**Task 2. Inheritance and Interfaces CO2**

1. **To Implement Method overloading, constructor and Method Overriding.**
2. **Use of this, super, static and final keywords**
3. **Developing user-defined interfaces and implementation**
4. **Use of predefined interfaces**
5. Vinay is working in HR department of XYZ Company. He need to calculate the salary for all employee for the month of August 2022. But Chef Accountant gave a constraint stating that employee those who have basic pay above 30000 need to 15% tax. Vinay feels difficult to do all calculation for each employee manually. So he decided to develop java program to calculate the Salary for each employee by considering following classes and methods with method overloading concept.

Vinay create a class Employee with constructor that initialize the name and Id of the employee, the he created two methods with same name Calculate\_salary (Method overloading) with different number of parameters. First method carries one parameter basic pay alone. Second method carries two parameter basic pay and tax. Based on the basic pay entered by vinay appropriate method need to be invoked.

**Input:**

Enter the Employee id

201

Enter the Employee Name

Rolex

Enter the Basic Salary

45000

**Output:**

Employee Id is: 201

Employee Name is: Rolex

Employee Salary is: 51637.5

Algorithm:

Step 1: Create a class Employee with constructor that initialize Employee name and Employee ID

Step 2: Create Calculate salary method with basic pay as a parameter

Step 3: Calculate HRA which is 10% of basic pay

Step 4: Calculate DA which is 25% of basic pay

Step 5: Calculate Gross Salary by adding HRA, DA and basic pay

Step 6: Since salary is less than 30000 netsalary is same as Gross Salary

Step 7: Create one more Calculate salary method with basic pay and tax as a parameter

Step 8: Repeat step 3, 4 and 5

Step 9: Calculate tax amount which is 15% of basic pay

Step 10: calculate netsalary by subtracting tax amount from Gross salary

Program:

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** Employee {  **int** empid;  String name;  **double** DA;  **double** HRA;  **double** tax;  **double** GS;  **double** netsalary;  **public** Employee(**int** empid, String name) {  **this**.empid=empid;  **this**.name=name;  }    **void** calculatesalary(**double** basic) {  HRA=basic\*(0.1);  DA=basic\*(0.25);  GS=basic+DA+HRA;  netsalary=GS;  }  **void** calculatesalary( **double** basic,**double** taxper) {  HRA=(0.1)\*basic;  DA=(0.25)\*basic;  GS=basic+DA+HRA;  tax=(taxper)\*GS;  netsalary=GS-tax;  }  **void** display()  {  System.***out***.println("Employee Id is: "+empid);  System.***out***.println("Employee Name is: "+name);  System.***out***.println("Employee Salary is: "+netsalary);    }  **public** **static** **void** main(String args[])  {    Scanner sr=**new** Scanner(System.***in***);  System.***out***.println("Enter the Employee id");  **int** empid=sr.nextInt();  System.***out***.println("Enter the Employee Name");  String name=sr.next();  Employee obj=**new** Employee(empid,name);  System.***out***.println("Enter the Basic Salary");  **double** basic=sr.nextInt();  **if**(basic>30000)  {  obj.calculatesalary(basic,0.15);  }  **else**  {  obj.calculatesalary(basic);  }  obj.display();  sr.close();  }  } |

1. Implement a class Car as a parent class with constructor that initialize speed, mileage and number of airbags. Create a display method that prints maximum of speed of the car, Mileage of the car and Number of airbags it contains. Now create a child class Alto that implements the constructor by initialize car model and brand as instance variable. Use super keyword to access the parent class constructor for initializing the speed, mileage and airbag details. Create Display method in child class that prints the Car model and brand before that access the parent class display method to print the car speed, mileage and number of airbags.

Output:

Speed of the car is: 90

Milage of the car is: 25

No.of Airbags of the car is: 1

Car Model is:2019

Car brand is:Maruti

Algorithm:

Step 1: Create parent class Car with constructor to initialize the speed, mileage and Airbags instance variable

Step 2: Create display to display instance variable value

Step3: create child class Alto that extends the parent class car

Step 4: create constructor in child class to initialize model and brand of the car

Step 5: Use super method to access the parent class constructor

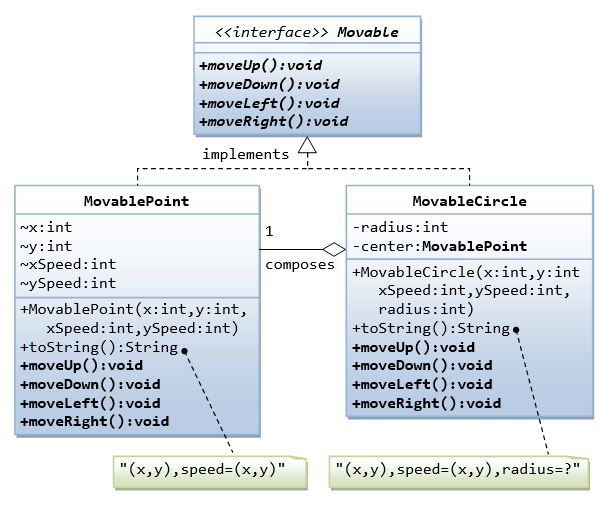
Step 6: create display method in child class to print the model and brand of the car

Step 7: use Super method to access parent class display method to print speed, mileage and number of airbags.

