Java 8 訓練課程
Chapter 10

antallen@gmail.com 紀宏宜

PS:本教材內容取自「良葛格」網站－https://openhome.cc/Gossip/Java/Protected.html

Outline

- Java 繼承語法細節
  - protected 成員
  - 重新定義的細節
  - 再看建構式
  - 再看 final 關鍵字
  - java.lang.Object
  - 再看抽象類別

Java 繼承語法細節

- protected 成員
  - 就之前的RPG遊戲來說，如果建立了一個角色，想顯示角色的細節，必須如下撰寫：

```
SwordsMan swordsMan = new SwordsMan();
...略
System.out.printf("劍士 (%s, %d, %d)%n", swordsMan.getName(),
        swordsMan.getLevel(), swordsMan.getBlood());

Magician magician = new Magician();
...略
System.out.printf("魔法師 (%s, %d, %d)%n", magician.getName(),
        magician.getLevel(), magician.getBlood());
```

  - 這對使用SwordsMan或Magician的 Demo 程式端有點不方便，有改善的空間嗎？

Java 繼承語法細節

- protected 成員
  - 在SwordsMan或Magician上定義個toString()方法，傳回角色的字串描述：

```
public class SwordsMan extends Role {
    ...略
    public String toString() {
        return String.format("劍士 (%s, %d, %d)", this.getName(),
                this.getLevel(), this.getBlood());
    }
}
```

```
public class Magician extends Role {
    ...略
    public String toString() {
        return String.format("魔法師 (%s, %d, %d)", this.getName(),
                this.getLevel(), this.getBlood());
    }
}
```

Java 繼承語法細節

- protected 成員
  - Demo 程式端就可以如下撰寫：

```
SwordsMan swordsMan = new SwordsMan();
...略
        System.out.println(swordsMan.toString());

Magician magician = new Magician();
...略
        System.out.printf(magician.toString());
```

  - 但是....
    - toString()在取得名稱、等級與血量時不是很方便
    - 因為Role中的name、level與blood被定義為private
    - 只能透過getName()、getLevel()、getBlood()來取得。

Java 繼承語法細節

- protected 成員
  - 將Role中的name、level與blood定義為protected:

不要改成public

```
public abstract class Role {

    protected String name;
    protected int level;
    protected int blood;

    ...略
}
```

  - 被宣告為protected的成員:
    - 相同套件（Class 目錄）中的類別可以直接存取
    - 不同套件（Class 目錄）中的類別可以在繼承後的子類別直接存取。

Java 繼承語法細節

- protected 成員
  - 現在SwordsMan可以如下定義toString()：

```
public class SwordsMan extends Role {
    ...略
    public String toString() {
        return String.format("劍士 (%s, %d, %d)", this.name,
                this.level, this.blood);
    }
}
```

利用 this直接取得物件屬性！

  - Magician也可以如下撰寫：

```
public class Magician extends Role {
    ...略
    public String toString() {
        return String.format("魔法師 (%s, %d, %d)", this.name,
                this.level, this.blood);
    }
}
```

Java 繼承語法細節

- protected 成員
  - 現在，整理一下權限：

關鍵字	類別內部	相同套件類別	不同套件類別
public	可存取	可存取	可存取
protected	可存取	可存取	子類別可存取
無	可存取	可存取	不可存取
private	可存取	不可存取	不可存取

  - 權限小-->大:
    - private -> (無) -> protected -> public
    - 一開始不知道使用哪個權限，就先使用private

Java 繼承語法細節

- 重新定義(Override)的細節
  - 重新定義方法：
    - 非完全不滿意父類別中的方法，只是希望在執行父類別中方法的前、後作點加工。
    - 例如，也許Role類別中原本就定義了toString()方法：

```
public abstract class Role {
    ...略
    public String toString() {
        return String.format("(%s, %d, %d)", this.name,
                this.level, this.blood);
    }
}
```

