**Note:**

* The assignment is designed to practice constructor, getter/setter and toString method.
* Create a separate project for each question and create separate file for each class.
* Try to test the functionality by using menu-driven program.

**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 the class LoanAmortizationCalculator with fields, an appropriate constructor, getter and setter methods, a toString method and business logic methods. Define the class LoanAmortizationCalculatorUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method and test the functionality of the utility class.

**package** org.example;

**import** java.text.DecimalFormat;

**public** **class** LoanAmortizationCalculator {

**private** **double** principal;

**private** **double** annualInterestRate;

**private** **int** loanTerm;

**public** LoanAmortizationCalculator(**double** principal, **double** annualInterestRate, **int** loanTerm) {

**this**.principal = principal;

**this**.annualInterestRate = annualInterestRate;

**this**.loanTerm = loanTerm;

}

**public** **double** getPrincipal() {

**return** principal;

}

**public** **void** setPrincipal(**double** principal) {

**this**.principal = principal;

}

**public** **double** getAnnualInterestRate() {

**return** annualInterestRate;

}

**public** **void** setAnnualInterestRate(**double** annualInterestRate) {

**this**.annualInterestRate = annualInterestRate;

}

**public** **int** getLoanTerm() {

**return** loanTerm;

}

**public** **void** setLoanTerm(**int** loanTerm) {

**this**.loanTerm = loanTerm;

}

**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);

}

**public** **double** calculateTotalPayment() {

**return** calculateMonthlyPayment() \* loanTerm \* 12;

}

@Override

**public** String toString() {

DecimalFormat df = **new** DecimalFormat("₹###,###.00");

**return** "Principal Amount: " + df.format(principal) + "\n"

+ "Annual Interest Rate: " + annualInterestRate + "%\n"

+ "Loan Term: " + loanTerm + " years\n"

+ "Monthly Payment: " + df.format(calculateMonthlyPayment()) + "\n"

+ "Total Payment over the loan: " + df.format(calculateTotalPayment()) + "\n";

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** LoanAmortizationCalculatorUtil {

**public** LoanAmortizationCalculator acceptRecord() {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter Principal Amount (₹): ");

**double** principal = sc.nextDouble();

System.***out***.print("Enter Annual Interest Rate (%): ");

**double** annualInterestRate = sc.nextDouble();

System.***out***.print("Enter Loan Term (years): ");

**int** loanTerm = sc.nextInt();

**return** **new** LoanAmortizationCalculator(principal, annualInterestRate, loanTerm);

}

**public** **void** printRecord(LoanAmortizationCalculator calculator) {

System.***out***.println("\nLoan Amortization Details:");

System.***out***.println(calculator.toString());

}

**public** **void** menuList() {

System.***out***.println("\n----- Loan Amortization Calculator -----");

System.***out***.println("1. Enter Loan Details");

System.***out***.println("2. Display Loan Details");

System.***out***.println("3. Exit");

System.***out***.print("Choose an option: ");

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **void** main(String[] args) {

LoanAmortizationCalculatorUtil util = **new** LoanAmortizationCalculatorUtil();

LoanAmortizationCalculator calculator = **null**;

Scanner sc = **new** Scanner(System.***in***);

**while** (**true**) {

util.menuList();

**int** choice = sc.nextInt();

**switch** (choice) {

**case** 1:

calculator = util.acceptRecord();

**break**;

**case** 2:

**if** (calculator != **null**) {

util.printRecord(calculator);

} **else** {

System.***out***.println("Please enter loan details first.");

}

**break**;

**case** 3:

System.***out***.println("Exiting...");

sc.close();

System.*exit*(0);

