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.

```
package com.mycompany.prc4;

public 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 settempDesignation(String tempDesignation){
   this.empDesignation=tempDesignation;
}
}
//Create new class
package com.mycompany.prc4;
public class testclass {
   public static void main(String[] args) {
      Employee MrBogdan =new Employee();
       Employee MsBird  =new Employee();

       MrBogdan.setEmpID(12);
       MrBogdan.setEmpName("vimu");
       MrBogdan.settempDesignation("softwer Engenner");


       MsBird.setEmpID(13);
       MrBogdan.setEmpName("vimu");
       MrBogdan.settempDesignation("softwer Engenner");
   }

}
```

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() ); }
```

}

Output : 9

5

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

//Person Class:

```
public class Person {
 private String name;
   private int id;
   public Person() {
   }
   public Person(String name, int id) {
      this.name = name;
      this.id = 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;
    public Student() {
    }
    public Student(String name, int id, String course) {
        super(name, id);
        this.course = course;
    }
    public String getCourse() {
        return course;
    }
    public void setCourse(String course) {
        this.course = course;
```

```
    }
}
//Lecturer Class
public class Lecturer extends Person {
    private String programme;
    public Lecturer() {
    }
    public Lecturer(String name, int id, String programme) {
        super(name, id);
        this.programme = programme;
    }
    public String getProg() {
        return programme;
    }
    public void setProg(String programme) {
        this.programme = programme;
    }
}
//TestClass to Invoke Lecturer and Student Classes
public class TestPerson {
    public static void main(String[] args) {
        // Create a Student object
        Student student = new Student("John Doe", 12345, "Computer Science");
        System.out.println("Student Name: " + student.getName());
        System.out.println("Student ID: " + student.getID());
```

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


        // Create a Lecturer object
        Lecturer lecturer = new Lecturer("Jane Smith", 98765, "Mathematics");
        System.out.println("Lecturer Name: " + lecturer.getName());
        System.out.println("Lecturer ID: " + lecturer.getID());
        System.out.println("Lecturer Programme: " + lecturer.getProg());
    }
}
```

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);
  }
}
```

```
// Animal class
package com.mycompany.dog;
public class Animal {
}
// Mammal class
package com.mycompany.dog;
public class Mammal extends Animal {
}
// Reptile class
package com.mycompany.dog;
public class Reptile extends Animal {
}
//dog class
package com.mycompany.dog;
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);  // Output: true
      System.out.println(d instanceof Mammal);  // Output: true
      System.out.println(d instanceof Animal);  // Output: true
   }
}
```

Practical 04: Encapsulation & Inheritance