ASSIGNMENT-4

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

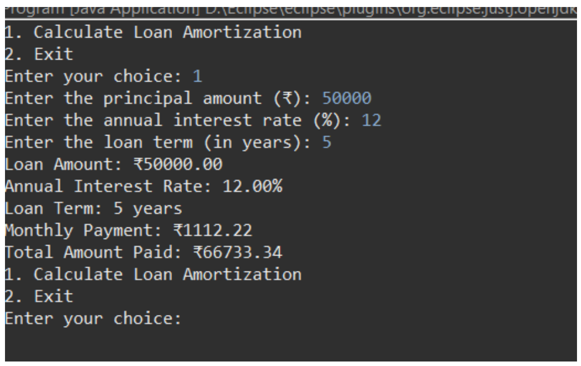
Implement a system to calculate and display the monthly payments for a mortgage loan. The system should:
1. Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
2. Calculate the monthly payment using the standard mortgage formula:
o Monthly Payment 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 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 com.org.LoanAmortization;
public class Program {
	public static void main(String[] args) {
		int choice;
		do {
			LoanAmortizationCalculatorUtil.menuList();
			choice = LoanAmortizationCalculatorUtil.getMenuChoice();
			switch (choice) {
			case 1:
				LoanAmortizationCalculator loan =
LoanAmortizationCalculatorUtil.acceptRecord();
				LoanAmortizationCalculatorUtil.printRecord(loan);
				break;
```

```
                        case 2:
                                System.out.println("Exiting...");
                                break;
                        default:
                                System.out.println("Invalid choice. Please try again.");
                        }
                } while (choice != 2);
        }
}
```

```
package com.org.LoanAmortization;
public class LoanAmortizationCalculator {
        private double principal;
        private double annualInterestRate;
        private int loanTerm;
        // Constructor
        public LoanAmortizationCalculator(double principal, double annualInterestRate, int loanTerm) {
                this.principal = principal;
                this.annualInterestRate = annualInterestRate;
                this.loanTerm = loanTerm;
        }
        // Getters and Setters
        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 calculateTotalAmountPaid() {
                return calculateMonthlyPayment() * loanTerm * 12;
        }

        public String toString() {
                return String.format(
                                "Loan Amount: ₹%.2f\nAnnual Interest Rate: %.2f%%\nLoan Term: %d
years\n"
                                                + "Monthly Payment: ₹%.2f\nTotal Amount Paid: ₹%.2f",
                                principal, annualInterestRate, loanTerm, calculateMonthlyPayment(),
calculateTotalAmountPaid());
        }
}
package com.org.LoanAmortization;
import java.util.Scanner;
public class LoanAmortizationCalculatorUtil {
        private static Scanner scanner = new Scanner(System.in);
        public static LoanAmortizationCalculator acceptRecord() {
                System.out.print("Enter the principal amount (₹): ");
                double principal = scanner.nextDouble();
                System.out.print("Enter the annual interest rate (%): ");
                double annualInterestRate = scanner.nextDouble();
                System.out.print("Enter the loan term (in years): ");
                int loanTerm = scanner.nextInt();
                return new LoanAmortizationCalculator(principal, annualInterestRate, loanTerm);
        }
        public static void printRecord(LoanAmortizationCalculator loan) {
                System.out.println(loan.toString());
        }
        public static void menuList() {
                System.out.println("1. Calculate Loan Amortization");
                System.out.println("2. Exit");
        }
        public static int getMenuChoice() {
                System.out.print("Enter your choice: ");
                return scanner.nextInt();
        }
}
```



