Practical 08

Exercise 01:

Create a class named "BankAccount" with private instance variables "accountNumber" and "balance." Implement encapsulation by providing public getter and setter methods for both variables. Additionally, create an abstract method called "calculateInterest" in the "BankAccount" class. Implement two subclasses, "SavingsAccount" and "CheckingAccount," that extend the "BankAccount" class and provide their own implementations of the "calculateInterest" method. Write the implementation code for the getter and setter methods in the "BankAccount" class, and the "calculateInterest" method in both the "SavingsAccount" and "CheckingAccount" classes. Assuming that the interest for saving is 12% and checking is 2% (both private variables), find out What will be the interest for a person with 1 million in his checking and 20 million in his saving account.

```
abstract class BankAccount {

   private int accountNumber;

   private double balance;


   public int getAccountNumber() {

      return accountNumber;

   }


   public void setAccountNumber(int accountNumber) {

      this.accountNumber = accountNumber;

   }


   public double getBalance() {

      return balance;

   }


   public void setBalance(double balance) {
```

```
        this.balance = balance;
    }

    abstract double calculateInterest();
}

class SavingsAccount extends BankAccount {
    private final double interestRate = 0.12;

    double calculateInterest() {
        return getBalance() * interestRate;
    }
}

class CheckingAccount extends BankAccount {
    private final double interestRate = 0.02;

    double calculateInterest() {
        return getBalance() * interestRate;
    }
}

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

```
        CheckingAccount checking = new CheckingAccount();
        checking.setBalance(1000000);
        double checkingInterest = checking.calculateInterest();
        System.out.println("Checking interest: " + checkingInterest);

        SavingsAccount savings = new SavingsAccount();
        savings.setBalance(20000000);
        double savingsInterest = savings.calculateInterest();
        System.out.println("Savings interest: " + savingsInterest);
    }
}
```

Exercise 02:

Create an interface called "Shape" with two abstract methods: "double calculateArea()" and "double calculatePerimeter()". Implement the "Shape" interface in three classes: "Circle", "Rectangle", and "Triangle". Each class should have private instance variables relevant to its shape, and provide public getter and setter methods for these variables. Additionally, each class should define a constructor that initializes the instance variables. Write the implementation code for the "Shape" interface, the getter and setter methods in each class, and the constructors in each class.

```
public interface Shape {
    double calculateArea();
    double calculatePerimeter();
}

public class Circle implements Shape {
    private double radius;
```

```
    // Constructor
    public Circle(double radius) {
        this.radius = radius;
    }

    // Getter for 'radius'
    public double getRadius() {
        return radius;
    }

    // Setter for 'radius'
    public void setRadius(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }

    @Override
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
```

```
    }
}
public class Rectangle implements Shape {
    private double length;
    private double width;

    // Constructor
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Getters for 'length' and 'width'
    public double getLength() {
        return length;
    }

    public double getWidth() {
        return width;
    }

    // Setters for 'length' and 'width'
    public void setLength(double length) {
        this.length = length;
```

```
    }

    public void setWidth(double width) {
        this.width = width;
    }

    @Override
    public double calculateArea() {
        return length * width;
    }

    @Override
    public double calculatePerimeter() {
        return 2 * (length + width);
    }
}

public class Triangle implements Shape {
    private double side1;
    private double side2;
    private double side3;

    // Constructor
    public Triangle(double side1, double side2, double side3) {
```

```
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    // Getters for 'side1', 'side2', and 'side3'
    public double getSide1() {
        return side1;
    }

    public double getSide2() {
        return side2;
    }

    public double getSide3() {
        return side3;
    }

    // Setters for 'side1', 'side2', and 'side3'
    public void setSide1(double side1) {
        this.side1 = side1;
    }

    public void setSide2(double side2) {
```

```
        this.side2 = side2;
    }

    public void setSide3(double side3) {
        this.side3 = side3;
    }

    @Override
    public double calculateArea() {
        // Assuming the triangle is valid (sum of any two sides is greater than the third side)
        double s = (side1 + side2 + side3) / 2;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }

    @Override
    public double calculatePerimeter() {
        return side1 + side2 + side3;
    }
}
```