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.

```
public class Employee {
   private int empID;
   private String empName;
   private String empDesignation;
   // Getter and Setter for empID
   public int getEmpID() {
      return empID;
   }
    public void setEmpID(int empID) {
      this.empID = empID;
   }
   // Getter and Setter for empName
   public String getEmpName() {
      return empName;
   }
    public void setEmpName(String empName) {
      this.empName = empName;
   }
  // Getter and Setter for empDesignation
   public String getEmpDesignation() {
      return empDesignation;
   }
```

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

Main class

```
public class EmployeeTest {
    public static void main(String[] args) {
        Employee mrBogdan = new Employee();
        mrBogdan.setEmpID(1);
        mrBogdan.setEmpName("Mr. Bogdan");
        mrBogdan.setEmpDesignation("Manager");
        Employee msBird = new Employee();
        msBird.setEmpID(2);
        msBird.setEmpName("Ms. Bird");
        msBird.setEmpDesignation("Engineer");
        // Printing values using getters
        System.out.println("Mr. Bogdan:");
        System.out.println("Employee ID: " + mrBogdan.getEmpID());
        System.out.println("Employee Name: " + mrBogdan.getEmpName());
        System.out.println("Employee Designation: " + mrBogdan.getEmpDesignation());
        System.out.println();
        System.out.println("Ms. Bird:");
        System.out.println("Employee ID: " + msBird.getEmpID());
        System.out.println("Employee Name: " + msBird.getEmpName());
        System.out.println("Employee Designation: " + msBird.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 expected output of the TestInheritance class is:

```
9
6
```

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

```
public class Person {
   private String name;
   private int id;
   public void setName(String name) {
      this.name = name;
   }
   public String getName() {
      return name;
   }
  public void setID(int id) {
      this.id = id;
```

```
    }
  public int getID() {
        return id;
    }
}
public class Student extends Person {
    private String course;
  public void setCourse(String course) {
        this.course = course;
    }
   public String getCourse() {
        return course;
    }
}
public class Lecturer extends Person {
    private String programme;

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

Main class

```
public class TestPerson {
    public static void main(String[] args) {
```

```
        Student student = new Student();
        student.setName("John Doe");
        student.setID(123);
        student.setCourse("Computer Science");

        Lecturer lecturer = new Lecturer();
        lecturer.setName("Jane Smith");
        lecturer.setID(456);
        lecturer.setProg("Data Science");

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

  }

}
```

The expected output of the Dog class is:

```
true

true

true
```