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 – III: Inheritance
==

Objective:

1. To test the following Inheritance types: single-level, multi-level, hierarchical and hybrid inheritance.

2. To test the scope of private and protected variables, constructors in inherited class hierarchy.

Sample Learning Outcome:

1. Need of inheritance and it's 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

Create a class hierarchy for the classes defined below:

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

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 -totalmark:float
+Student(aadhaar,name,address,gender,program ,year,total) +getProgram():String +getYear():int +setYear(year):void +getTotal():float +setTotal(tot):void +calGPA():float

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

Faculty
-designation:String -department:String -basicpay:float
+Faculty(aadhaar,name,address,gender,designati on,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 is the ***Hierarchical*** *Inheritance*.

3. Note the use of constructors at all levels of class hierarchy.

EXERCISE : 3A

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

%%%%%%%%%%%%%%%%%%###################%%%%%%%%%%%%%%%
%%######

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

+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 : 3B

1. Draw the class diagram of the above class hierarchy.

2. Write a *test driver* called `TestShape` to test all the `public` methods. Display the area and perimeter of all the shapes (Circle, Rectangle and Square).

3. Note down the scope of the variable declared as *protected*.

############$$$$$$$$$$$$$$$$$###################$$$$$$$$$$$$$$$####