國立屏東大學 資訊工程學系 物件導向程式設計

14. Inheritance and Overriding

Inheritance(繼承性)是指讓某一類別的物件繼承來自其它類別的屬性與行為。C++支援多重繼承，讓一個類別可以繼承多個類別；但Java語言僅支援單一繼承，這只得其成為較為簡單的語言，對初學者來說，是一件讓人輕鬆的事。不過Java也可以單一繼承、多重實作的方式，來達成多重繼承的效果。

14.1 IS-A relationship

「A is a kind of B.」

繼承性其實就是所謂的ISA關係 – IS-A relationship，是指兩個類別A與B間存在著「A類別就是B類別的一種特殊化。所謂的「特殊化(specialization)」是指A類別其實就是B類別，但A類別比B類別特殊些。我們將A類別稱為子類別，B類別則稱為父類別; 在Java語言的術語，則分別稱為sub class與super class。

回顧我們在前一章開始介紹的Person類別，每個屬於這個類別的物件都會具有firstname, lastname等屬性，以及showInfo()的行為。

```
public class Person
{
   // field declarations
   private String firstname;
   private String lastname;

   Person()
   {
       firstname="unknown";
       lastname="unknown";
   }

   Person(String f, String l)
   {
       firstname=f;
       lastname=l;
   }

   // method declarations
   public void showInfo()
   {
      System.out.println("Name: " + firstname + " " + lastname );
   }

   public void setFirstname(String f) { firstname = f; }
```

```
   public void setLastname(String l) { lastname = l;}
   public String getFirstname() {return firstname;}
   public String getLastname() {return lastname;}

}
```

STUScoreMan.Student

如果我們要設計一個學生成績管理系統(Student Score Management System, STUScoreMan)，在我們所要建構的物件導向世界中，必然會存在有「學生」這種類別的物件。那要如何開始設計學生這個類別呢？先考慮學生只需具備姓名與學號兩個屬性，那麼我們可以撰寫以下的程式碼：

```
class Student
{
   // field declarations
   String firstName;
   String lastName;
   String ID;

   // constructors
   ...
   // method delcarations
   ...
}
```

就如同前一章所介紹的，我們還應該為這個類別加上建構子、顯示學生資料的method，設定存取權限、設計setters與getters...，我們發現這個新設計的Student類別，與Person類別很相似….

「Student也是一種Person，只是比較特別！比起Person，Student還多了ID(目前為止)。」

以這樣的角度看問題時，我們可以說，Student is a kind of Person!，我們可以用以下的程式碼，告訴電腦這件事：

```
class Student extends Person { }
```

「extends」是Java的keywords之一，用以表示Student類別是「延伸自」Person類別，換句話說，Student類別要繼承Person類別。

在這個ISA的繼承關係中，我們將Person類別稱為「父類別(super class)」，Student類別則稱為「子類別(sub class)」。

一旦你完成這樣的設計，儘管現在在Student類別中，一行程式碼都還沒寫，但Person類別該有的，Student類別也都會有，包含Person類別的屬性與行為。因此，我們可以在物件導向的世界中，把Student類別的物件視為是Person類別的物件，並且使用它所公開(public)的屬性與行為。請參考下面的程式碼：

```
public class test {
```

```
    public static void main(String[] args) {
        Student amy;
        amy = new Student();
        amy.setFirstname("Amy");
        amy.setLastname("Wang");
        amy.showInfo();
    }
}
```

沒有意外地，其執行結果如下：

```
Name: Amy Wang
```

但這不是我們要的結果，雖然Student已經是a kind of Person，但Student不夠特殊，它跟Person根本是一樣的。讓我們將新的屬性與行為加到Student類別中，請參考下面的程式碼：

```
public class Student extends Person
{
    // fields
    private String ID;

    // contructors
    Student()
    {
        setFirstname("unknown");
        setLastname ("unknown");
        ID = "unknown";
    }

    Student(String f, String l, String i)
    {
        setFirstname(f);
        setLastname(l);
        ID = i;
    }
    // methods
    public void setID(String id) { ID=id; }
    public String getID() { return ID; }
}
```