不太符合子類別利用！

    - 如果可以執行Role中的toString()方法取得字串結果，再串接"劍士"字樣，才符合想要的描述！
    - 應該如何做？

Java 繼承語法細節

- 重新定義(Override)的細節
  - 在Java中，如果想取得父類別中的方法定義，可以於呼叫方法前，加上super鍵字。例如：

```
public class SwordsMan extends Role {
    ...略
    @Override
    public String toString() {
        return "劍士 " + super.toString();
    }
}
```

```
public class Magician extends Role {
    ...略
    @Override
    public String toString() {
        return "魔法師 " + super.toString();
    }
}
```

Java 繼承語法細節

- 重新定義(Override)的細節
  - 重新定義方法要注意，對於父類別中的方法權限，只能擴大但不能縮小。
  - 子類別中重新定義時不可為private或protected。
  - 在JDK5之前，重新定義方法時除了可以定義權限較大的關鍵字外，其它部份必須與父類別中方法簽署(方法名稱)完全一致。例如：

```
public class Bird {
    protected String name;
    public Bird(String name) {
        this.name = name;
    }
    public Bird copy() {
        return new Bird(name);
    }
}
```

Java 繼承語法細節

- 重新定義(Override)的細節
  - 如果 Chicken 繼承Bird，打算讓 copy()方法傳回Chicken，那麼在JDK5之前會發生編譯錯誤：

```
public class Chicken extends Bird {
        public Chicken(String name){
            super(name);
        }
        public Chicken copy() {
            Chicken chicken  =  new Chicken(“Justice”);
            chicken.name = “peter”;
        return chicken;
    }
}
```

JDK5 之後，將不會出現編譯錯誤訊息！

  - static方法屬於類別擁有，如果子類別中定義了相同簽署的static成員，該成員屬於子類別所有，而非重新定義
  - static方法也沒有多型，因為物件不會個別擁有static成員。

Java 繼承語法細節

- 再看建構式
  - 先執行父類別建構式定義的流程，再執行子類別建構式定義的流程
  - 建構式可以重載，父類別中可重載多個建構式，如果子類別建構式中沒有指定執行父類別中哪個建構式，預設會呼叫父類別中無參數建構式。

```
class Some {
    Some() {
        System.out.println("呼叫Some()");
    }
}
class Other extends Some {
    Other() {
        System.out.println("呼叫Other()");
    }
}
```

先執行Some()中的流程，
再執行Other()中的流程

Java 繼承語法細節

- 再看建構式
  - this()與super()只能擇一呼叫，而且一定要在建構式第一行執行。
  - 錯誤的範例：

```
class Some {
        Some(int i) {
        System.out.println("呼叫Some(int i)");
    }
}
class Other extends Some {
    Other() {
        System.out.println("呼叫Other()");
    }
}
```

父類別中，沒有無參數的建構式

Other的建構式中呼叫父類別中無參數建構式,所以無法編譯！

  - 上例要如何解？

Java 繼承語法細節

- 再看 final 關鍵字
  - 在指定變數值之後，就不想再改變變數值，可以在宣告變數時加上final限定。
  - 如果後續撰寫程式時，想修改final變數，就會出現編譯錯誤。
  - 如果物件資料成員被宣告為final，但沒有明確使用=指定值，那表示延遲物件成員值的指定
    - 在建構式執行流程中，一定要有對該資料成員指定值的動作，否則編譯錯誤。
  - class前也可以加上 final 關鍵字
    - 表示這個類別是最後一代，不會再有子類別，不能被繼承。
  - 方法前面也可以加上 final 關鍵字
    - 表示最後一次定義方法，子類別不可以重新定義 final 方法。

Java 繼承語法細節

- java.lang.Object
  - 在Java中，子類別只能繼承一個父類別。
  - 如果定義類別時沒有使用extends關鍵字指定繼承任何類別，那一定是繼承java.lang.Object:

```
public class Some {
   ...
}

//其實 Java 是當作下列程式:
public class Some extends Object {
   ...
}
```