**default**:

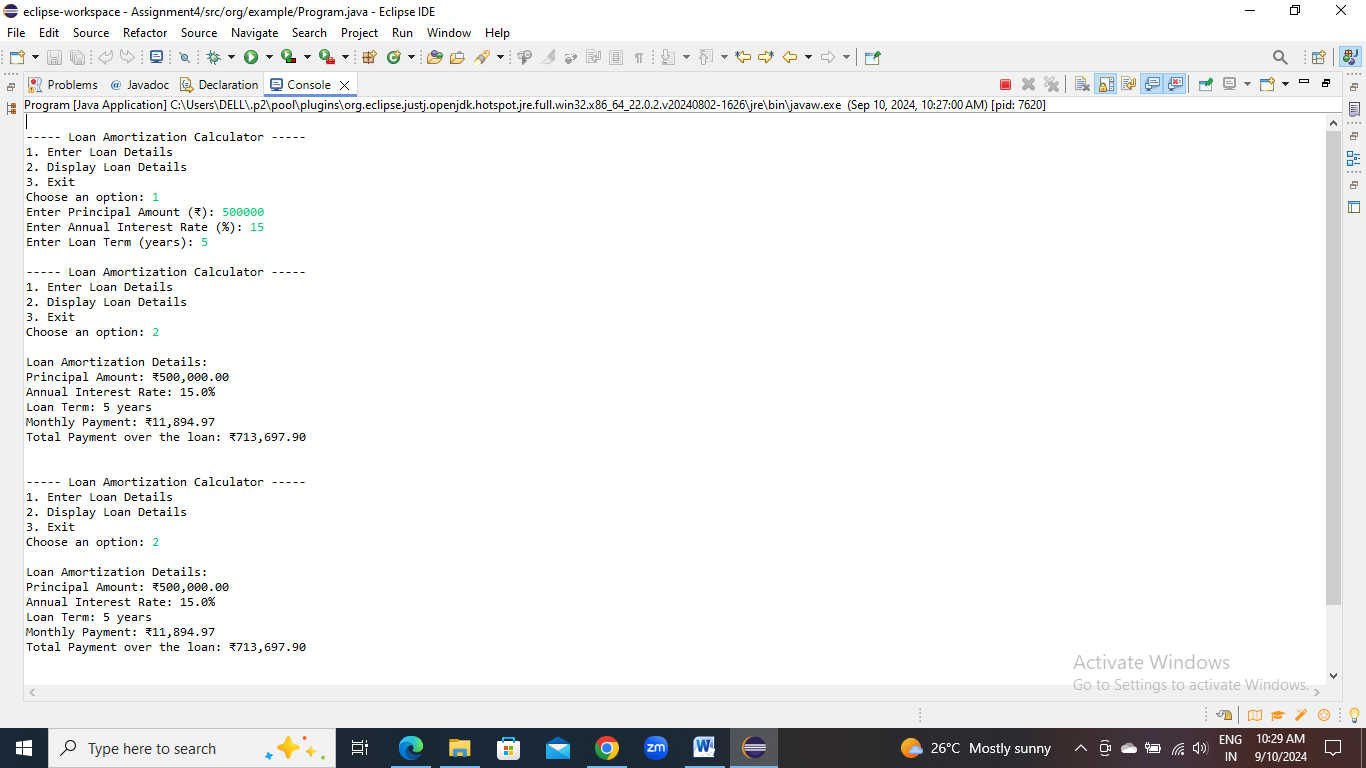
System.***out***.println("Invalid choice. Please try again.");

}

}

}

}



**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 the class CompoundInterestCalculator with fields, an appropriate constructor, getter and setter methods, a toString method and business logic methods. Define the class CompoundInterestCalculatorUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method to test the functionality of the utility class.