Program:

|  |
| --- |
| // Create a class file and name it as car.java  **public** **class** car {  **public** **int** speed,milage,airbags;  **public** car(**int** speed, **int** milage, **int** airbags)  {  **this**.speed=speed;  **this**.milage=milage;  **this**.airbags=airbags;  }  **public** **void** display()  {  System.***out***.println("Speed of the car is: "+**this**.speed);  System.***out***.println("Milage of the car is: "+**this**.milage);  System.***out***.println("No.of Airbags of the car is: "+**this**.airbags);  }  }  //Create a class file and name it as alto.java  **public** **class** alto **extends** car {  **public** **int** model;  **public** String brand;  **public** alto(String brand,**int** model,**int** milage,**int** speed,**int** airbags)  {  **super**(speed,milage, airbags);  **this**.model=model;  **this**.brand=brand;  }  **public** **void** display()  {  **super**.display();  System.***out***.println("Car Model is:"+**this**.model);  System.***out***.println("Car brand is:"+**this**.brand);  }  **public** **static** **void** main(String[] args) {  alto a=**new** alto("Maruti", 2019,25,90,1);  a.display();  }  } |

1. Write an interface called Movaable, which contains 4 abstract methods moveUp(), moveDown(), moveLeft() and moveRight(), as shown in the class diagram. Also write the implementation classes called MovablePoint and MovableCircle. Mark all the overridden methods with annotation @Override.



Output:

MovablePoint (5, 6) with xSpeed = 10 and ySpeed = 10

MovablePoint (-5, 6) with xSpeed = 10 and ySpeed = 10

MovableCircle at point MovablePoint (2, 1) with xSpeed = 2 and ySpeed = 2 with radius = 20

MovableCircle at point MovablePoint (4, 1) with xSpeed = 2 and ySpeed = 2 with radius = 20

**Program:**

|  |
| --- |
| //movable.java  **public** **interface** movable {  **public** **void** moveUp();  **public** **void** moveDown();  **public** **void** moveLeft();  **public** **void** moveRight();  }  //MovablePoint.java  **public** **class** MovablePoint **implements** movable {    **int** x, y, xSpeed, ySpeed;  **public** MovablePoint(**int** x, **int** y, **int** xSpeed, **int** ySpeed) {  **this**.x = x;  **this**.y = y;  **this**.xSpeed = xSpeed;  **this**.ySpeed = ySpeed;  }  @Override  **public** **void** moveUp() {  y -= ySpeed;  }  @Override  **public** **void** moveDown() {  y += ySpeed;  }  @Override  **public** **void** moveLeft() {  x -= xSpeed;  }  @Override  **public** **void** moveRight() {  x += xSpeed;  }  @Override  **public** String toString() {  **return** String.*format*("MovablePoint (%1$d, %2$d) with xSpeed = %3$d and ySpeed = %4$d"  , x, y, xSpeed, ySpeed);  }  }  //MovableCircle.java  **public** **class** MovableCircle **implements** movable{  **private** MovablePoint center;  **private** **int** radius;  **public** MovableCircle(**int** x, **int** y, **int** xSpeed, **int** ySpeed, **int** radius) {  center = **new** MovablePoint(x, y, xSpeed, ySpeed);  **this**.radius = radius;  }  **public** **void** moveUp() {  center.y -= center.ySpeed;  }  **public** **void** moveDown() {  center.y += center.ySpeed;  }  **public** **void** moveLeft() {  center.x -= center.xSpeed;  }  **public** **void** moveRight() {  center.x += center.xSpeed;  }    **public** String toString() {  **return** String.*format*("MovableCircle at point %1$s with radius = %2$d"  , center.toString(), radius);  }  }  //TestMovableCircle.java  **public** **class** TestMovableCircle {  **public** **static** **void** main(String[] args)  {  movable m1 = **new** MovablePoint(5, 6, 10, 10);  System.***out***.println(m1);  m1.moveLeft();  System.***out***.println(m1);    movable m2 = **new** MovableCircle(2, 1, 2, 2, 20);  System.***out***.println(m2);  m2.moveRight();  System.***out***.println(m2);  }  } |