```
Program [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk
1. Calculate Loan Amortization
2. Exit
Enter your choice: 1
Enter the principal amount (₹): 50000
Enter the annual interest rate (%): 12
Enter the loan term (in years): 5
Loan Amount: ₹50000.00
Annual Interest Rate: 12.00%
Loan Term: 5 years
Monthly Payment: ₹1112.22
Total Amount Paid: ₹66733.34
1. Calculate Loan Amortization
2. Exit
Enter your choice:
```

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:

o Future Value Calculation:
futureValue = principal * (1 + annualInterestRate / numberOfCompounds)^(numberOfCompounds * years)

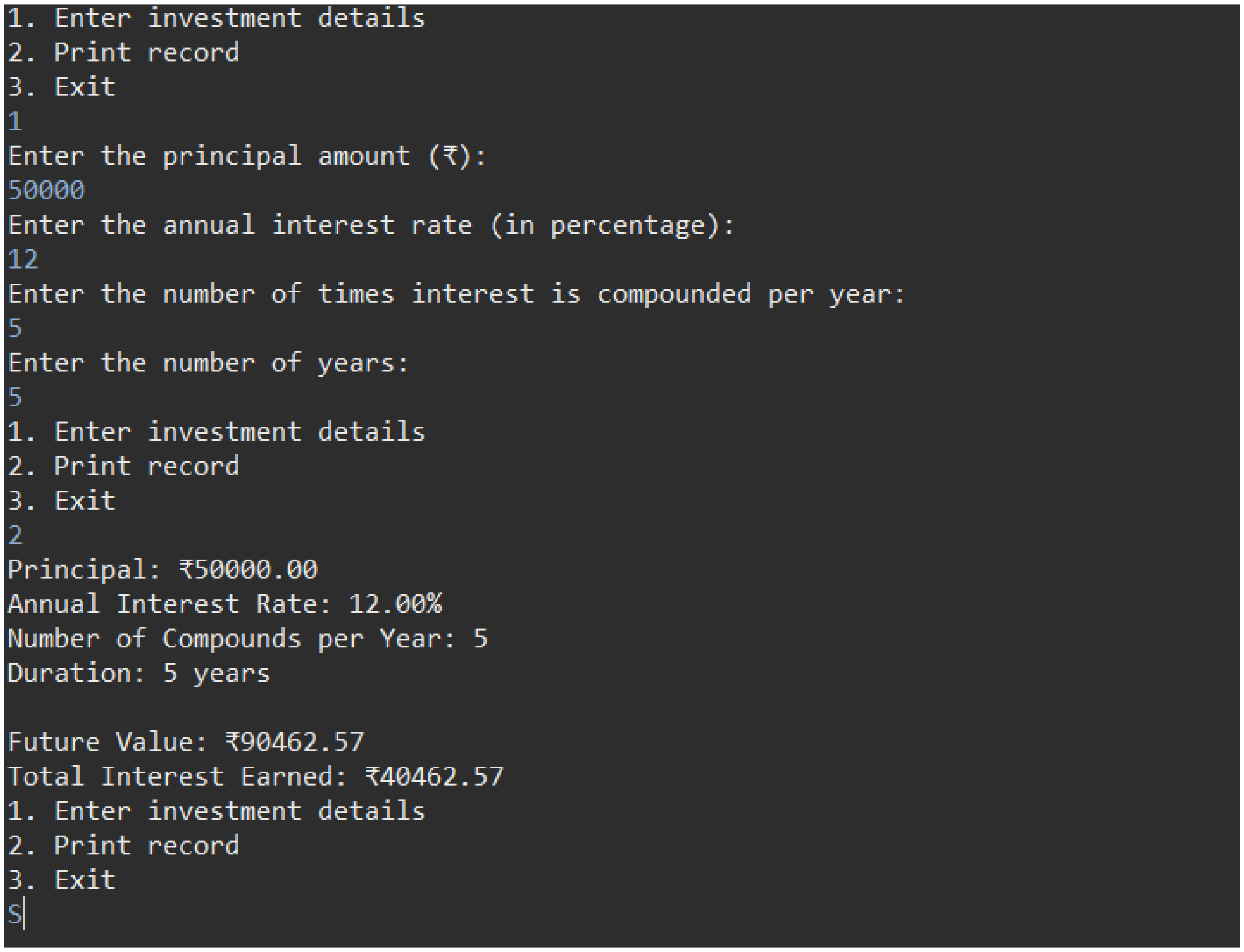
o Total Interest Earned: totalInterest = futureValue - principal

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 com.org.CompoundInterestCalculator;
public class CompoundInterestCalculator {
        private double principal;
  private double annualInterestRate;
  private int numberOfCompounds;
  private int years;
  // Constructor
  public CompoundInterestCalculator(double principal, double annualInterestRate, int
numberOfCompounds, int years) {
      this.principal = principal;
      this.annualInterestRate = annualInterestRate;
      this.numberOfCompounds = numberOfCompounds;
      this.years = years;
  }
  // Getters and Setters
  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) {
```

```java
        this.numberOfCompounds = numberOfCompounds;
    }
    public int getYears() {
        return years;
    }
    public void setYears(int years) {
        this.years = years;
    }
    // calculate future value
    public double calculateFutureValue() {
        return principal * Math.pow(1 + annualInterestRate / numberOfCompounds, numberOfCompounds *
years);
    }
    // calculate total interest earned
    public double calculateTotalInterest() {
        return calculateFutureValue() - principal;
    }
    // toString method

    public String toString() {
        return String.format("Principal: ₹%.2f\nAnnual Interest Rate: %.2f%%\nNumber of Compounds per
Year: %d\nDuration: %d years\n",
                             principal, annualInterestRate * 100, numberOfCompounds, years);
    }
}
package com.org.CompoundInterestCalculator;
import java.util.Scanner;
public class CompoundInterestCalculatorUtil {
        private static Scanner scanner = new Scanner(System.in);
        // accept investment details from user
        public static CompoundInterestCalculator acceptRecord() {
