Practical 04: Encapsulation & Inheritance

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.

Answers

```
public class Employee {

    private int empID;

    private String empName;

    private String empDesignation;


    // Getter for empID

    public int getEmpID() {

        return empID;

    }


    // Setter for empID

    public void setEmpID(int empID) {

        this.empID = empID;

    }


    // Getter for empName

    public String getEmpName() {

        return empName;
```

```
    }

    // Setter for empName
    public void setEmpName(String empName) {
        this.empName = empName;
    }

    // Getter for empDesignation
    public String getEmpDesignation() {
        return empDesignation;
    }

    // Setter for empDesignation
    public void setEmpDesignation(String empDesignation) {
        this.empDesignation = empDesignation;
    }
}
public class TestEmployee {
    public static void main(String[] args) {
        // Create two Employee objects
        Employee mrBogdan = new Employee();
        Employee msBird = new Employee();

        // Set values for Mr. Bogdan using setters
```

```
        mrBogdan.setEmpID(101);
        mrBogdan.setEmpName("Mr. Bogdan");
        mrBogdan.setEmpDesignation("Software Engineer");

        // Set values for Ms. Bird using setters
        msBird.setEmpID(202);
        msBird.setEmpName("Ms. Bird");
        msBird.setEmpDesignation("Project Manager");

        // Print the details of Mr. Bogdan using getters
        System.out.println("Employee ID: " + mrBogdan.getEmpID());
        System.out.println("Employee Name: " + mrBogdan.getEmpName());
        System.out.println("Employee Designation: " + mrBogdan.getEmpDesignation());

        // Print the details of Ms. Bird using getters
        System.out.println("Employee ID: " + msBird.getEmpID());
        System.out.println("Employee Name: " + msBird.getEmpName());
        System.out.println("Employee Designation: " + msBird.getEmpDesignation());
    }
}
```

Practical 04: Encapsulation & Inheritance

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() ); }
```

}

Answers or explanation

1. For the '**SuperB'** object 'b':

- **'b.setIt(2)'** sets **'x'** to 2.
- **'b. increase()**' increases **'x'** by 1, making it 3.
- **'b.triple()'** triples the value of **'x'**, making it 9.
- **'System.out.println(b.returnIt())**' prints the value of '**x'**, which is 9.

2. For the **'SubC'** object **'c'**:

- **'c.setIt(2)'** sets **'x'** to 2.
- **'c.increase()**' increases **'x'** by 1, making it 3.
- **'c.triple()'** (overridden method) adds 3 to **'x'**, making it 6.
- **'System.out.println(c.returnIt())**' prints the value of **'x',** which is 6.

3.TestInheritance class:

- It has the **'main'** method, where we create objects of **'SuperB'** and **'SubC'**.
- We first create a **'SuperB'** object **'b'**, set its value to 2, increase it by 1, and then triple it (multiply by 3). So, **'x'** will be (2+1) * 3 = 9.
- We print the value of **'x'** using the **'returnIt()'** method of **'SuperB'**, which will print 9.
- Next, we create a **'SubC'** object **'c',** set its value to 2, increase it by 1, and then triple it (add 3). So, **'x'** will be (2+1) + 3 = 6.
- We print the value of **'x'** using the **'returnIt()'** method of **'SubC'**, which will print 6.

Practical 04: Encapsulation & Inheritance

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

Answers

//person class

```
public class Person {

   private String name;

   private int id;

   // Getters and Setters for name and 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;
    }
}
```

//Student class

```
public class Student extends Person {
    private String course;
    // Getter and Setter for course
    public String getCourse() {
        return course;
    }
    public void setCourse(String course) {
        this.course = course;
    }
}
```

//lecturer

```
public class Lecturer extends Person {
    private String programme;

    // Getter and Setter for programme
```

```
    public String getProgramme() {
        return programme;
    }
    public void setProgramme(String programme) {
        this.programme = programme;
    }
}
```

//test person class (student and lecturer)

```
public class TestPerson {
    public static void main(String[] args) {
        // Create a Student object
        Student student = new Student();
        student.setName("Mohamed rila");
        student.setID(27868);
        student.setCourse("Software engineering");

        // Create a Lecturer object
        Lecturer lecturer = new Lecturer();
        lecturer.setName("Mr.shafraz");
        lecturer.setID(1234);
        lecturer.setProgramme("Java devoloper");
```

```
        // Print student details
        System.out.println("Student Name: " + student.getName());
        System.out.println("Student ID: " + student.getID());
        System.out.println("Student Course: " + student.getCourse());

        // Print lecturer details
        System.out.println("\nLecturer Name: " + lecturer.getName());
        System.out.println("Lecturer ID: " + lecturer.getID());
        System.out.println("Lecturer 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);
  }
}
```

Answers

1.We have four classes: **'Animal'**, **'Mammal'**, **'Reptile'**, and **'Dog'**. Each class is stored in separate files.

2**.'Animal'** class:

- It is a superclass that doesn't have any specific fields or methods in this example.

3.'**Mammal**' class:

- It extends the **'Animal'** class, which means it is a subclass of **'Animal'**.
- As 'Mammal' is a subclass of **'Animal'**, it inherits all the members (fields and methods) of **'Animal'.**

4.**'Reptile**' class:

- It also extends the **'Animal'** class and inherits its members.

5.**'Dog'** class:

- It extends the **'Mammal'** class, which means it is a subclass of both **'Mammal'** and **'Animal'**.

- As a subclass of **'Mammal'**, it inherits all the members (fields and methods) of **'Mammal'**, including those inherited from **'Animal'**.

6.In the **'main'** method of '**Dog**' class, we create three objects of different classes: '**Animal', 'Mammal'**, and **'Dog'**.

7.We then use the **'instanceof'** operator to check whether each object is an instance of a particular class:

- **'m instanceof Animal'**: As **'m**' is an object of **'Mammal'** class, which extends **'Animal'**, this will be true.
- **'d instanceof Mammal'**: As **'d'** is an object of **'Dog'** class, which extends **'Mammal'**, this will also be true.
- **'d instanceof Animal'**: As **'d'** is an object of **'Dog'** class, which extends **'Mammal'**, and **'Mammal'** extends **'Animal'**, this will be true as well

s