在Person.java中，firstname與lastname是定義為private，所以連子類別都不能使用，必須透過其setters與getters才能存取。

讓我們為這個小節做個總結：

Student IS A (kind of) Person

在Java語言裡，使用extends這個keyword來完成的ISA繼承關係，子類別(subclass)可以繼承父類別(super class)所有的屬性與行為，除了那些被定義為私有的(private)

14.1.1 instanceof

如果在程式執行時，我們想知道某個物件是屬於何種類別，那可以使用instanceof運算子。

```
System.out.println(amy instanceof Person);
System.out.println(amy instanceof Student);
```

其輸出結果為：

```
true
true
```

instanceof運算傳回的是boolean值，表示amy這個物件是Person類別的實體，同時它也是Student類別的實體。

14.2 this and super

14.2.1 this

在類別定義裡this表示自己這個類別。我們可以將其建構子改寫如下：

```
    Student()
    {
        //等於呼叫Student("unknown", "unknown", "unknown");
        this("unknown", "unknown", "unknown");
    }

    Student(String f, String l, String i)
    {
        setFirstname(f);
        setLastname(l);
        ID = i;
    }
```

14.2.2 super

在類別定義裡super表示自己所繼承的父類別(super class)我們可以將其建構子改寫如下：

```
    Student()
    {
        //等於呼叫Student("unknown", "unknown", "unknown");
        this("unknown", "unknown", "unknown");
    }

    Student(String f, String l, String i)
    {
        super(f,l);  //等於去呼叫父類別的Person(f,l)
        ID = i;
    }
```

14.3 Overriding (覆載)

STUScoreMan.Teacher

再舉一個例子，同樣針對這個STUScoreMan我們還需要設計一個名為Teacher的類別。每個老師當然會有姓名，以及「一項」專長(當然只是假設啦，我們有些老師連廚師證照都有呢！)。首先，我們針對專長的部份，設計一個列舉型別，如下：

```
public enum Expertise
{
   Java, OS, Database, Algorithms, Networking
}
```

接著透過同樣透過　繼承Person來快速地完成Teacher類別的設計：

```
public class Teacher extends Person
{
   private Expertise expertise;

   Teacher()
   {
      this("unknown","unknown", null);
   }

   Teacher(String f, String l, Expertise exp)
   {
      super(f,l);
      expertise = exp;
   }

   public void setExpertise(Expertise exp) { expertise = exp; }
   public Expertise getExpertise() { return expertise; }
```

```
}
```

可是當我們使用以下這行來宣告並產生一個Teacher類別的物件實體時：

```
Teacher junwu = new Teacher("Jun", "Wu", Expertise.Java);
```

若呼叫junwu.showInfo()，我們只會得到這位老師的姓名，並沒有其它資訊。所以我們將在Teacher.java中加入以下的method:

```
   public void showInfo()
   {
       super.showInfo();
       System.out.print("Expertise: ");
       switch(expertise)
       {
          case Java:
               System.out.println("Java Programming");
               break;
          case OS:
          case Database:
          case Algorithm:
          case Networking:
               System.out.println("something interesting");
       }
   }
```

在ISA的繼承關係中，若sub class定義與super class相同的method，就稱為Overriding，意即提供新的版本供sub class類別使用。若是要呼叫原本super class類別中的method時，可以使用super.XXX()方式達成，例如super.showInfo()，就是呼叫其所繼承的super類別的showInfo() method，其結果可以輸出其姓名；然後我們再進一步將其專長輸出。