                System.out.println("Enter the principal amount (₹): ");
                double principal = scanner.nextDouble();
                System.out.println("Enter the annual interest rate (in percentage): ");
                double annualInterestRate = scanner.nextDouble() / 100;
                System.out.println("Enter the number of times interest is compounded per year: ");
                int numberOfCompounds = scanner.nextInt();
                System.out.println("Enter the number of years: ");
                int years = scanner.nextInt();
                return new CompoundInterestCalculator(principal, annualInterestRate,
numberOfCompounds, years);
        }
        // print the investment record and calculations
        public static void printRecord(CompoundInterestCalculator calculator) {
                System.out.println(calculator);
                double futureValue = calculator.calculateFutureValue();
                double totalInterest = calculator.calculateTotalInterest();
                System.out.printf("Future Value: ₹%.2f\n", futureValue);
                System.out.printf("Total Interest Earned: ₹%.2f\n", totalInterest);
    }
    // display menu options
    public static void menuList() {
        System.out.println("1. Enter investment details");
        System.out.println("2. Print record");
        System.out.println("3. Exit");
    }
    // Getter for the Scanner instance
    public static Scanner getScanner() {
  return scanner;
}}
package com.org.CompoundInterestCalculator;
public class Program {
    public static void main(String[] args) {
        CompoundInterestCalculator calculator = null;
        boolean running = true;
        while (running) {
            CompoundInterestCalculatorUtil.menuList();
            int choice = CompoundInterestCalculatorUtil.getScanner().nextInt();
            switch (choice) {
            case 1:
                calculator = CompoundInterestCalculatorUtil.acceptRecord();
                break;
            case 2:
                if (calculator != null) {
                    CompoundInterestCalculatorUtil.printRecord(calculator);
                } else {
                    System.out.println("No investment details available. Please
enter details first.");
                }
                break;
            case 3:
                running = false;
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```



