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.

CODE:

```java
import java.util.Scanner;

class LoanAmortizationCalculator {

    double principal;
    double annualInterestRate;
    int loanTerm;
    double monthlyPayment;

    public void acceptrecord() {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the loan amount");
        principal = sc.nextDouble();

        System.out.println("Enter the interest rate");
        annualInterestRate = sc.nextDouble();

        System.out.println("Enter the loan term");
        loanTerm = sc.nextInt();

        sc.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);

    }

    public void printrecord() {
        double totalAmount = monthlyPayment * loanTerm * 12;

        System.out.printf("Monthly payment over the life is: %.2f\n ",
monthlyPayment);
        System.out.printf("total payment Amount over the life is: %.2f\n ",
totalAmount);
    }
}

public class loanAmount {

    public static void main(String[] args) {

        LoanAmortizationCalculator dis = new LoanAmortizationCalculator();

        dis.acceptrecord();

        dis.calculateMonthlyPayment();

        dis.printrecord();

    }

}
```

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

```java
import java.util.Scanner;

class CompoundInterestCalculator {

    double principal;
    double annualInterestRate;
    double numberOfCompounds;
    int years;
    double futureValue;

    public void acceptrecord() {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the principal ");
        principal = sc.nextDouble();

        System.out.println("Enter the annualInterestRate ");
        annualInterestRate = sc.nextDouble();

        System.out.println("Enter the numberOfCompounds ");
        numberOfCompounds = sc.nextInt();

        System.out.println("Enter the years ");
        years = sc.nextInt();

    }

    public void calculateFutureValue() {
        double interestRateDecimal = annualInterestRate / 100;
        futureValue = principal
                * Math.pow((1 + interestRateDecimal / numberOfCompounds),
numberOfCompounds * years);
    }

    public void printRecord() {

        double totalInterest = futureValue - principal;

        System.out.printf("Future value is: %.2f\n", futureValue);
        System.out.printf("totalInterest earn is: %.2f\n", totalInterest);

    }
}

public class Prgm2 {

    public static void main(String[] args) {

        CompoundInterestCalculator c = new CompoundInterestCalculator();

        c.acceptrecord();

        c.calculateFutureValue();

        c.printRecord();

    }

}
```

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
import java.util.Scanner;

class BMITracker {
    int weight;
    float height;
    double BMI;

    public void acceptrecord() {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the Weight ");
        weight = sc.nextInt();

        System.out.println("Enter the height ");
        height = sc.nextFloat();

    }

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

    public String classifyBMI() {
        if (BMI < 18.5) {
            return "Underweight";

        }

        else if (BMI > 18.5 && BMI < 24.9) {
            return "Normal weight";
        }

        else if (BMI > 25 && BMI < 29.9) {
            return "Overweight";

        }

        else {
            return "Obese";
        }
    }

    public void printrecord() {

        String classification = classifyBMI();

        System.out.printf("Bmi is:%.2f ", BMI);
        System.out.println("\nYou are: " + classification);

    }
}

public class Prgm3 {

    public static void main(String[] args) {
        BMITracker b = new BMITracker();

        b.acceptrecord();

        b.calculateBMI();

        b.classifyBMI();

        b.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.

```java
import java.util.Scanner;

class DiscountCalculator {
    double originalPrice;
    double discountRate;
    double discountAmount;
    public void acceptrecord() {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the Original Price ");
        originalPrice = sc.nextDouble();

        System.out.println("Enter the Discount percent");
        discountRate = sc.nextDouble();

        sc.close();
    }

    public void discountRate() {
        discountAmount = (originalPrice * (discountRate / 100));
    }

    public void printrecord() {

        double finalPrice = originalPrice - discountAmount;

        System.out.println("Discount is: " + discountAmount);
        System.out.println("final price is: " + finalPrice);
    }

}
public class Prgm4 {

    public static void main(String[] args) {

        DiscountCalculator d = new DiscountCalculator();

        d.acceptrecord();

        d.discountRate();

        d.printrecord();
    }

}
```

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.

```java
import java.util.Scanner;

class TollBoothRevenueManager {

    double CarRate;
    double TruckRate;
    double MotorcycleRate;

    int CarCnt;
    int TruckCnt;
    int MotorCnt;

    double totalrevenue;

    Scanner sc = new Scanner(System.in);

    public void acceptrecord() {

        System.out.println("Enter the rate for car");
        CarRate = sc.nextInt();

        System.out.println("Enter the rate for Truck");
        TruckRate = sc.nextInt();

        System.out.println("Enter the rate for motor");
        MotorcycleRate = sc.nextInt();
    }

    public void setTollRates() {

        System.out.println("Enter the number of  car");
        CarCnt = sc.nextInt();

        System.out.println("Enter the number of Truck");
        TruckCnt = sc.nextInt();

        System.out.println("Enter the number of motor");
        MotorCnt = sc.nextInt();

    }

    public void calculateRevenue() {
        totalrevenue = (CarRate * CarCnt + TruckRate * TruckCnt +
MotorcycleRate * MotorCnt);
    }

    public void printrecord() {

        int totalnovehicle = CarCnt + MotorCnt + TruckCnt;
        System.out.println("total no of vehicle is: " + totalnovehicle);
        System.out.println("total revenue is: " + totalrevenue);
    }
}

public class Prgm5 {

    public static void main(String[] args) {

        TollBoothRevenueManager r = new TollBoothRevenueManager();
        r.acceptrecord();

        r.setTollRates();

        r.calculateRevenue();

        r.printrecord();
    }
}
```