Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110 (An Autonomous Institution, Affiliated to Anna University, Chennai) Department of Computer Science & Engineering UCS2313 – Object Oriented Programming Using Java Lab

Exercise 3. Inheritance

Objective:

- 1. To test the following Inheritance types: single-level, multi-level and hierarchical inheritance.
- 2. To test the scope of private and protected variables, constructors in inherited class hierarchy.

Sample Learning Outcome:

- 1. Need of inheritance and its implementation in Java
- 2. Type of inheritance
- 3. Working of constructors in inherited class
- 4. Accessing inherited class through base class reference
- 5. Method overloading and overriding in inheritance

Best Practices:

- 1. Class Diagram usage
- 2. Naming convention for file names, variables
- 3. Comment usage at proper places
- 4. Prompt messages during reading input and displaying output
- 5. Incremental program development
- 6. Modularity
- 7. All possible test cases in output

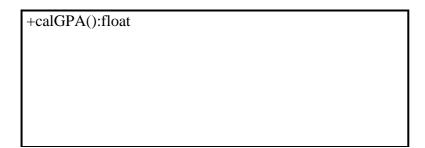
I) Create a class hierarchy for the classes defined below: Design a class called Person as described below:

- Private members
- + Public members
- # Protected members
- ~ Default (Package private)

Person -aadhaar:int -name:String -address:String -gender:char +Person(aadhaar,name,address,gender) +getName():String +getAddress():String +setAddress(address):void +getGender():char

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

Student
-program:String
-year:int
-sub1-grade:char
-sub2-grade:char
-sub3-grade:char
-sub1-credit:int
-sub2-credit:int
-sub3-credit:int
+Student(aadhaar,name,address,gender,program,year, sub1-grade, sub2-grade, sub3-grade, sub1-credit, sub2-credit, sub3-credit) +getProgram():String +getYear():int +setYear(year):void +getsub1-grade():char +getsub2-grade():char +getsub3-grade():int +getsub2-credit():int +getsub3-credit():int +setsub1-grade(char):void +setsub2-grade(char):void
+setsub3-grade(char):void
+setsub1-credit(int):void
+setsub2-credit(int):void +setsub3-credit(int):void
+scisuos-ciculi(iiii).voiu



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

Faculty
-designation:String
-department:String
-basicpay:float
+Faculty(aadhaar,name,address,gender,designation,dept,pay)
+getDesig():String
+setDesig(desig):void
+setBasic(basic):void
+getBasic():float
+calSalary():float

Note the following:

- 1. The hierarchy Person -> Student (or) Person -> Faculty is a *Single-level inheritance* type.
- **2.** The type of above entire class hierarchy (Person -> Student, Person -> Faculty) is the *Hierarchical Inheritance*.
- **3.** Note the use of constructors at all levels of class hierarchy.

EXERCISE: I)

- 1. Draw the class diagram of the above class hierarchy.
- 2. Write a *test driver* called TestInheritance to test all the public methods that display the student and faculty details.

Use the following to calculate Net Salary:

Gross salary = Basicpay + DA as 60% of basic + HRA as 10% of basic

Deductions = Medical Insurance as 8.5% of basic + PF as 8% of basic

Net salary = Gross salary - Deductions

Use the following to calculate GPA

Grade	Point value for the grade
A	5
В	4
C	3
D	2
E	0

Grade point = credit * point value for the grade

GPA = Total points earned / Total credits

II) Create a class hierarchy for the classes as defined below: Design a class Shape as described below: # - protected

Shape
#color:String="red"
+Shape()
+Shape(color)
+getColor():String
+setColor(color):void

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* is designed as shown below:

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

Note the following:

- 1. The hierarchy Shape --> Rectangle --> Square is a *Multi-level inheritance* type.
- 2. The type of above entire class hierarchy is the *Hierarchical Inheritance*.
- 3. Note the constructor overloading at all the levels.
- 4. # denotes protected variable. The protected variables can be accessed by its subclasses and classes in the same package.

EXERCISE: II)

- 1. Draw the class diagram of the above class hierarchy.
- 2. Write a *test driver* called <code>TestShape</code> to test all the <code>public</code> methods. Use an array of objects of type *Shape* and display the area and perimeter of all the shapes (Circle, Rectangle and Square).
- 3. Note down the scope of the variable declared as *protected*.