物件導向軟體工程

(OBJECT-ORIENTED SOFTWARE ENGINEERING)

Homework 1

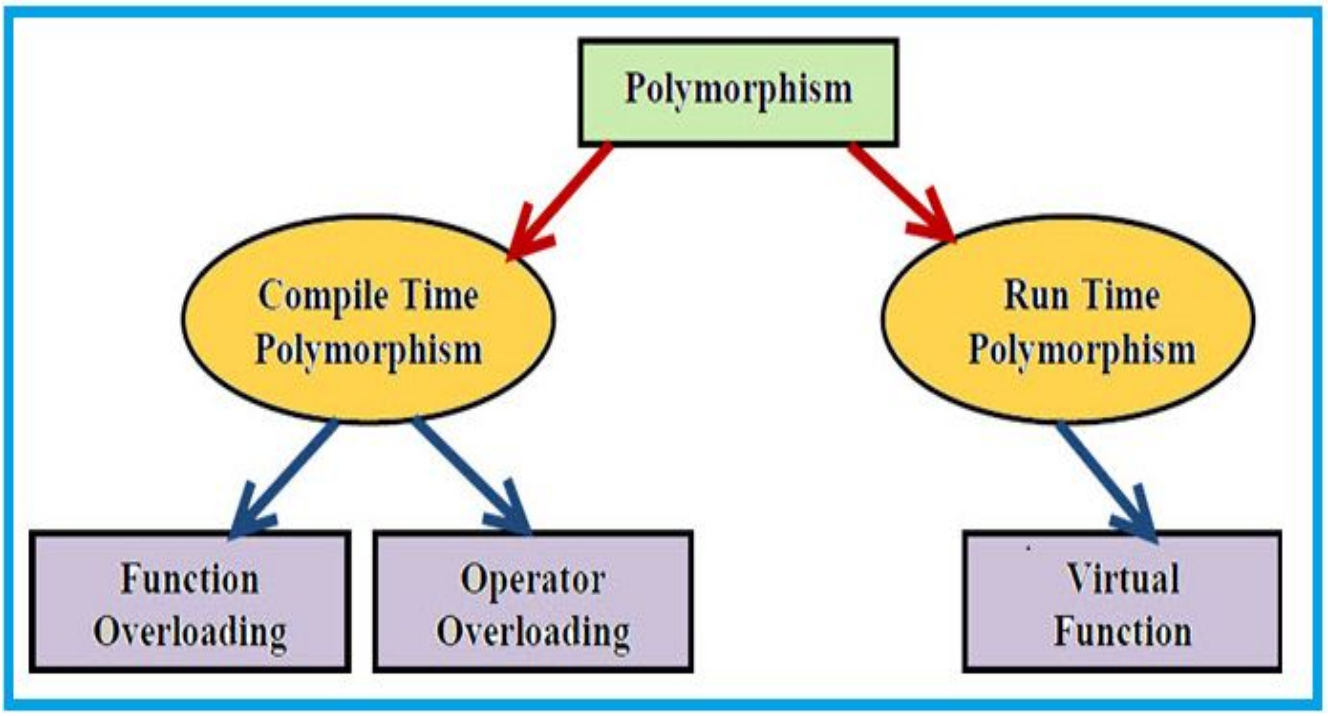
日期:2017/10/05
學號:P76064538
姓名:簡君聿

多型 (Polymorphism)

簡介:

多型操作指的是使用同一操作介面，以操作不同的物件實例(A same operation can behave differently)。

主要可分為兩種，靜態多型 (static polymorphism) 與動態多型 (dynamic polymorphism)，也就是 Early Binding (Static Binding) Compile time 和 Late Binding (Dynamic Binding) Run time。



Early Binding (function overloading)

重載可以對多個類型的參數進行調用(an invocation can be operated on arguments of more than one type)

Ex:

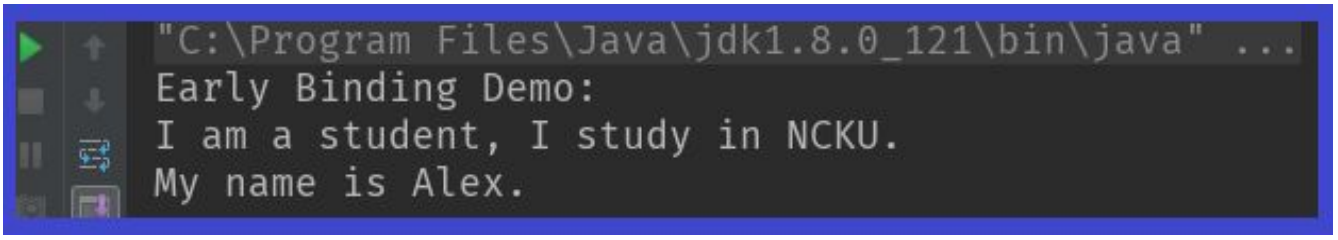
主程式 Main Function

```java
package edu.ncku;

/**
 * This is a Demo.
 *
 * @version 1.0 2017年10月03日
 * @author Alex
 *
 */
public class Main {
    /**
     * Main Function.
     * @param args system default
     */
    public static void main(String[] args) {
        System.out.println("Early Binding Demo:");
        Student studentStaticPolymorphism = new Student( schoolName: "NCKU");
        studentStaticPolymorphism.introduction();
        studentStaticPolymorphism.introduction( studentName: "Alex");
```

處理邏輯 Logic

```java
    // 以下兩個 function 比較表示出靜態多型，introduction function 實作動態多型
    /**
     * Introduction myself.
     */
    void introduction() {
        System.out.println("I am a student, I study in " + schoolName + ".");
    }

    /**
     * Introduction myself.
     * @param studentName student name
     */
    void introduction(String studentName) {
        System.out.println("My name is " + studentName + ".");
    }
```

輸出結果 Output



Early Binding (operator overloading)

Java doesn't support operator overloading.