**package** org.example;

**import** java.text.DecimalFormat;

**public** **class** CompoundInterestCalculator {

**private** **double** principal;

**private** **double** annualInterestRate;

**private** **int** numberOfCompounds;

**private** **int** years;

**public** CompoundInterestCalculator(**double** principal, **double** annualInterestRate, **int** numberOfCompounds, **int** years) {

**this**.principal = principal;

**this**.annualInterestRate = annualInterestRate;

**this**.numberOfCompounds = numberOfCompounds;

**this**.years = years;

}

**public** **double** getPrincipal() {

**return** principal;

}

**public** **void** setPrincipal(**double** principal) {

**this**.principal = principal;

}

**public** **double** getAnnualInterestRate() {

**return** annualInterestRate;

}

**public** **void** setAnnualInterestRate(**double** annualInterestRate) {

**this**.annualInterestRate = annualInterestRate;

}

**public** **int** getNumberOfCompounds() {

**return** numberOfCompounds;

}

**public** **void** setNumberOfCompounds(**int** numberOfCompounds) {

**this**.numberOfCompounds = numberOfCompounds;

}

**public** **int** getYears() {

**return** years;

}

**public** **void** setYears(**int** years) {

**this**.years = years;

}

**public** **double** calculateFutureValue() {

**return** principal \* Math.*pow*(1 + (annualInterestRate / numberOfCompounds) / 100, numberOfCompounds \* years);

}

**public** **double** calculateTotalInterest() {

**return** calculateFutureValue() - principal;

}

@Override

**public** String toString() {

DecimalFormat df = **new** DecimalFormat("₹###,###.00");

**return** "Principal Amount: " + df.format(principal) + "\n"

+ "Annual Interest Rate: " + annualInterestRate + "%\n"

+ "Number of Times Compounded per Year: " + numberOfCompounds + "\n"

+ "Investment Duration: " + years + " years\n"

+ "Future Value: " + df.format(calculateFutureValue()) + "\n"

+ "Total Interest Earned: " + df.format(calculateTotalInterest()) + "\n";

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** CompoundInterestCalculatorUtil {

// Method to accept user input

**public** CompoundInterestCalculator acceptRecord() {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter Initial Investment Amount (₹): ");

**double** principal = sc.nextDouble();

System.***out***.print("Enter Annual Interest Rate (%): ");

**double** annualInterestRate = sc.nextDouble();

System.***out***.print("Enter Number of Times Interest is Compounded per Year: ");

**int** numberOfCompounds = sc.nextInt();

System.***out***.print("Enter Investment Duration (years): ");

**int** years = sc.nextInt();

**return** **new** CompoundInterestCalculator(principal, annualInterestRate, numberOfCompounds, years);

}

// Method to print the record

**public** **void** printRecord(CompoundInterestCalculator calculator) {

System.***out***.println("\nCompound Interest Calculation Details:");

System.***out***.println(calculator.toString());

}

// Method to display the menu

**public** **void** menuList() {

System.***out***.println("\n----- Compound Interest Calculator -----");

System.***out***.println("1. Enter Investment Details");

System.***out***.println("2. Display Investment Details");

System.***out***.println("3. Exit");

System.***out***.print("Choose an option: ");

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **void** main(String[] args) {

CompoundInterestCalculatorUtil util = **new** CompoundInterestCalculatorUtil();

CompoundInterestCalculator calculator = **null**;

Scanner sc = **new** Scanner(System.***in***);

**while** (**true**) {

util.menuList();

**int** choice = sc.nextInt();

**switch** (choice) {

**case** 1:

calculator = util.acceptRecord();

**break**;

**case** 2:

**if** (calculator != **null**) {

util.printRecord(calculator);

} **else** {

System.***out***.println("Please enter investment details first.");

}

**break**;

**case** 3:

System.***out***.println("Exiting...");

sc.close();

System.*exit*(0);

