Assignment 4

1. Loan Amortization Calculator

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

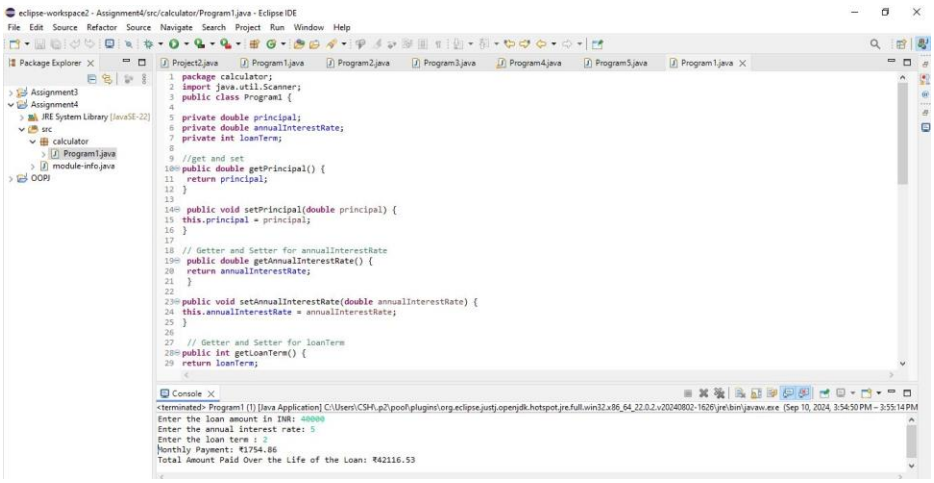
1. Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
2. Calculate the monthly payment using the standard mortgage formula:
   - **Monthly Payment Calculation:**
     - `monthlyPayment = principal * (monthlyInterestRate * (1 + monthlyInterestRate)^(numberOfMonths)) / ((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 (₹).

Code :-

```
package calculator;
import java.util.Scanner;
public class Program1 {

private double principal;
private double annualInterestRate;
private int loanTerm;

//get and set
public double getPrincipal() {
 return principal;
}

 public void setPrincipal(double principal) {
this.principal = principal;
}

// Getter and Setter for annualInterestRate
 public double getAnnualInterestRate() {
 return annualInterestRate;
 }

public void setAnnualInterestRate(double annualInterestRate) {
this.annualInterestRate = annualInterestRate;
}

 // Getter and Setter for loanTerm
public int getLoanTerm() {
return loanTerm;
```

```
 }

public void setLoanTerm(int loanTerm) {
this.loanTerm = loanTerm;
 }

// calculate the monthly payment
public double calculateMonthlyPayment() {
double monthlyInterestRate = annualInterestRate / 12 / 100;
int numberOfMonths = loanTerm * 12;
 return principal * (monthlyInterestRate * Math.pow(1 + monthlyInterestRate,
numberOfMonths))/ (Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1);
 }

//  calculate the total amount paid
public double calculateTotalAmountPaid() {
 return calculateMonthlyPayment() * loanTerm * 12;
 }

 public static void main(String[] args) {

Program1 calculator = new Program1();

 // Create a Scanner
Scanner scanner = new Scanner(System.in);


System.out.print("Enter the loan amount in INR: ");
calculator.setPrincipal(scanner.nextDouble());

System.out.print("Enter the annual interest rate: ");
calculator.setAnnualInterestRate(scanner.nextDouble());

System.out.print("Enter the loan term : ");
calculator.setLoanTerm(scanner.nextInt());

scanner.close();

 // Calculate monthly payment and total amount paid
double monthlyPayment = calculator.calculateMonthlyPayment();
double totalAmountPaid = calculator.calculateTotalAmountPaid();

// Display the results
System.out.printf("Monthly Payment: ₹%.2f%n", monthlyPayment);
System.out.printf("Total Amount Paid Over the Life of the Loan: ₹%.2f%n",
totalAmountPaid);
                  }
              }
```

Output –



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 (₹).