Late Binding (through overriding)

覆蓋，某個父類別對物件的定義加以擴充，而制訂出一個新的子類別定義，子類別可以繼承父類別原來的某些定義，並也可能增加原來的父類別所沒有的定義，或者是重新定義父類別中的某些特性。

以下是我所整理之步驟:
-> 繼承(inheritance)父類別的型態
-> 接受子類別的物件(overriding)
-> 做相同的行為或相同的資訊
-> 引發不同的行為資訊

Ex:

主程式 Main Function

```
21          System.out.println("\nLate Binding Demo:");
22          Teacher teacherDynamicPolymorphism = new Teacher( schoolName: "NCKU");
23          Teacher studentPolymorphism = new Student( schoolName: "NCKU");
24          teacherDynamicPolymorphism.introduction();
25          studentPolymorphism.introduction();
```

Teacher (Student 父類別)

```java
package edu.ncku;

/**
 * Teacher Class.
 *
 * @version 1.0 2017年10月03日
 * @author Alex
 *
 */
class Teacher extends School {
    /**
     * Constructor.
     * @param schoolName school name
     */
    Teacher(String schoolName) {
        super(schoolName);
    }

    // 以下兩個 function 比較表示出靜態多型，introduction function 實作動態多型
    /**
     * Introduction myself.
     */
    void introduction() {
        System.out.println("I am a Teacher, I work at " + schoolName + ".");
    }
```

Student (Teacher 子類別)

```java
package edu.ncku;

/**
 * Student Class.
 *
 * @version 1.0 2017年10月03日
 * @author Alex
 *
 */
class Student extends Teacher implements Sleep, Exercise {
    /**
     * Constructor.
     * @param schoolName school name
     */
    Student(String schoolName) {
        super(schoolName);
    }

    // 以下兩個 function 比較表示出靜態多型，introduction function 實作動態多型
    /**
     * Introduction myself.
     */
    void introduction() {
        System.out.println("I am a student, I study in " + schoolName + ".");
    }
```

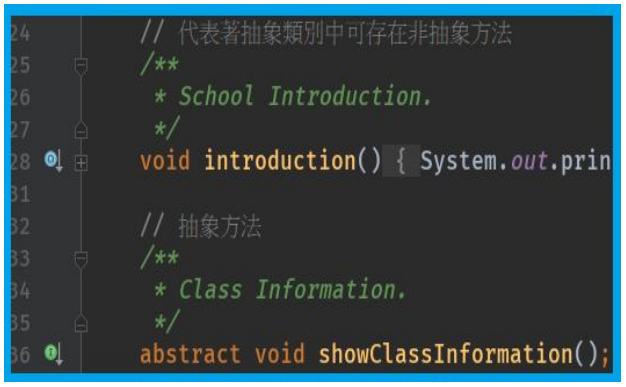
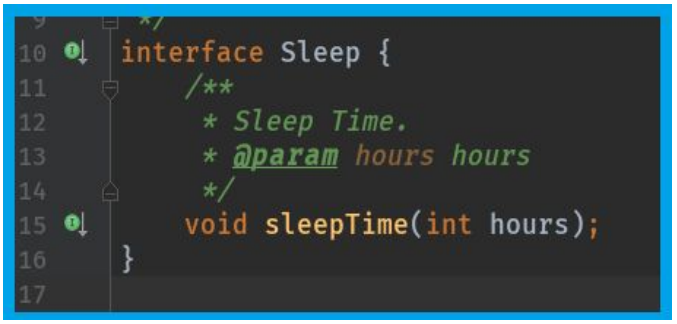
輸出結果 Output

```
Late Binding Demo:
I am a Teacher, I work at NCKU.
I am a student, I study in NCKU.
```

Interface與Abstract 的差別(虛擬類別)

Abstract Class	Interface

創建一個 Abstract Class
 	
`abstract class School {....}`
	創建兩個 Interface
	
	`interface Sleep {....}`
	
	`interface Exercise {....}`

• 只能繼承一個類別 Ex: `class Teacher extends School {....}`	• 可以繼承多個介面 Ex: `class Student extends Teacher implements Sleep, Exercise {....}`
• 可包含非抽象方法  	• 不可包含非抽象方法 
• In relationship, we can say Teacher is School !	• In relationship, we can say Student has Sleep、Exercise (method) !

以下為我所上傳的程式碼：
https://github.com/Alex-CHUN-YU/OOP