**default**:

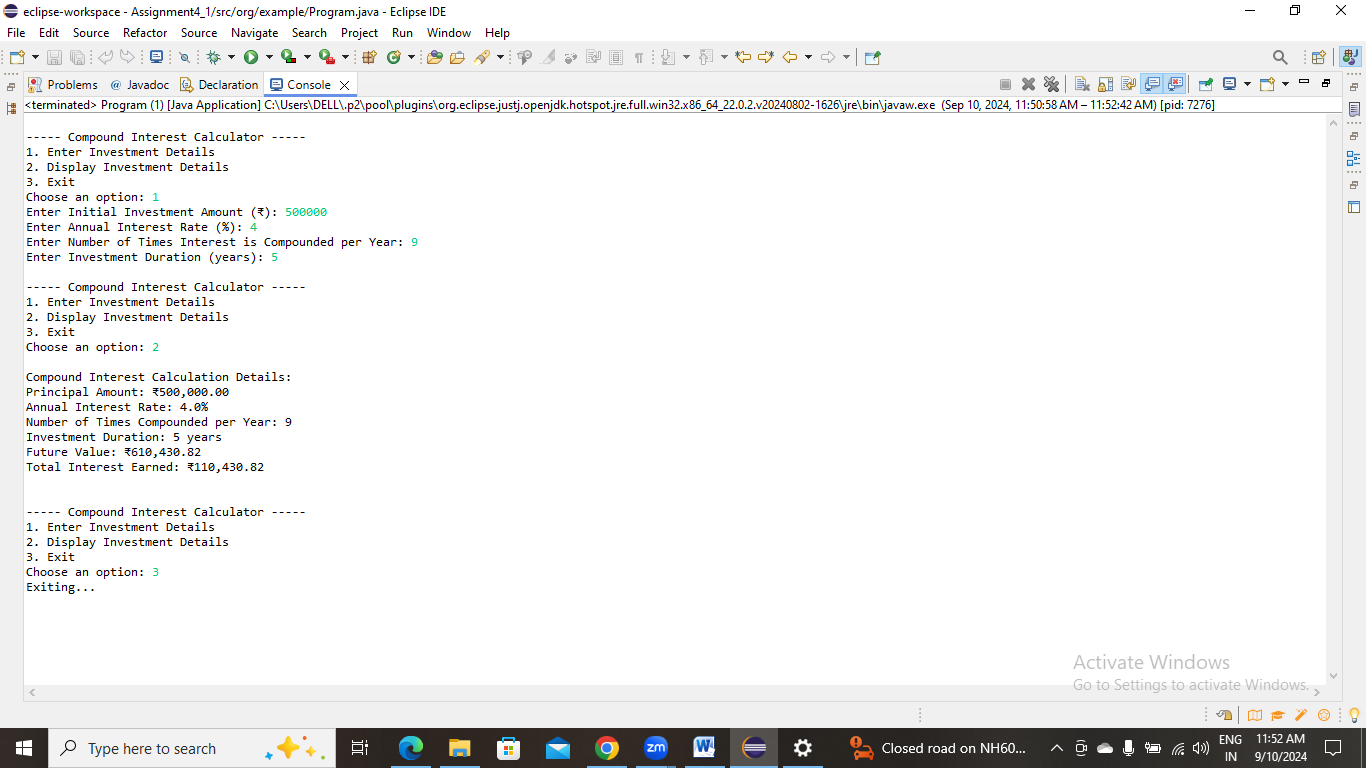
System.***out***.println("Invalid choice. Please try again.");

}

}

}

}



**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 the class BMITracker with fields, an appropriate constructor, getter and setter methods, a toString method, and business logic methods. Define the class BMITrackerUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method to test the functionality of the utility class.

**package** org.example;

**public** **class** BMITracker {

**private** **double** weight;

**private** **double** height;

**public** BMITracker(**double** weight, **double** height) {

**this**.weight = weight;

**this**.height = height;

}

**public** **double** getWeight() {

**return** weight;

}

**public** **void** setWeight(**double** weight) {

**this**.weight = weight;

}

**public** **double** getHeight() {

**return** height;

}

**public** **void** setHeight(**double** height) {

**this**.height = height;

}

**public** **double** calculateBMI() {

**return** weight / (height \* height);

}

**public** String classifyBMI() {

**double** bmi = calculateBMI();

**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";

}

}

@Override

**public** String toString() {

**return** String.*format*("Weight: %.2f kg\nHeight: %.2f m\nBMI: %.2f\nClassification: %s",

weight, height, calculateBMI(), classifyBMI());