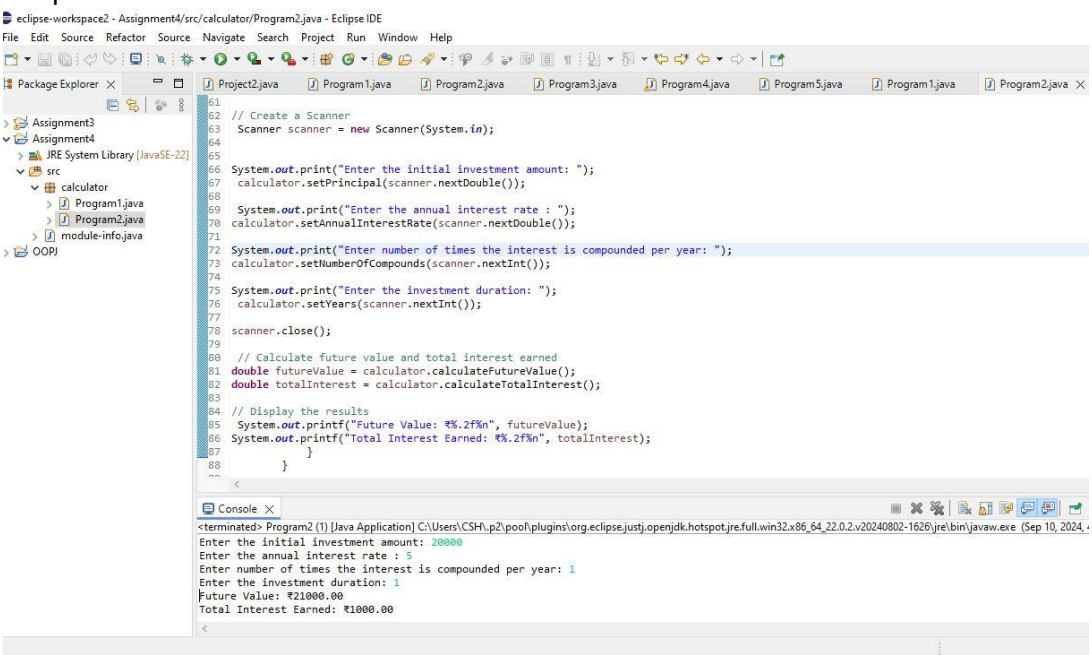
Code :-

```
package calculator;
import java.util.Scanner;
public class Program2 {

private double principal;
private double annualInterestRate;
private int numberOfCompounds;
private int years;

// Get and Set
public double getPrincipal() {
return principal;
 }

 public void setPrincipal(double principal) {
 this.principal = principal;
```

```
 }

 // Getter and Setter for annualInterestRate
public double getAnnualInterestRate() {
return annualInterestRate;
 }

public void setAnnualInterestRate(double annualInterestRate) {
this.annualInterestRate = annualInterestRate;
}

// Getter and Setter for numberOfCompounds
 public int getNumberOfCompounds() {
 return numberOfCompounds;
 }

 public void setNumberOfCompounds(int numberOfCompounds) {
 this.numberOfCompounds = numberOfCompounds;
}

// Getter and Setter for years
public int getYears() {
return years;
 }

public void setYears(int years) {
this.years = years;
}

//calculate the future value
 public double calculateFutureValue() {
double ratePerPeriod = annualInterestRate / numberOfCompounds / 100;
 int totalPeriods = numberOfCompounds * years;
 return principal * Math.pow(1 + ratePerPeriod, totalPeriods);
 }

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

 public static void main(String[] args) {

Program2 calculator = new Program2();

// Create a Scanner
 Scanner scanner = new Scanner(System.in);


System.out.print("Enter the initial investment amount: ");
 calculator.setPrincipal(scanner.nextDouble());

 System.out.print("Enter the annual interest rate : ");
calculator.setAnnualInterestRate(scanner.nextDouble());
```

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

System.out.print("Enter the investment duration: ");
 calculator.setYears(scanner.nextInt());

scanner.close();

 // Calculate future value and total interest earned
double futureValue = calculator.calculateFutureValue();
double totalInterest = calculator.calculateTotalInterest();

