**Java Lab 4**

**Exercise-2**

Let's analyze and discuss the provided Java code for Exercise 02:

The class SuperB:

It has an instance variable x.

It has four methods:

setIt(int n): sets the value of x to the given integer n.

increase(): increments the value of x by 1.

triple(): multiplies the value of x by 3.

returnIt(): returns the current value of x.

The class SubC:

It extends the SuperB class, meaning it inherits all the fields and methods from SuperB.

It overrides the triple() method from the SuperB class. Instead of tripling x, it adds 3 to the current value of x.

The class TestInheritance:

It contains the main method, where we test the behavior of the SuperB and SubC classes.

Let's walk through the execution step by step:

SuperB b = new SuperB();: We create an object b of the SuperB class.

b.setIt(2);: We set the value of x to 2 using the setIt method.

b.increase();: We increase the value of x by 1 using the increase method. Now, x is 3.

b.triple();: We triple the value of x using the triple method. Now, x becomes 9.

System.out.println(b.returnIt());: We print the current value of x, which is 9.

Output for the above part of the code: 9

Next, we create an object c of the SubC class and repeat the steps:

SubC c = new SubC();: We create an object c of the SubC class, which also has access to the methods of the SuperB class.

c.setIt(2);: We set the value of x to 2 using the setIt method. Now, x is 2.

c.increase();: We increase the value of x by 1 using the increase method. Now, x is 3.

c.triple();: Here, we use the overridden triple method from the SubC class. Instead of tripling, we add 3 to the current value of x. Now, x becomes 6.

System.out.println(c.returnIt());: We print the current value of x, which is 6.

Output for this part of the code: 6

So, when we run the entire TestInheritance class, we will get the following output:

9

6

This output confirms that the method overriding works as expected, and objects of the subclass can have different behaviors than objects of the superclass, even if they share the same method names.

**Exercise-3**

Person.java:

java

Copy code

public class Person {

// Common attributes for Student and Lecturer

private String name;

private int id;

// Getters and setters for name

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// Getters and setters for id

public int getID() {

return id;

}

public void setID(int id) {

this.id = id;

}

}

Student.java:

java

Copy code

public class Student extends Person {

// Specific attribute for Student

private String course;

// Getter and setter for course

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

}

Lecturer.java:

java

Copy code

public class Lecturer extends Person {

// Specific attribute for Lecturer

private String programme;

// Getter and setter for programme

public String getProg() {

return programme;

}

public void setProg(String programme) {

this.programme = programme;

}

}

TestPerson.java:

java

Copy code

public class TestPerson {

public static void main(String[] args) {

Student student = new Student();

student.setName("John Doe");

student.setID(12345);

student.setCourse("Computer Science");

Lecturer lecturer = new Lecturer();

lecturer.setName("Dr. Jane Smith");

lecturer.setID(98765);

lecturer.setProg("Data Science");

System.out.println("Student Name: " + student.getName());

System.out.println("Student ID: " + student.getID());

System.out.println("Student Course: " + student.getCourse());

System.out.println("Lecturer Name: " + lecturer.getName());

System.out.println("Lecturer ID: " + lecturer.getID());

System.out.println("Lecturer Programme: " + lecturer.getProg());

}

}

Exercise 04: Java code to discuss the output of the provided code:

In the given code, we have three classes: Animal, Mammal, and Reptile. The Dog class extends the Mammal class and contains the main method to test some instanceof conditions.

When we run the Dog class, the output will be:

arduino

Copy code

true

true

true

Explanation:

m instanceof Animal: This checks whether the m object (which is of type Mammal) is an instance of the Animal class. Since Mammal is a subclass of Animal, the result is true.

d instanceof Mammal: This checks whether the d object (which is of type Dog) is an instance of the Mammal class. Since Dog is a subclass of Mammal, the result is true.

d instanceof Animal: This checks whether the d object (which is of type Dog) is an instance of the Animal class. Since Dog is a subclass of Mammal, and Mammal is a subclass of Animal, the result is true.

The output confirms that the instanceof operator returns true for all three conditions, demonstrating the inheritance relationship between the classes.