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** BMITrackerUtil {

**public** BMITracker acceptRecord() {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter weight (in kg): ");

**double** weight = sc.nextDouble();

System.***out***.print("Enter height (in meters): ");

**double** height = sc.nextDouble();

**return** **new** BMITracker(weight, height);

}

**public** **void** printRecord(BMITracker bmiTracker) {

System.***out***.println("\nBMI Calculation Details:");

System.***out***.println(bmiTracker.toString());

}

**public** **void** menuList() {

System.***out***.println("\n----- BMI Tracker -----");

System.***out***.println("1. Enter Weight and Height");

System.***out***.println("2. Display BMI and Classification");

System.***out***.println("3. Exit");

System.***out***.print("Choose an option: ");

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **void** main(String[] args) {

BMITrackerUtil util = **new** BMITrackerUtil();

BMITracker bmiTracker = **null**;

Scanner sc = **new** Scanner(System.***in***);

**while** (**true**) {

util.menuList();

**int** choice = sc.nextInt();

**switch** (choice) {

**case** 1:

bmiTracker = util.acceptRecord();

**break**;

**case** 2:

**if** (bmiTracker != **null**) {

util.printRecord(bmiTracker);

} **else** {

System.***out***.println("Please enter weight and height first.");

}

**break**;

**case** 3:

System.***out***.println("Exiting...");

sc.close();

System.*exit*(0);

**default**:

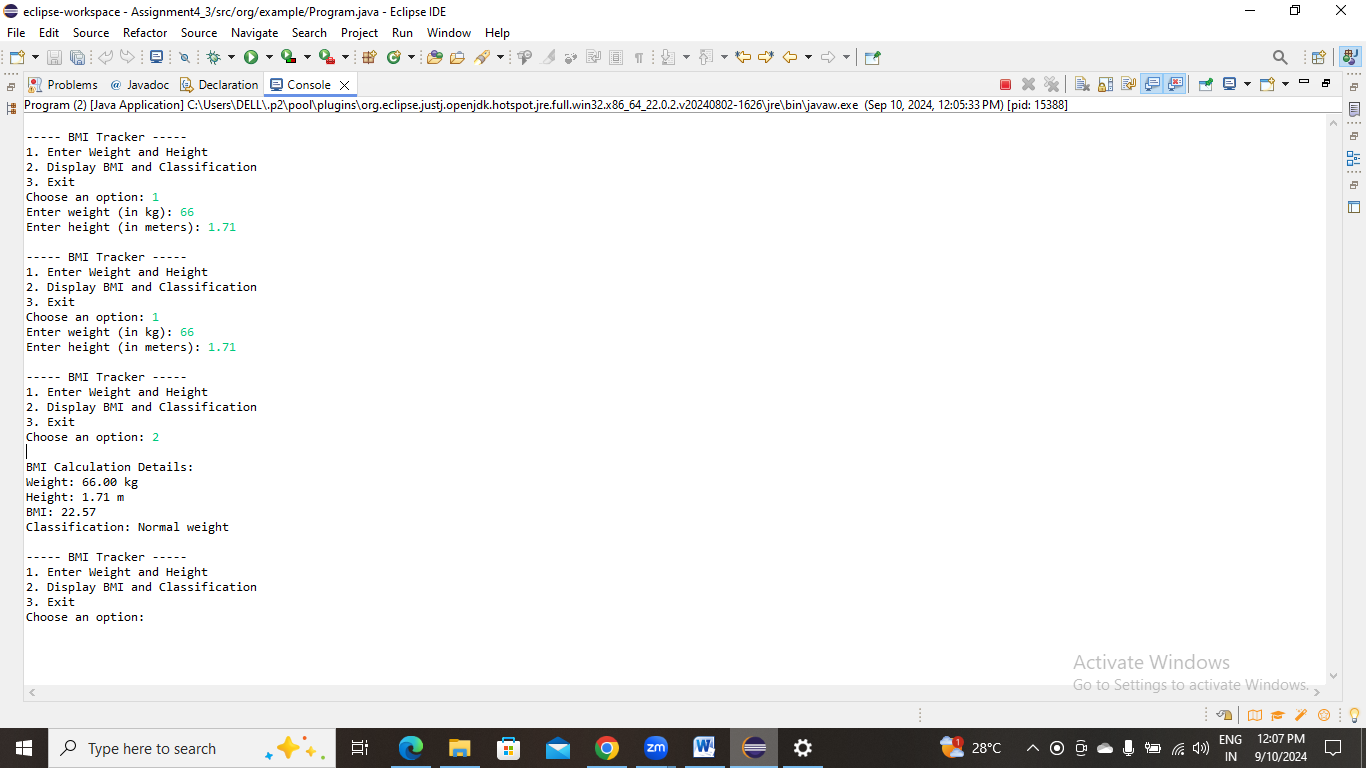
System.***out***.println("Invalid choice. Please try again.");

