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.

Class

```
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 setEmpDesignation(String empDesignation)
    {
        this.empDesignation = empDesignation;
    }
}
```

Main Method

```
public class TestEmployee
 {
    public static void main(String[] args)
   {
         Employee mrBogdan = new Employee();
         mrBogdan.setEmpID(101);
         mrBogdan.setEmpName("Mr. Bogdan");
         mrBogdan.setEmpDesignation("Software Engineer");

         Employee msBird = new Employee();
         msBird.setEmpID(102);
         msBird.setEmpName("Ms. Bird");
         msBird.setEmpDesignation("Project Manager");

         System.out.println("Employee ID: " + mrBogdan.getEmpID());
         System.out.println("Employee Name: " + mrBogdan.getEmpName());
         System.out.println("Employee Designation: " + mrBogdan.getEmpDesignation());

         System.out.println("Employee ID: " + msBird.getEmpID());
         System.out.println("Employee Name: " + msBird.getEmpName());
         System.out.println("Employee Designation: " + msBird.getEmpDesignation());
    }
}
```

Output

Employee ID: 101

Employee Name: Mr. Bogdan

Employee Designation: Software Engineer

Employee ID: 102

Employee Name: Ms. Bird

Employee Designation: Project Manager

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() ); }
```

}

SuperB class:

```
class SuperB
{
    int x;

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

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

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

    int returnIt()
   {
        return x;
    }
}
```

SubC class:

```
class SubC extends SuperB
{
    void triple()
    {
        x = x + 3;
    }

    void quadruple()
    {
        x = x * 4;
    }
}
```

Practical 04: Encapsulation & Inheritance

TestInheritance_class:

```
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:

(2 + 1) * 3 = 9

(2 + 1) + 3 = 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

Person class

```
public class Person
{
    private String name;
    private String id;

    public void setName(String name)
    {
        this.name = name;
    }

    public String getName()
    {
        return name;
    }

    public void setID(String id)
   {
        this.id = id;
    }

    public String getID()
    {
        return id;
    }
}
```

Student class (subclass of Person)

```
public class Student extends Person
 {
    private String course;

    public void setCourse(String course)
    {
        this.course = course;
    }

    public String getCourse()
   {
        return course;
    }
}
```

Lecturer class (subclass of Person)

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

    public void setProg(String programme)
    {
        this.programme = programme;
    }

    public String getProg()
   {
        return programme;
    }
}
```

Practical 04: Encapsulation & Inheritance

Student and Lecturer classes

```
public class TestPerson
{
    public static void main(String[] args)
  {
        Student student = new Student();
        student.setName("Atheeb");
        student.setID("28436");
        student.setCourse("Computer Science");

        Lecturer lecturer = new Lecturer();
        lecturer.setName("Thuvaragan");
        lecturer.setID("28435");
        lecturer.setProg("Computer Engineering");

        System.out.println("Student Information:");
        System.out.println("Name: " + student.getName());
        System.out.println("ID: " + student.getID());
        System.out.println("Course: " + student.getCourse());

        System.out.println("\nLecturer Information:");
        System.out.println("Name: " + lecturer.getName());
        System.out.println("ID: " + lecturer.getID());
        System.out.println("Programme: " + lecturer.getProg());
    }
}
```

Output

Student Information:

Name: Atheeb

ID: 28436

Course: Computer Science

Lecturer Information:

Name: Thuvaragan

ID: 28435

Programme: Computer Engineering

Practical 04: Encapsulation & Inheritance

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

```
public class Animal
{

}
```

Mammal class

```
public class Mammal extends Animal
{

}
```

Practical 04: Encapsulation & Inheritance

Reptile class

```
public class Reptile extends Animal
{

}
```

Dog class

```
public class Dog extends Mammal
{

}
```

main() method in Dog class

```
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);
}
```

Output

true

true

true