Assignment - 3

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:
o 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.

```
package loan_amortization_calculator;
import java.util.Scanner;
public class Loan_calculator {
        private double principal;
  private double annualInterestRate;
  private int loanTerm;
  private double monthlyPayment;
  private double totalPayment;
  public void acceptRecord() {
     Scanner scanner = new Scanner(System.in);
     System.out.print("Enter the principal amount (loan amount) in ₹: ");
     principal = scanner.nextDouble();
     System.out.print("Enter the annual interest rate (in %): ");
     annualInterestRate = scanner.nextDouble();
     System.out.print("Enter the loan term (in years): ");
     loanTerm = scanner.nextInt();
     scanner.close();
  }
```

```
  public void calculateMonthlyPayment() {
      double monthlyInterestRate = annualInterestRate / 12 / 100;
      int numberOfMonths = loanTerm * 12;
      monthlyPayment = principal * (monthlyInterestRate * Math.pow(1 +
monthlyInterestRate, numberOfMonths)) /
                      (Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1);
      totalPayment = monthlyPayment * numberOfMonths;
  }
  public void printRecord() {
      System.out.printf("Monthly Payment: ₹%.2f%n", monthlyPayment);
      System.out.printf("Total Amount Paid over the life of the loan:
₹%.2f%n", totalPayment);
  }
  public static void main(String[] args) {
        Loan_calculator calculator = new Loan_calculator();

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

```
Enter the principal amount (loan amount) in ₹: 20000
Enter the annual interest rate (in %): 2
Enter the loan term (in years): 3
Monthly Payment: ₹572.85
Total Amount Paid over the life of the loan: ₹20622.66
```

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.

```
package CI_calculator;
import java.util.Scanner;
public class cicalculator {
        private double principal;
  private double annualInterestRate;
  private int numberOfCompounds;
  private int years;

  // Method to accept user input
  public void acceptRecord() {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the initial investment amount (₹): ");
      principal = scanner.nextDouble();

      System.out.print("Enter the annual interest rate (in percentage): ");
      annualInterestRate = scanner.nextDouble() / 100;

      System.out.print("Enter the number of times the interest is
compounded per year: ");
      numberOfCompounds = scanner.nextInt();

      System.out.print("Enter the investment duration (in years): ");
      years = scanner.nextInt();
  }

  // Method to calculate the future value
  public double calculateFutureValue() {
```

```
        return principal * Math.pow(1 + annualInterestRate /
numberOfCompounds, numberOfCompounds * years);
    }

    // Method to calculate total interest earned
    public double calculateTotalInterest(double futureValue) {
        return futureValue - principal;
    }

    // Method to print the results
    public void printRecord() {
        double futureValue = calculateFutureValue();
        double totalInterest = calculateTotalInterest(futureValue);

        System.out.printf("Future Value of the investment: ₹%.2f%n",
futureValue);
        System.out.printf("Total Interest Earned: ₹%.2f%n", totalInterest);
    }

    // Main method to test the functionality
    public static void main(String[] args) {
          cicalculator calculator = new cicalculator();

        calculator.acceptRecord();  // Accept input from user
        calculator.printRecord();
}}
```

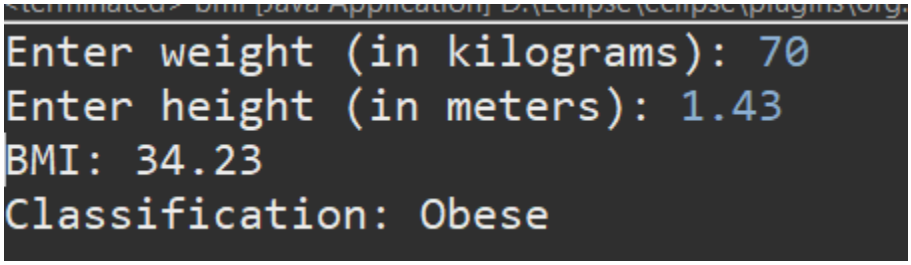
```
Enter the initial investment amount (₹): 50000
Enter the annual interest rate (in percentage): 12
Enter the number of times the interest is compounded per year: 10
Enter the investment duration (in years): 2
Future Value of the investment: ₹63471.72
Total Interest Earned: ₹13471.72
```

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




```java
package BMI;
import java.util.Scanner;
public class bmi {
        private double weight;
  private double height;

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

     System.out.print("Enter weight (in kilograms): ");
     weight = scanner.nextDouble();

     System.out.print("Enter height (in meters): ");
     height = scanner.nextDouble();
```

```
    }

    // Method to calculate BMI
    public double calculateBMI() {
        if (height <= 0) {
            throw new IllegalArgumentException("Height must be greater than
zero.");
        }
        return weight / (height * height);
    }

    // Method to classify BMI
    public String classifyBMI(double bmi) {
        if (bmi < 18.5) {
            return "Underweight";
        } else if (bmi < 24.9) {
            return "Normal weight";
        } else if (bmi < 29.9) {
            return "Overweight";
        } else {
            return "Obese";
        }
    }

    // Method to print the BMI and its classification
    public void printRecord() {
        double bmi = calculateBMI();
        String classification = classifyBMI(bmi);

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

    public static void main(String[] args) {
          bmi tracker = new bmi();
```

```
        tracker.acceptRecord();  // Accept input from user
        tracker.printRecord();
}
}
```

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.

```
package discount;
import java.util.Scanner;
public class discount {
        private double originalPrice;
  private double discountRate;

  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();
  }

  public double[] calculateDiscount() {
```

```
    double discountAmount = originalPrice * (discountRate / 100);
    double finalPrice = originalPrice - discountAmount;
    return new double[] { discountAmount, finalPrice };
  }

  public void printRecord() {
    double[] results = calculateDiscount();
    double discountAmount = results[0];
    double finalPrice = results[1];

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

  public static void main(String[] args) {
      discount calculator = new discount();

    calculator.acceptRecord();  // Accept input from user
    calculator.printRecord();
}
  }
```

```
Enter the original price of the item (₹): 1000
Enter the discount percentage: 20
Discount Amount: ₹200.00
Final Price: ₹800.00
```

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.

```
package toll_booth;
import java.util.Scanner;
public class toll_booth {
        private double carTollRate;
        private double truckTollRate;
        private double motorcycleTollRate;
        private int numCars;
        private int numTrucks;
        private int numMotorcycles;

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

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

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

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

        public double calculateRevenue() {
            return (numCars * carTollRate) + (numTrucks * truckTollRate) +
(numMotorcycles * motorcycleTollRate);
        }

        public void printRecord() {
            double totalRevenue = calculateRevenue();
            int totalVehicles = numCars + numTrucks + numMotorcycles;

            System.out.println("Total number of vehicles: " + totalVehicles);
            System.out.printf("Total revenue collected: ₹%.2f%n",
totalRevenue);
        }

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

            manager.setTollRates();  // Set toll rates for vehicles
            manager.acceptRecord();  // Accept number of vehicles
            manager.printRecord();   // Calculate and print total revenue and
vehicle count
        }
    }
```

```
Enter the toll rate for Car (₹): 10
Enter the toll rate for Truck (₹): 20
Enter the toll rate for Motorcycle (₹): 5
Enter the number of Cars: 2
Enter the number of Trucks: 2
Enter the number of Motorcycles: 2
Total number of vehicles: 6
Total revenue collected: ₹70.00
```