}

}

}

}



**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 the class DiscountCalculator with fields, an appropriate constructor, getter and setter methods, a toString method, and business logic methods. Define the class DiscountCalculatorUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method to test the functionality of the utility class.

**package** org.example;

**import** java.text.DecimalFormat;

**public** **class** DiscountCalculator {

**private** **double** originalPrice;

**private** **double** discountRate;

**public** DiscountCalculator(**double** originalPrice, **double** discountRate) {

**this**.originalPrice = originalPrice;

**this**.discountRate = discountRate;

}

**public** **double** getOriginalPrice() {

**return** originalPrice;

}

**public** **void** setOriginalPrice(**double** originalPrice) {

**this**.originalPrice = originalPrice;

}

**public** **double** getDiscountRate() {

**return** discountRate;

}

**public** **void** setDiscountRate(**double** discountRate) {

**this**.discountRate = discountRate;

}

**public** **double** calculateDiscountAmount() {

**return** originalPrice \* (discountRate / 100);

}

**public** **double** calculateFinalPrice() {

**return** originalPrice - calculateDiscountAmount();

}

@Override

**public** String toString() {

DecimalFormat df = **new** DecimalFormat("₹###,###.00");

**return** "Original Price: " + df.format(originalPrice) + "\n"

+ "Discount Rate: " + discountRate + "%\n"

+ "Discount Amount: " + df.format(calculateDiscountAmount()) + "\n"

+ "Final Price: " + df.format(calculateFinalPrice()) + "\n";

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** DiscountCalculatorUtil {

**public** DiscountCalculator acceptRecord() {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter Original Price (₹): ");

**double** originalPrice = sc.nextDouble();

System.***out***.print("Enter Discount Percentage (%): ");

**double** discountRate = sc.nextDouble();

**return** **new** DiscountCalculator(originalPrice, discountRate);

}

**public** **void** printRecord(DiscountCalculator calculator) {

System.***out***.println("\nDiscount Calculation Details:");

System.***out***.println(calculator.toString());

}

**public** **void** menuList() {

System.***out***.println("\n----- Discount Calculator -----");

System.***out***.println("1. Enter Price and Discount Details");

System.***out***.println("2. Display Discount Details");

System.***out***.println("3. Exit");

System.***out***.print("Choose an option: ");

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **void** main(String[] args) {

DiscountCalculatorUtil util = **new** DiscountCalculatorUtil();

DiscountCalculator calculator = **null**;

Scanner sc = **new** Scanner(System.***in***);

**while** (**true**) {

util.menuList();

**int** choice = sc.nextInt();

**switch** (choice) {

**case** 1:

calculator = util.acceptRecord();

**break**;

**case** 2:

**if** (calculator != **null**) {

util.printRecord(calculator);

} **else** {

System.***out***.println("Please enter price and discount details first.");

}

**break**;

**case** 3:

System.***out***.println("Exiting...");

sc.close();

System.*exit*(0);

**default**:

System.***out***.println("Invalid choice. Please try again.");