```
1. Enter investment details
2. Print record
3. Exit
1
Enter the principal amount (₹):
50000
Enter the annual interest rate (in percentage):
12
Enter the number of times interest is compounded per year:
5
Enter the number of years:
5
1. Enter investment details
2. Print record
3. Exit
2
Principal: ₹50000.00
Annual Interest Rate: 12.00%
Number of Compounds per Year: 5
Duration: 5 years

Future Value: ₹90462.57
Total Interest Earned: ₹40462.57
1. Enter investment details
2. Print record
3. Exit
S
```

3.BMI (Body Mass Index) Tracker

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

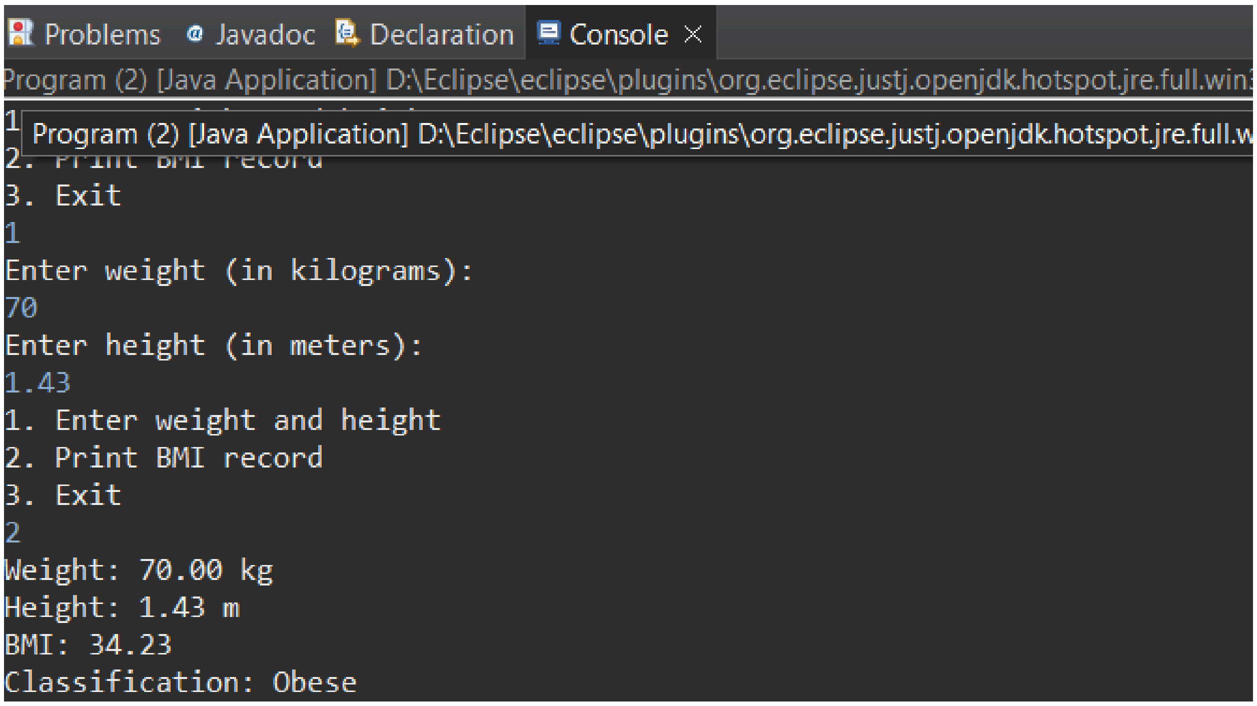
1. Accept weight (in kilograms) and height (in meters) from the user.
2. Calculate the BMI using the formula:
   - **BMI Calculation:** `BMI = weight / (height * height)`
3. Classify the BMI into one of the following categories:
   - Underweight: BMI < 18.5
   - Normal weight: 18.5 ≤ BMI < 24.9
   - 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 com.org.BMITracker;

public class BMITracker {
	private double weight;
	private double height;
	// Constructor
	public BMITracker(double weight, double height) {
		this.weight = weight;
		this.height = height;
	}

	// Getters and Setters
	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;
	}
	// Method to calculate BMI
	public double calculateBMI() {
		return weight / (height * height);
	}
	// Method to classify BMI
	public String classifyBMI() {
		double bmi = calculateBMI();
		if (bmi < 18.5) {
			return "Underweight";
		} else if (bmi < 24.9) {
			return "Normal weight";
		} else if (bmi < 29.9) {
			return "Overweight";
		} else {
			return "Obese";
		}
	}
	// toString method

	public String toString() {
			return String.format("Weight: %.2f kg\nHeight: %.2f m\nBMI: %.2f\nClassification: %s",
weight, height,
					calculateBMI(), classifyBMI());
		}
}
```

```
package com.org.BMITracker;
import java.util.Scanner;
public class BMITrackerUtil {
	private static Scanner scanner = new Scanner(System.in);
	// accept weight and height from user
	public static BMITracker acceptRecord() {
		System.out.println("Enter weight (in kilograms): ");
		double weight = scanner.nextDouble();
		System.out.println("Enter height (in meters): ");
		double height = scanner.nextDouble();
		return new BMITracker(weight, height);
	}
	// print the BMI record and classification
	public static void printRecord(BMITracker tracker) {
		System.out.println(tracker);
	}
	// Method to display menu options
	public static void menuList() {
		System.out.println("1. Enter weight and height");
		System.out.println("2. Print BMI record");
		System.out.println("3. Exit");
	}
	// Getter for the Scanner instance
	public static Scanner getScanner() {
		return scanner;
	}
}
```

```
package com.org.BMITracker;
public class Program {
	public static void main(String[] args) {
    BMITracker tracker = null;
    boolean running = true;
    while (running) {
      BMITrackerUtil.menuList(); // Display menu
      int choice = BMITrackerUtil.getScanner().nextInt(); // Get user choice
      switch (choice) {
        case 1:
          tracker = BMITrackerUtil.acceptRecord(); // Accept new record
          break;
        case 2:
          if (tracker != null) {
            BMITrackerUtil.printRecord(tracker); // Print the record
          } else {
            System.out.println("No BMI record available. Please enter details first.");
          }
          break;
        case 3:
          running = false; // Exit loop
          System.out.println("Exiting...");
          break;
        default:
          System.out.println("Invalid choice. Please try again.");
      }
    }
  }
}
```



```
Program (2) [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win
1  Program (2) [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
2. Print BMI record
3. Exit
1
Enter weight (in kilograms):
70
Enter height (in meters):
1.43
1. Enter weight and height
2. Print BMI record
3. Exit
2
Weight: 70.00 kg
Height: 1.43 m
BMI: 34.23
Classification: Obese
```