Overriding常用於提供不同於super class的method版本，畢竟ISA是一種特殊化的關係，sub class與super class十分相同，也繼承了來自super class的一切定義與宣告，可是可以針對自己的特殊性，定義自己的method，由於是做同一件事情，所以我們使用同樣的method名稱，以符合「真實世界」的通則(當然，這也是一種簡化，否則我們可能要提供showTeacherInfo()這種命名，如此會使得不同class對同一件事情的操作方法不同)。修改後完整的程式碼如下：

```
public class Teacher extends Person
{
   private Expertise expertise;

   Teacher()
   {
      this("unknown","unknown", null);
   }

   Teacher(String f, String l, Expertise exp)
```

```
   {
      super(f,l);
      expertise = exp;
   }

   public void setExpertise(Expertise exp) { expertise = exp; }
   public Expertise getExpertise() { return expertise; }

   public void showInfo()
   {
       super.showInfo();
       System.out.print("Expertise: ");
       switch(expertise)
       {
          case Java:
               System.out.println("Java Programming");
               break;
          case OS:
          case Database:
          case Algorithm:
          case Networking:
               System.out.println("something interesting");
       }
   }
}
```

要覆載一個Mthod必須遵守以下的規則：

- 新Method的引數列必須與要覆載的對象一致。
- 新Method的傳回型態必須與要覆載的對象一致或是其子型態。
- 新Method的存取限制不可以比原先還要更嚴格。
- 被宣告為final的method不可以被overridden
- Constructors不可以被overridden

14.4 HAS-A relationship

A has a B

類別間的HAS-A關係是指某類別包含有其它類別。例如A has a B如此一來，在A類別的程式碼中，還可以透過其擁有的B類別的物件，去使用B類別所定義的屬性或行為。

回顧我們所要設計的一個學生成績管理系統(Student Score Management System, STUScoreMan)假設一個學生，只能修一門課，那麼在我們所要建構的物件導向世界中，必然會存在有「學生」這種類別的物件，以及「課程」類別的物件。

我們先設計一個課程的類別如下(其中包含有授課教師的資料):

```
public class Course
{
   private String courseName;
   private String classroom;
   private Teacher teacher;   // Here is a HAS-A relationship

   Course()
   {
      this("unknown", null, "unknown");
   }

   Course(String cn, Teacher t, String cr)
   {
      courseName = cn;
      teacher = t;
      classroom = cr;
   }

    public void showInfo()
    {
        System.out.println("Course Name:" + courseName);
        System.out.println("Teacher: ");
        teacher.showInfo();
        System.out.println("Classroom: " + classroom);
    }

    public String getCourseName()
    {
        return courseName;
    }

}
```

我們也將Student.java進行部份的修改，加上「一個學生只能修一門課」的假設，以及將showInfo()加以overriding其程式碼如下：

```
public class Student extends Person
{
    // fields
    private String ID;
    private Course course; // Here is a HAS-A relationship

    // contructors
    Student()
    {
        this("unknown","unknown","unknown"); //改成用this
        ID = "unknown";
    }
```

```
    Student(String f, String l, String i)
    {
        super(f,l);
        ID = i;
    }
    // methods
    public void setID(String id) { ID=id; }
    public String getID() { return ID; }

    public void takeCourse(Course c)
    {
        course = c;
    }

    public void showInfo()
    {
        super.showInfo();
        if(course!=null)
            System.out.println("Course: " + course.getCourseName());
    }

}
```

這兩個類別中，都含有Has-A的關係，例如Course中的private Teacher teacher;與Student中的private Course course;

最後，讓我們設計一個Main類別，以便測試程式是否正確執行。

```
public class StudentScoreMan {

    public static void main(String[] args) {

        Student amy;
        amy = new Student();

        amy.setFirstname("Amy");
        amy.setLastname("Wang");
        amy.showInfo();

        System.out.println(amy instanceof Person);
        System.out.println(amy instanceof Student);

        Teacher junwu = new Teacher("Jun", "Wu", Expertise.Java);
        Course oop = new Course("Object-Oriented Programming", junwu,
"CC0802");
        junwu.showInfo();
        oop.showInfo();
```

```
        amy.takeCourse(oop);
        amy.showInfo();
    }
}
```

```
Name: Amy Wang
Name: Jun Wu
Expertise: Java Programming
Course Name:Object-Oriented Programming
Teacher:
Name: Jun Wu
Expertise: Java Programming
Classroom: CC0802
Name: Amy Wang
Course: Object-Oriented Programming
```