Assignment 3- OOPJ

Note:

- The assignment is designed to practice class, fields, and methods only.
- Create a separate project for each question.
- Do not use getter/setter methods or constructors for these assignments.
- Define two classes: one class to implement the logic and another class to test it.

1. Loan Amortization Calculator

Implement a system to calculate and display the monthly payments for a mortgage loan. The system should:

1. Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
2. Calculate the monthly payment using the standard mortgage formula:
    - Monthly Payment Calculation:
        - monthlyPayment = principal * (monthlyInterestRate * (1 + monthlyInterestRate)^(numberOfMonths)) / ((1 + monthlyInterestRate)^(numberOfMonths)- 1)
        - Where monthlyInterestRate = annualInterestRate / 12 / 100 and numberOfMonths = loanTerm * 12
        - Note: Here ^ means power and to find it you can use Math.pow( ) method
3. Display the monthly payment and the total amount paid over the life of the loan, in Indian Rupees (₹).

Define class LoanAmortizationCalculator with methods acceptRecord, calculateMonthlyPayment & printRecord and test the functionality in main method.

Solution:

```
import java.util.Scanner;

public class LoanAmortizationTest {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      LoanAmortizationCalculator loanCalculator = new LoanAmortizationCalculator();

      System.out.print("Enter principal amount: ");
      double principal = sc.nextDouble();

      System.out.print("Enter annual interest rate: ");
      double rate = sc.nextDouble();

      System.out.print("Enter loan term (in years): ");
      int term = sc.nextInt();
```

```
        loanCalculator.acceptRecord(principal, rate, term);
        loanCalculator.calculateMonthlyPayment();
        loanCalculator.printRecord();

        sc.close();
    }
}

class LoanAmortizationCalculator {
    double principal;
    double rate;
    int term;

    void acceptRecord(double p, double r, int t) {
        principal = p;
        rate = r;
        term = t;
    }

    void calculateMonthlyPayment() {
        double monthlyRate = rate / 12 / 100;
        int months = term * 12;
        double monthlyPayment = principal * (monthlyRate * Math.pow(1 + monthlyRate, months)) /
(Math.pow(1 + monthlyRate, months)- 1);
        System.out.printf("Monthly Payment: ₹%.2f%n", monthlyPayment);
    }

    void printRecord() {
        double totalPayment = principal * rate * term / 100 + principal;
        System.out.printf("Total Payment: ₹%.2f%n", totalPayment);
    }
}
```

```
Enter principal amount: 500000
Enter annual interest rate: 7.5
Enter loan term (in years): 15
Monthly Payment: ₹4635.06
Total Payment: ₹1062500.00
```

2. Compound Interest Calculator for Investment

Develop a system to compute the future value of an investment with compound interest. The system should:

1. Accept the initial investment amount, annual interest rate, number of times the interest is compounded per year, and investment duration (in years) from the user.
2. Calculate the future value of the investment using the formula:
    - Future Value Calculation:
        - futureValue = principal * (1 + annualInterestRate / numberOfCompounds)^(numberOfCompounds * years)
    - Total Interest Earned: totalInterest = futureValue- principal
3. Display the future value and the total interest earned, in Indian Rupees (₹).

Define class CompoundInterestCalculator with methods acceptRecord , calculateFutureValue, printRecord and test the functionality in main method.

Solution:

```
import java.util.Scanner;

public class CompoundInterestCalculatorTest{
   public static void main (String[] args) {
      Scanner sc = new Scanner (System.in);

      CompoundInterestCalculator calculator = new CompoundInterestCalculator();

      System.out.print ("Enter initial investment amount: ");
      double principal = sc.nextDouble();

      System.out.print ("Enter annual interest rate: ");
      double rate = sc.nextDouble();

      System.out.print ("Enter number of times interest is compounded per year: ");
      int compoundsPerYear = sc.nextInt();

      System.out.print ("Enter investment duration (in years): ");
      int years = sc.nextInt();


      calculator.acceptRecord(principal, rate, compoundsPerYear, years);
      calculator.calculateFutureValue();
      calculator.printRecord();

      sc.close();
   }
}

class CompoundInterestCalculator{
   double principal;
   double rate;
```

```
    int compoundsPerYear;
    int years;
    double futureValue;

    void acceptRecord (double p, double r, int c, int y){
        principal = p;
        rate = r;
        compoundsPerYear = c;
        years = y;
    }

    void calculateFutureValue(){
        futureValue = principal * Math.pow(1 + rate / compoundsPerYear / 100, compoundsPerYear *
years);
    }

    void printRecord(){
        double totalInterest = futureValue- principal;
        System.out.printf ("Future Value: ₹%.2f%n", futureValue);
        System.out.printf ("Total Interest Earned: ₹%.2f%n", totalInterest);
    }
}
```

```
Enter initial investment amount: 100000
Enter annual interest rate: 5.5
Enter number of times interest is compounded per year: 4
Enter investment duration (in years): 10
Future Value: ₹172677.08
Total Interest Earned: ₹72677.08
```

3. BMI (Body Mass Index) Tracker

Create a system to calculate and classify Body Mass Index (BMI). The system should:

1. Accept weight (in kilograms) and height (in meters) from the user.
2. Calculate the BMI using the formula:
    - BMI Calculation: BMI = weight / (height * height)
3. Classify the BMI into one of the following categories:
    - Underweight: BMI < 18.5
    - Normal weight: 18.5 ≤ BMI < 24.9
    - Overweight: 25 ≤ BMI < 29.9
    - Obese: BMI ≥ 30
4. Display the BMI value and its classification.

Define class BMITracker with methods acceptRecord, calculateBMI, classifyBMI & printRecord and test the functionality in main method.