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

Output –



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

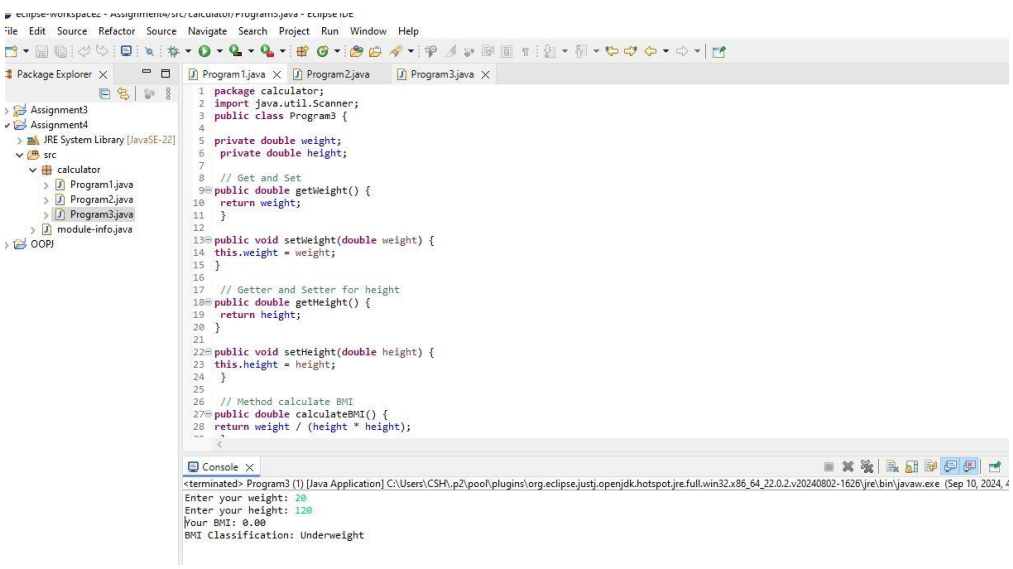
4. Display the BMI value and its classification.

Code:-

```
package calculator;
import java.util.Scanner;
public class Program3 {

private double weight;
 private double height;

 // Get and Set
public double getWeight() {
 return weight;
 }

public void setWeight(double weight) {
this.weight = weight;
}

 // Getter and Setter for height
public double getHeight() {
 return height;
}

public void setHeight(double height) {
this.height = height;
 }

 // Method calculate BMI
public double calculateBMI() {
return weight / (height * height);
 }

// Method classify BMI
public String classifyBMI(double bmi) {
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 static void main(String[] args) {
```

```java
Program3 bmiTracker = new Program3();

// Create a Scanner object for user input
Scanner scanner = new Scanner(System.in);

// Prompt the user to enter weight and height
System.out.print("Enter your weight: ");
bmiTracker.setWeight(scanner.nextDouble());

System.out.print("Enter your height: ");
bmiTracker.setHeight(scanner.nextDouble());

scanner.close();

// Calculate BMI
double bmi = bmiTracker.calculateBMI();

// Classify BMI
String classification = bmiTracker.classifyBMI(bmi);

// Display the results
System.out.printf("Your BMI: %.2f%n", bmi);
System.out.println("BMI Classification: " + classification);
                 }
             }
```

Output –



4. Discount Calculation for Retail Sales

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

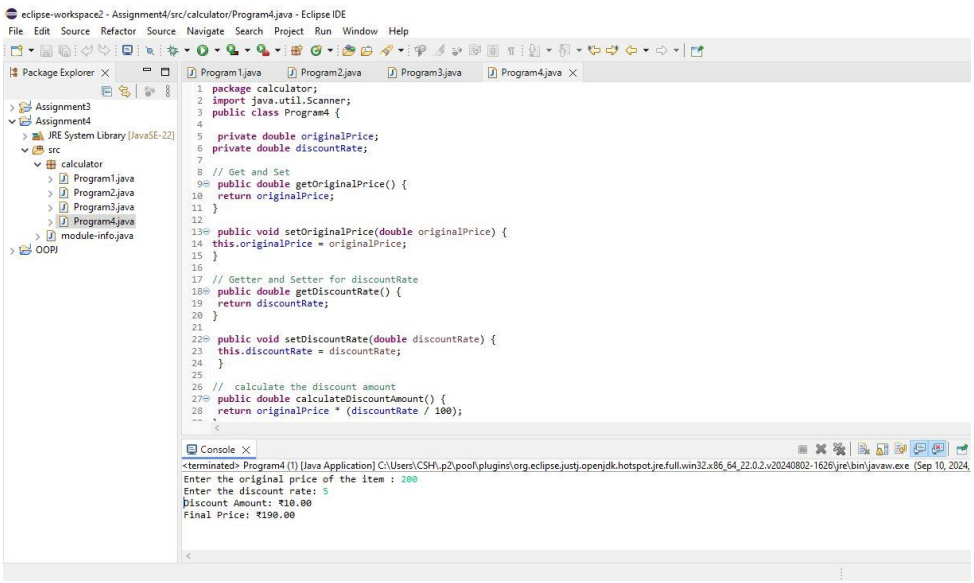
1. Accept the original price of an item and the discount percentage from the user.
2. Calculate the discount amount and the final price using the following formulas:
   - **Discount Amount Calculation:** `discountAmount = originalPrice * (discountRate / 100)`
   - **Final Price Calculation:** `finalPrice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

