S.Rajalakshmi | B.Senthil Kumar | S. Lakshmi Priya SSN COLLEGE OF ENGINEERING Department of Computer Science & Engineering UCS1313: Object Oriented Programming Using Java Lab 2019-2020 Odd - III Semester



Assignment - 4: Polymorphism

===========

Objective:

- 1. To test the following Inheritance type: multiple inheritance.
- 2. To test the Polymorphism through Interface / abstract classes by method overriding.

Design a class called **Person** as described below:

Person -name:String -address:String +Person(name,address) +getName():String +getAddress():String +setAddress(address):void

A sub-class Employee of class Person is designed as shown below:

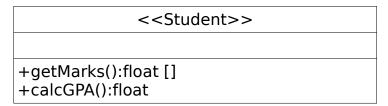
Employee
-empid:String -dept:String -basic:int
+Employee(name,address,empid,dept,basic) +getEmpid():int +getDept():String +setDept(dept):void +setBasic(basic):void +getBasic():int +calSalary():float

A sub-class Faculty of class Employee is designed as shown below:

Faculty
-designation:String -course:String
+Faculty(name,address,empid,dept,basic,desig,course)

+getDesig():String +setDesig(desig):void +setCourse(course):void +getCourse():float +calSalary():float

Design an Interface Student:



Design a sub-class ResearchAssistant of class Employee, implements << Student>>

ResearchAssistant
-project:String -course:String
+ResearchAssitant(name,address,empid,de pt,basic,project,course) +getProject():String +getCourse():String +setCourse(course):void
+getMarks():float []
+calcGPA():float
+calSalary():float

Create a class hierarchy for the following using Interface / Abstract class:

Design **Shape** as described below:

```
Shape

#color:String="red"

+Shape()
+Shape(color)
+getColor():String
+setColor(color):void

abs getArea():float
abs getPerimeter():float
```

Where abs – abstract method

A sub-class **Circle** of class *Shape* is designed as shown below:

Circle	
--------	--

```
#radius:float=1.0

+Circle()
+Circle(radius)
+Circle(radius,color)
+getRadius():float
+setRadius(radius):void
+getArea():float
+getPerimeter():float
```

A sub-class **Rectangle** of class *Shape* is designed as shown below:

Rectangle
#width:float=1.0 #length:float=1.0
+Rectangle() +Rectangle(width,length) +Rectangle(width,length,color) +getWidth():float +setWidth(width):void +getLength():float +setLength(length):void
+getArea():float
+getPerimeter():float

A sub-class **Square** of class *Rectangle* designed as shown below:

Square
+Square() +Square(side) +Square(side,color) +getSide():float +setSide(side):void
+getArea():float +getPerimeter():float

Note the following:

- 1. Shape contains the abstract methods.
- 2. Those abstract methods are to be implemented by the defining classes.

EXERCISE:

- 1. Draw the class diagram of the above class hierarchy.
- 2. Implement the above class hierarchy by using Interface and Abstract class.

Hint:

To write an Interface:

- a. Only abstract methods can be declared inside the Interface.
- b. Identify the common behavior of the set of objects and declare that as abstract methods inside the Interface.
- c. The classes that implement the Interface will provide the actual implementation of those abstract methods.

To write an Abstract class:

- a. An abstract class can have constructor(s), abstract or non-abstract method(s).
- b. Define the constructors and non-abstract method in the Abstract class Shape. Declare the common behavior as the abstract method.
- c. Let the classes Rectangle, Circle, Square define its own constructors, member variable and methods.
- 3. Write a *test driver* called TestInterface | TestAbstract . Use an array of objects of type Shape to display the area, perimeter of all the shapes (Circle, Rectangle, Square).
- 4. Note down the differences while implementing the Inheritance through Interface and Abstract class.
- 5. Note the run-time polymorphism in resolving the method call exhibited by Java through method overriding.