沒定義繼承任何類別！

就當作這支程式是繼承最上層的 Object 類別！

Java 繼承語法細節

- java.lang.Object
  - 在Java中，任何類別追溯至最上層父類別，一定就是java.lang.Object。
  - 以下程式碼正確：

```
Object o1 = "Peter";
Object o2 = new Date();
```

    - String是一種Object，Date是一種Object，任何型態的物件，都可以使用Object宣告的名稱來參考。
  - 如果有個需求是使用陣列收集各種物件，那該宣告為什麼型態呢？

答案是Object[]

```
Object[] objs = {"Monica", new Date(), new SwordsMan()};
String name = (String) objs[0];
Date date = (Date) objs[1];
SwordsMan swordsMan = (SwordsMan) objs[2];
```

Java 繼承語法細節

- java.lang.Object
  - 以下定義的ArrayList類別，則可以不限長度地收集物件:

```
import java.util.Arrays;

public class ArrayList {
    private Object[] list;
    private int next;

    public ArrayList(int capacity) {
        list = new Object[capacity];
    }

    public ArrayList() {
        this(16);
    }
```

```
    public void add(Object o) {
        if(next == list.length) {
            list = Arrays.copyOf(list,
list.length * 2);
        }
        list[next++] = o;
    }

    public Object get(int index) {
        return list[index];
    }

    public int size() {
        return next;
    }
}
```

Java 繼承語法細節

- java.lang.Object
  - 範例：收集訪客名稱，並將名單轉為大寫後顯示！

```
import java.util.Scanner;
import static java.lang.System.out;

public class Guest {
    public static void main(String[] args) {
        ArrayList names = new ArrayList();
        collectNameTo(names);
        out.println("訪客名單：");
        printUpperCase(names);
    }

    static void collectNameTo(ArrayList names) {
        Scanner scanner = new Scanner(System.in);
        String name;
        while(true) {
            out.print("訪客名稱：");
            name = scanner.nextLine();
            if(name.equals("quit")) {
                break;
            }
            names.add(name);
        }
    }
    static void printUpperCase(ArrayList names) {
        for(int i = 0; i < names.size(); i++) {
            String name = (String) names.get(i);
            out.println(name.toUpperCase());
        }
    }
}
```

Java 繼承語法細節

- java.lang.Object
  - 重新定義toString()
    - toString()是Object上定義的方法，Object的toString()預設定義為:

```
public String toString() {
        return getClass().getName() + "@" +
Integer.toHexString(hashCode());
}
```

    - 許多方法若傳入物件，預設都會呼叫toString()
    - 實際上只要這麼撰寫就可以:

```
SwordsMan swordsMan = new SwordsMan();
...略
System.out.println(swordsMan);
```

Java 繼承語法細節

- java.lang.Object
  - 重新定義equals()
    - 比較兩個物件的實質相等性，並不是使用==，而是透過equals()方法
    - 實際上equals()方法是Object類別就有定義的方法，其程式碼實作是:

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

    - 如果沒有重新定義equals()，使用equals()方法時，作用等同於==，所以要比較實質相等性，必須自行重新定義。

Java 繼承語法細節

- java.lang.Object
  - 重新定義equals()
    - 範例:

```
public class Cat {
    ...略過....
    public boolean equals(Object other) {
        // other參考的就是這個物件，當然是同一物件
        if (this == other) {
            return true;
        }
        /* other參考的物件是不是Cat建構出來的
            例如:若是Dog建構出來的當然就不用比了 */
        if (other instanceof Cat) {
            Cat cat = (Cat) other;
            // 定義如果名稱與生日，表示兩個物件實質上相等
            return getName().equals(cat.getName()) && getBirthday().equals(cat.getBirthday());
        }
        return false;
    }
}
```