Code –

```
package calculator;
import java.util.Scanner;
public class Program4 {

 private double originalPrice;
private double discountRate;

// Get and Set
 public double getOriginalPrice() {
 return originalPrice;
}

 public void setOriginalPrice(double originalPrice) {
this.originalPrice = originalPrice;
}

// Getter and Setter for discountRate
 public double getDiscountRate() {
 return discountRate;
}

 public void setDiscountRate(double discountRate) {
 this.discountRate = discountRate;
 }

//  calculate the discount amount
 public double calculateDiscountAmount() {
 return originalPrice * (discountRate / 100);
}

 // calculate the final price
 public double calculateFinalPrice() {
return originalPrice - calculateDiscountAmount();
 }

public static void main(String[] args) {

Program4 calculator = new Program4();

// Create a Scanner
Scanner scanner = new Scanner(System.in);

// enter original price and discount rate
System.out.print("Enter the original price of the item : ");
```

```
calculator.setOriginalPrice(scanner.nextDouble());

System.out.print("Enter the discount rate: ");
 calculator.setDiscountRate(scanner.nextDouble());

 scanner.close();

// Calculate discount amount and final price
double discountAmount = calculator.calculateDiscountAmount();
double finalPrice = calculator.calculateFinalPrice();

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

Output –



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

Code :-

```
package calculator;
import java.util.Scanner;
public class Program5 {

private double carTollRate;
private double truckTollRate;
private double motorcycleTollRate;
private int carCount;
private int truckCount;
private int motorcycleCount;

// Get and Set
public double getCarTollRate() {
return carTollRate;
 }

public void setCarTollRate(double carTollRate) {
this.carTollRate = carTollRate;
 }

// Getter and Setter for truckTollRate
public double getTruckTollRate() {
return truckTollRate;
}

 public void setTruckTollRate(double truckTollRate) {
this.truckTollRate = truckTollRate;
 }

// Getter and Setter for motorcycleTollRate
public double getMotorcycleTollRate() {
return motorcycleTollRate;
}

public void setMotorcycleTollRate(double motorcycleTollRate) {
this.motorcycleTollRate = motorcycleTollRate;
}

 // Getter and Setter for carCount
public int getCarCount() {
 return carCount;
                 }

public void setCarCount(int carCount) {
this.carCount = carCount;
}

// Getter and Setter for truckCount
public int getTruckCount() {
return truckCount;
```

```
}

public void setTruckCount(int truckCount) {
this.truckCount = truckCount;
 }

// Getter and Setter for motorcycleCount
public int getMotorcycleCount() {
return motorcycleCount;
}

public void setMotorcycleCount(int motorcycleCount) {
this.motorcycleCount = motorcycleCount;
}

 // Method to calculate total revenue
 public double calculateTotalRevenue() {
 return (carCount * carTollRate) + (truckCount * truckTollRate) + (motorcycleCount *
motorcycleTollRate);
}

// Method to calculate total number of vehicles
public int calculateTotalVehicles() {
return carCount + truckCount + motorcycleCount;
}

public static void main(String[] args) {
// Create an instance of TollBooth
Program5 tollBooth = new Program5();

// Create a Scanner
Scanner scanner = new Scanner(System.in);

// Set toll rates for vehicles
System.out.print("Enter toll rate for Car (in INR): ");
tollBooth.setCarTollRate(scanner.nextDouble());

System.out.print("Enter toll rate for Truck (in INR): ");
tollBooth.setTruckTollRate(scanner.nextDouble());

System.out.print("Enter toll rate for Motorcycle (in INR): ");
tollBooth.setMotorcycleTollRate(scanner.nextDouble());

// Accept the number of vehicles of each type
System.out.print("Enter the number of Cars: ");
tollBooth.setCarCount(scanner.nextInt());

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

System.out.print("Enter the number of Motorcycles: ");
tollBooth.setMotorcycleCount(scanner.nextInt());

scanner.close();
```

```java
// Calculate total revenue and total number of vehicles
double totalRevenue = tollBooth.calculateTotalRevenue();
int totalVehicles = tollBooth.calculateTotalVehicles();

// Display the total number of vehicles and total revenue
System.out.printf("Total Number of Vehicles: %d%n", totalVehicles);
System.out.printf("Total Revenue Collected: ₹%.2f%n", totalRevenue);
                }
            }
```

Output –