}

}

}

}

**package** org.example;

**public** **class** TollBoothRevenueManager {

**private** **double** carTollRate;

**private** **double** truckTollRate;

**private** **double** motorcycleTollRate;

**private** **int** carCount;

**private** **int** truckCount;

**private** **int** motorcycleCount;

**public** TollBoothRevenueManager(**double** carTollRate, **double** truckTollRate, **double** motorcycleTollRate) {

**this**.carTollRate = carTollRate;

**this**.truckTollRate = truckTollRate;

**this**.motorcycleTollRate = motorcycleTollRate;

**this**.carCount = 0;

**this**.truckCount = 0;

**this**.motorcycleCount = 0;

}

**public** **double** getCarTollRate() {

**return** carTollRate;

}

**public** **void** setCarTollRate(**double** carTollRate) {

**this**.carTollRate = carTollRate;

}

**public** **double** getTruckTollRate() {

**return** truckTollRate;

}

**public** **void** setTruckTollRate(**double** truckTollRate) {

**this**.truckTollRate = truckTollRate;

}

**public** **double** getMotorcycleTollRate() {

**return** motorcycleTollRate;

}

**public** **void** setMotorcycleTollRate(**double** motorcycleTollRate) {

**this**.motorcycleTollRate = motorcycleTollRate;

}

**public** **int** getCarCount() {

**return** carCount;

}

**public** **void** setCarCount(**int** carCount) {

**this**.carCount = carCount;

}

**public** **int** getTruckCount() {

**return** truckCount;

}

**public** **void** setTruckCount(**int** truckCount) {

**this**.truckCount = truckCount;

}

**public** **int** getMotorcycleCount() {

**return** motorcycleCount;

}

**public** **void** setMotorcycleCount(**int** motorcycleCount) {

**this**.motorcycleCount = motorcycleCount;

}

**public** **double** calculateTotalRevenue() {

**return** (carCount \* carTollRate) + (truckCount \* truckTollRate) + (motorcycleCount \* motorcycleTollRate);

}

**public** **int** calculateTotalVehicles() {

**return** carCount + truckCount + motorcycleCount;

}

@Override

**public** String toString() {

**return** String.*format*(

"Cars: %d | Trucks: %d | Motorcycles: %d\nTotal Vehicles: %d\nTotal Revenue: ₹%.2f",

carCount, truckCount, motorcycleCount, calculateTotalVehicles(), calculateTotalRevenue()

);

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** TollBoothRevenueManagerUtil {

**public** TollBoothRevenueManager acceptRecord(TollBoothRevenueManager manager) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter number of cars: ");

**int** carCount = sc.nextInt();

System.***out***.print("Enter number of trucks: ");

**int** truckCount = sc.nextInt();

System.***out***.print("Enter number of motorcycles: ");

**int** motorcycleCount = sc.nextInt();

manager.setCarCount(carCount);

manager.setTruckCount(truckCount);

manager.setMotorcycleCount(motorcycleCount);

**return** manager;

}

**public** **void** printRecord(TollBoothRevenueManager manager) {

System.***out***.println("\nToll Booth Revenue Details:");

System.***out***.println(manager.toString());

}

**public** **void** menuList() {

System.***out***.println("\n----- Toll Booth Manager -----");

System.***out***.println("1. Enter Vehicle Counts");

System.***out***.println("2. Display Total Revenue and Vehicle Counts");

System.***out***.println("3. Exit");

System.***out***.print("Choose an option: ");

}

}

**package** org.example;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **void** main(String[] args) {

**double** carRate = 50.0;

**double** truckRate = 100.0;

**double** motorcycleRate = 30.0;

TollBoothRevenueManager manager = **new** TollBoothRevenueManager(carRate, truckRate, motorcycleRate);

TollBoothRevenueManagerUtil util = **new** TollBoothRevenueManagerUtil();

Scanner sc = **new** Scanner(System.***in***);

**while** (**true**) {

util.menuList();

**int** choice = sc.nextInt();

**switch** (choice) {

**case** 1:

util.acceptRecord(manager);

