Exercise 01:

Create a class called “Employee” which has 3 private variables (empID, empName, empDesignation) and create getters and setters for each field. Please note that this has no main method since this is just a blueprint not a application. Now crate a test class to invoke the Employee class. Create two objects for Mr.Bogdan and Ms.Bird and set required values using setters and print them back on the console using getters.

**class Employee {**

**private int empID;**

**private String empName;**

**private String empDesignation;**

**public int getEmpID() {**

**return empID;**

**}**

**public void setEmpID(int empID) {**

**this.empID = empID;**

**}**

**public String getEmpName() {**

**return empName;**

**}**

**public void setEmpName(String empName) {**

**this.empName = empName;**

**}**

**public String getEmpDesignation() {**

**return empDesignation;**

**}**

**public void setEmpDesignation(String empDesignation) {**

**this.empDesignation = empDesignation;**

**}**

**}**

**public class TestEmployee {**

**public static void main(String[] args) {**

**Employee bogdan = new Employee();**

**bogdan.setEmpID(101);**

**bogdan.setEmpName("Mr. Bogdan");**

**bogdan.setEmpDesignation("Software Engineer");**

**Employee bird = new Employee();**

**bird.setEmpID(102);**

**bird.setEmpName("Ms. Bird");**

**bird.setEmpDesignation("Graphic Designer");**

**System.out.println("Employee 1 Details:");**

**System.out.println("ID: " + bogdan.getEmpID());**

**System.out.println("Name: " + bogdan.getEmpName());**

**System.out.println("Designation: " + bogdan.getEmpDesignation());**

**System.out.println();**

**System.out.println("Employee 2 Details:");**

**System.out.println("ID: " + bird.getEmpID());**

**System.out.println("Name: " + bird.getEmpName());**

**System.out.println("Designation: " + bird.getEmpDesignation());**

**}**

**}**

Exercise 02:

Develop the following class execute and discuss the answer: Please note that each class stored in separate files. Write down the answer.

class SuperB {

int x;

void setIt (int n) { x=n;}

void increase () { x=x+1;}

void triple () {x=x\*3;};

int returnIt () {return x;}

}

class SubC extends SuperB {

void triple () {x=x+3;} // override existing method

void quadruple () {x=x\*4;} // new method

}

public class TestInheritance {

public static void main(String[] args) {

SuperB b = new SuperB();

b.setIt(2);

b.increase();

b.triple();

System.out.println( b.returnIt() );

SubC c = new SubC();

c.setIt(2);

c.increase();

c.triple();

System.out.println( c.returnIt() ); }

}

**The provided code defines two classes, SuperB and SubC, along with a test class TestInheritance. Let's walk through the code step by step and discuss the output.**

1. **The SuperB class has an instance variable x and several methods to modify and return its value.**
2. **The SubC class extends SuperB and overrides the triple() method to add 3 to the value of x instead of multiplying it by 3. It also has a new method quadruple() to multiply x by 4.**
3. **The TestInheritance class is used to demonstrate the behavior of these classes.**

**Here's the output you would get when you run the TestInheritance class:**

**18**

**13**

**Explanation:**

**In the first part, we create an instance of SuperB named b. We set its x to 2, then increase it by 1 using the increase() method (resulting in 3), and finally, the triple() method multiplies it by 3 (resulting in 9). The returnIt() method then returns the value 9.**

**In the second part, we create an instance of SubC named c. We follow the same steps as before: set x to 2, increase by 1, and apply the overridden triple() method which adds 3 (resulting in 6). The returnIt() method then returns the value 6.**

**The triple() method is overridden in SubC, so the behavior is different from the SuperB class. The output reflects the operations performed on the respective instances of the two classes.**

Exercise 03:

Recall the following scenario discussed during the class. Develop a code base to represent the scenario. Add a test class to invoke Lecturer and Student class by creating atleast one object from each.

Note: All the common attributes and behavior stored in the super class and only the specific fields and behavior stored in subclasses.

|  |
| --- |
| Student |
| * name |
| * id |
| * course |
| + setName()/getName() |
| + setID()/getID() |
| + setCourse()/getCourse() |

|  |
| --- |
| Lecturer |
| * name |
| * id |
| * programme |
| + setName()/getName() |
| + setID()/getID() |
| + setProg()/getProg() |

|  |
| --- |
| Person |
| Identify field and attributes to be stored in this class |

**class Person {**

**private String name;**

**private int id;**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**}**

**class Student extends Person {**

**private String course;**

**public String getCourse() {**

**return course;**

**}**

**public void setCourse(String course) {**

**this.course = course;**

**}**

**}**

**class Lecturer extends Person {**

**private String programme;**

**public String getProgramme() {**

**return programme;**

**}**

**public void setProgramme(String programme) {**

**this.programme = programme;**

**}**

**}**

**public class TestPersonInheritance {**

**public static void main(String[] args) {**

**Student student = new Student();**

**student.setName("John Doe");**

**student.setId(101);**

**student.setCourse("Computer Science");**

**System.out.println("Student Details:");**

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

**System.out.println("ID: " + student.getId());**

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

**System.out.println();**

**Lecturer lecturer = new Lecturer();**

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

**lecturer.setId(201);**

**lecturer.setProgramme("Software Engineering");**

**System.out.println("Lecturer Details:");**

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

**System.out.println("ID: " + lecturer.getId());**

**System.out.println("Programme: " + lecturer.getProgramme());**

**}**

**}**

Exercise 04

Develop the following class execute and discuss the answer: Please note that each public class stored in separate files. Write down the answer.

public class Animal{}

public class Mammal extends Animal{}

public class Reptile extends Animal{}

public class Dog extends Mammal{

public static void main(String args[]){

Animal a = new Animal();

Mammal m = new Mammal();

Dog d = new Dog();

System.out.println(m instanceof Animal);

System.out.println(d instanceof Mammal);

System.out.println(d instanceof Animal);

}

}

**The provided code demonstrates inheritance relationships using classes in Java. Let's discuss the code and its output.**

1. **The Animal class is a base class with no specific attributes or methods defined.**
2. **The Mammal class is a subclass of Animal.**
3. **The Reptile class is another subclass of Animal.**
4. **The Dog class is a subclass of Mammal.**

**Here's the output you would get when you run the Dog class:**

**false**

**true**

**true**

**Explanation:**

**In the main method:**

**a is an instance of Animal. Since Animal is the base class, the instanceof check for m instanceof Animal returns false because an instance of Mammal is not an instance of the base class Animal.**

**m is an instance of Mammal. The instanceof check for d instanceof Mammal returns true because Dog is a subclass of Mammal.**

**d is an instance of Dog. The instanceof check for d instanceof Animal returns true because Dog is a subclass of Mammal, which is a subclass of Animal.**

**The output reflects the hierarchy and inheritance relationships among the classes.**