件」是不是「某個類別之物件」。也可以檢查「某個物件」是不是有實作某個介面。

例：

```
boolean flag1 = ci instanceof circle;
boolean flag2 = ci instanceof printable;
```

程式範例: oop_ex114.java

1. 「某個物件」是其父類別之物件。
2. 「某個物件」實作了某子介面，亦實作了其父介面。
3. Java 物件均表現出多型行為。

12.7 Object Cloning in Java

Java 的物件，均是透過參考來做操作，對於參考間以等號相連，並非對其所各自參照的物件資料作拷貝，而拷貝的是位址值，故 Java 無法如 C++之拷貝指定運算子，複製產生物件。

對於複製物件，Java 提供了 Object Cloning 之機制，即一物件可依照本身的狀態複製產生一資料相同的新物件，並傳回新物件的位址，如下。

```
classX  ref1  =  (classX) ref2.clone( );
```

clone( )無法直接使用，自訂類別需依其本身的資料結構，有如 C++的拷貝指定運算子，實作其內容，Java 的設計是，欲提供 clone 功能的類別，均需 implement 內建的 Cloneable 介面，並提供實作。

程式範例: oop_ex115.java

1. Cloneable 為內建的介面。
2. Java 所有類別之 root 為 Object 類別，對於傳回值可能是各種不同自訂類別物件的情況，例如 clone( )，可以用 Object 作為傳回值型態來做為涵蓋所有情況之處理，因為所有的物件均可當作 Object 的物件，傳回之後再做型別轉換。
3. toString( )為定義於 Object 類別中的函式成員，所有類別均有繼承它，只要對此函式依類別內容做重新定義，其於必要時可被呼叫將物件的資料轉成字串，例如本範例，我們可使用參照到 Person 物件之參考直接做輸出。
4. Java 有非常豐富的內建類別與介面，針對各種應用，訂定了基本的架構，供程式設計者進一步的延伸。

12.8 Java Standard Packages(標準類別庫)

java.lang	The core language classes, such as String, Thread, Class, and so on.
java.util	Classes for general utility.
java.io	Input/Output and some file system manipulation
java.math	Mathematical manipulation.
java.text	Text manipulation, such as formatting and parsing numbers and dates, sorting/comparing strings, and message lookup, etc.
java.awt	The Abstract Window Toolkit abstract layer for writing platform-independent graphical user interfaces.
java.applet	The Applet class and related types for writing applets that can be run in HTML browsers.
java.net	Networking classes for sockets, URLs, and so on.
java.rmi	Remote Method Invocation, which lets you invoke methods on objects running in different virtual machines, usually on different host computer.
java.sql	The JDBC package for accessing relational database from Java.
java.security	Encryption, authentication, digital signatures, and other useful security-related code.
java.beans	For writing user-composable code.