**break**;

**case** 2:

util.printRecord(manager);

**break**;

**case** 3:

System.***out***.println("Exiting...");

sc.close();

System.*exit*(0);

**default**:

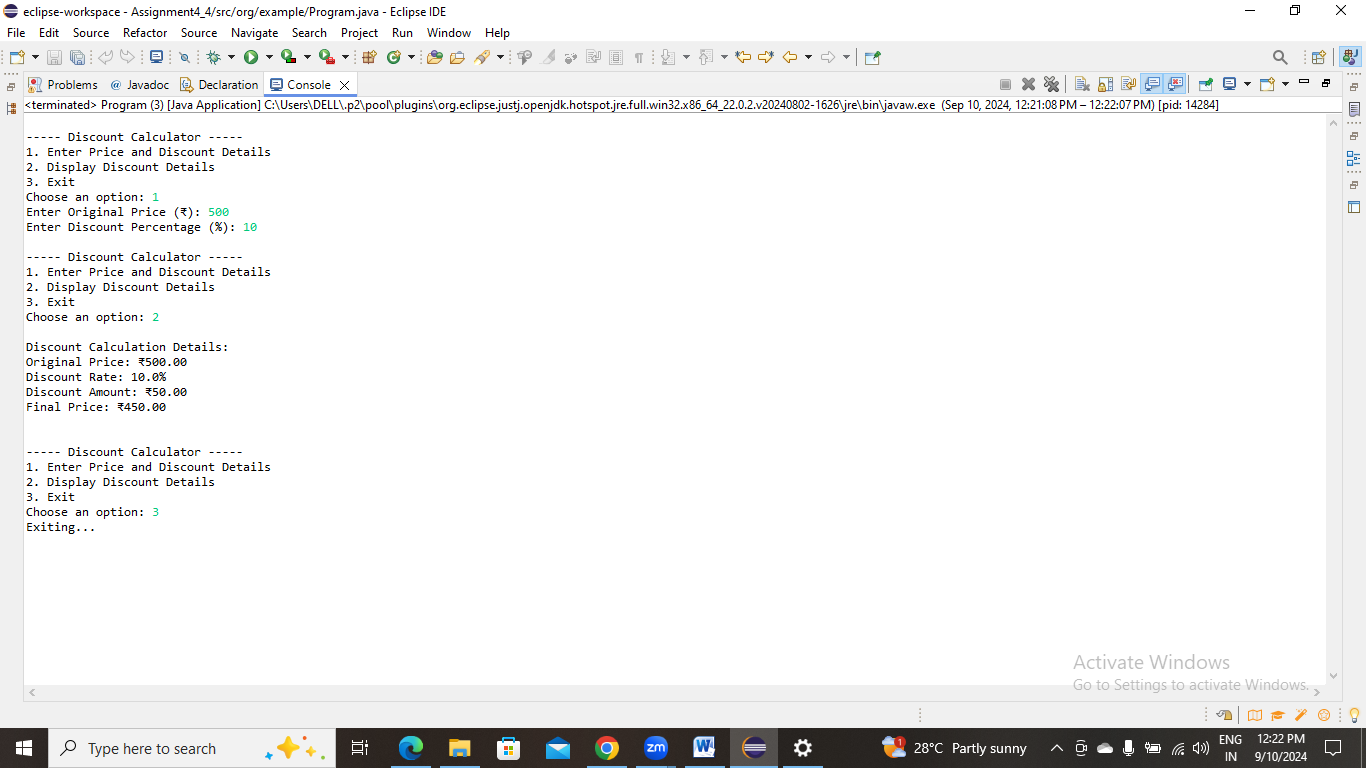
System.***out***.println("Invalid choice. Please try again.");

}

}

}

}



**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 the class TollBoothRevenueManager with fields, an appropriate constructor, getter and setter methods, a toString method, and business logic methods. Define the class TollBoothRevenueManagerUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method to test the functionality of the utility class.