只要左運算元型態是右運算元型態的子類型，instanceof也是傳回true。

Java 繼承語法細節

- java.lang.Object
  - 重新定義equals()
    - 範例:

```
public class Cat {
    ...略過....
    public boolean equals(Object other) {
        // other參考的就是這個物件，當然是同一物件
        if (this == other) {
            return true;
        }
        /* other參考的物件是不是Cat建構出來的
            例如:若是Dog建構出來的當然就不用比了 */
        if (other instanceof Cat) {
            Cat cat = (Cat) other;
            // 定義如果名稱與生日，表示兩個物件實質上相等
            return getName().equals(cat.getName()) && getBirthday().equals(cat.getBirthday());
        }
        return false;
    }
}
```

只要左運算元型態是右運算元型態的子類型，instanceof也是傳回true。

Java 繼承語法細節

- java.lang.Object
  - 重新定義equals()
    - 如果getName()或getBirthday()傳回null的話，那麼就會噴出NullPointerException, 範例：

```
import static java.util.Objects.equals;
public class Cat {
    ...略...
    public boolean equals(Object other) {
        // other參考的就是這個物件，當然是同一物件
        if (this == other) {
            return true;
        }
        /* other參考的物件是不是Cat建構出來的
            例如若是Dog建構出來的當然就不用比了 */
        if (other instanceof Cat) {
            Cat cat = (Cat) other;
            // 定義如果名稱與生日，表示兩個物件實質上相等
            return equals(getName(), cat.getName()) && equals(getBirthday(),
cat.getBirthday());
        }
        return false;
    }
}
```

Objects.equals()可以協助檢查是否為null的程式碼

Java 繼承語法細節

- java.lang.Object
  - 重新定義equals()
    - 實作equals()時通常也會實作hashCode()
    - 等到學習Collection時再說明

Java 繼承語法細節

- 再看抽象類別
  - 開發一個猜數字遊戲
    - 會隨機產生0到9的數字
    - 使用者輸入的數字與隨機產生的數字相比
      - 如果相同就顯示「猜中了
      - 如果不同就繼續讓使用者輸入數字，直到猜中為止。
    - 範例：

應該不難，初學者應該立馬想得出來！

```
import java.util.Scanner;

public class Guess {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int number = (int) (Math.random() * 10);
        int guess;
        do {
            System.out.print("輸入數字：");
            guess = scanner.nextInt();
        } while(guess != number);
        System.out.println("猜中了");
    }
}
```

Java 繼承語法細節

- 再看抽象類別
  - 開發一個猜數字遊戲
    - 但你的老闆不滿意呀！
    - 「也許在文字模式下執行，也許會用視窗程式，不過改成網頁也不錯」。
    - 設過設計（Design）來解決：
      - 如果，取得使用者輸入、顯示結果的環境未定:

```
import java.util.Scanner;

public abstract class GuessGame {
    public void go() {
        int number = (int) (Math.random() * 10);
        int guess;
        do {
            print("輸入數字：");
            guess = nextInt();
        } while(guess != number);
        println("猜中了");
    }
```

還有下頁！

Java 繼承語法細節

- 再看抽象類別
  - 開發一個猜數字遊戲

```
public void println(String text) {
        print(text + "\n");
    }

    public abstract void print(String text);
    public abstract int nextInt();
}
```

  - 這個類別的定義不完整，print()與nextInt()都是抽象方法！
  - 可以先實作的是猜數字的流程。
  - 如果，還是在文字模式下執行猜數字遊戲，就再撰寫ConsoleGame 類別，繼承抽象類別GuessGame，實作當中的抽象方法即可：

Java 繼承語法細節

- 再看抽象類別
  - 開發一個猜數字遊戲

```
import java.util.Scanner;

public class ConsoleGame extends GuessGame {
    private Scanner scanner = new
Scanner(System.in);

    @Override
    public void print(String text) {
        System.out.print(text);
    }
    @Override
    public int nextInt() {
        return scanner.nextInt();
    }
}
```

```
public class Guess {
    public static void main(String[] args) {
        GuessGame game = new ConsoleGame();
        game.go();
    }
}
```