Solution:

```
import java.util.Scanner;
```

```
public class BMITrackerTest {
    public static void main (String[] args){
        Scanner sc = new Scanner(System.in);

        BMITracker bmiTracker = new BMITracker();

        System.out.print("Enter weight (in kg): ");
        double weight = sc.nextDouble();

        System.out.print("Enter height (in meters): ");
        double height = sc.nextDouble();


        bmiTracker.acceptRecord(weight, height);

        bmiTracker.calculateBMI();
        bmiTracker.classifyBMI();

        bmiTracker.printRecord();

        sc.close();
    }
}

class BMITracker{
    double weight;
    double height;
    double bmi;
    String classification;

    void acceptRecord(double w, double h){
        weight = w;
        height = h;
    }

    void calculateBMI(){
        bmi = weight/(height * height);
    }

    void classifyBMI(){
        if (bmi < 18.5){
            classification = "Underweight";
        }
        else if (bmi >= 18.5 && bmi < 24.9){
            classification = "Normal weight";
        }
        else if (bmi >= 25 && bmi < 29.9){
```

```
            classification = "Overweight";
        }
        else{
            classification = "Obese";
        }
    }

    void printRecord(){
        System.out.printf("BMI: %.2f%n", bmi);
        System.out.println("Classification: " + classification);
    }
}
```

```
Enter weight (in kg): 45
Enter height (in meters): 1.55
BMI: 18.73
Classification: Normal weight
```

4. Discount Calculation for Retail Sales

Design a system to calculate the final price of an item after applying a discount. The system should:

1. Accept the original price of an item and the discount percentage from the user.
2. Calculate the discount amount and the final price using the following formulas:
    - Discount Amount Calculation: discountAmount = originalPrice * (discountRate / 100)
    - Final Price Calculation: finalPrice = originalPrice- discountAmount
3. Display the discount amount and the final price of the item, in Indian Rupees (₹).

Define class DiscountCalculator with methods acceptRecord, calculateDiscount & printRecord and test the functionality in main method

Solution:

```
import java.util.Scanner;

public class DiscountCalculator{
    private double originalPrice;
    private double discountRate;
    private double discountAmount;
    private double finalPrice;

    public void acceptRecord(){
        Scanner scanner = new Scanner(System.in);

        System.out.print ("Enter the original price of the item (₹): ");
```

```
        originalPrice = scanner.nextDouble();

        System.out.print ("Enter the discount percentage: ");
        discountRate = scanner.nextDouble();


        scanner.close();
    }

    public void calculateDiscount(){

        discountAmount = originalPrice*(discountRate/100);
        finalPrice = originalPrice- discountAmount;
    }

    public void printRecord(){

        System.out.printf ("Discount Amount: ₹%.2f%n", discountAmount);
        System.out.printf ("Final Price: ₹%.2f%n", finalPrice);
    }

    public static void main (String[] args){

        DiscountCalculator calculator = new DiscountCalculator();

        calculator.acceptRecord();
        calculator.calculateDiscount();
        calculator.printRecord();
    }
}
```

```
Enter the original price of the item (₹): 2000
Enter the discount percentage: 25
Discount Amount: ₹500.00
Final Price: ₹1500.00
```

5. Toll Booth Revenue Management

Develop a system to simulate a toll booth for collecting revenue. The system should:

1. Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.
2. Accept the number of vehicles of each type passing through the toll booth.
3. Calculate the total revenue based on the toll rates and number of vehicles.
4. Display the total number of vehicles and the total revenue collected, in Indian Rupees (₹).

- Toll Rate Examples:
    - Car: ₹50.00

- Truck: ₹100.00
- Motorcycle: ₹30.00

Define class TollBoothRevenueManager with methods
acceptRecord, setTollRates, calculateRevenue & printRecord and test the functionality in main method.

Solution:

```
import java.util.Scanner;
public class TollBoothRevenueManager{

   private double carRate;
   private double truckRate;
   private double motorcycleRate;
   private int numCars;
   private int numTrucks;
   private int numMotorcycles;
   private double totalRevenue;

   public void setTollRates(){
      Scanner scanner = new Scanner(System.in);

      System.out.print ("Enter the toll rate for Car (₹): ");
      carRate = scanner.nextDouble();

      System.out.print ("Enter the toll rate for Truck (₹): ");
      truckRate = scanner.nextDouble();

      System.out.print ("Enter the toll rate for Motorcycle (₹): ");
      motorcycleRate = scanner.nextDouble();
   }

   public void acceptRecord(){
      Scanner scanner = new Scanner(System.in);

      System.out.print ("Enter the number of Cars: ");
      numCars = scanner.nextInt();

      System.out.print ("Enter the number of Trucks: ");
      numTrucks = scanner.nextInt();

      System.out.print ("Enter the number of Motorcycles: ");
      numMotorcycles = scanner.nextInt();
   }

   public void calculateRevenue(){
```

```
        totalRevenue = (numCars * carRate) + (numTrucks * truckRate) + (numMotorcycles *
motorcycleRate);
    }

    public void printRecord(){

        int totalVehicles = numCars + numTrucks + numMotorcycles;
        System.out.printf ("Total Number of Vehicles: %d%n", totalVehicles);
        System.out.printf ("Total Revenue Collected: ₹%.2f%n", totalRevenue);
    }


    public static void main(String[] args) {
        TollBoothRevenueManager manager = new TollBoothRevenueManager();

        manager.setTollRates();
        manager.acceptRecord();
        manager.calculateRevenue();
        manager.printRecord();
    }
}
```

```
Enter the number of Cars: 20
Enter the number of Trucks: 10
Enter the number of Motorcycles: 15
Total Number of Vehicles: 45
Total Revenue Collected: ₹2450.00
```