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:

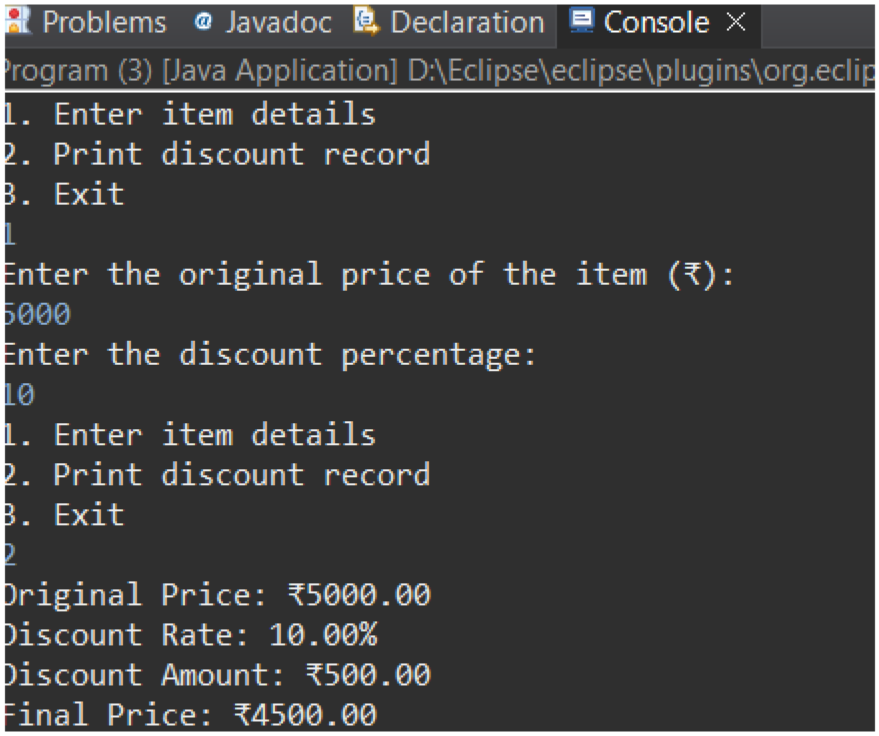
   - **Discount Amount Calculation:** `discountAmount = originalPrice * (discountRate / 100)`
   - **Final Price Calculation:** `finalPrice = originalPrice - discountAmount`
3. Display the discount amount and the final price of the item, in Indian Rupees (₹).

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 com.org.Discount;

public class DiscountCalculator {
          private double originalPrice;
  private double discountRate;
  // Constructor
  public DiscountCalculator(double originalPrice, double discountRate) {
     this.originalPrice = originalPrice;
     this.discountRate = discountRate;
  }
  // Getters and Setters
  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;
  }
  //calculate discount amount
  public double calculateDiscountAmount() {
     return originalPrice * (discountRate / 100);
  }
  //calculate final price
  public double calculateFinalPrice() {
     return originalPrice - calculateDiscountAmount();
  }
  // toString method

  public String toString() {
     return String.format("Original Price: ₹%.2f\nDiscount Rate: %.2f%%\nDiscount Amount: ₹%.2f\nFinal
Price: ₹%.2f",
                    originalPrice, discountRate, calculateDiscountAmount(), calculateFinalPrice());
  }}
```

```java
package com.org.Discount;
import java.util.Scanner;
public class DiscountCalculatorUtil {
            private static Scanner scanner = new Scanner(System.in);
  // Method to accept original price and discount rate from the user
  public static DiscountCalculator acceptRecord() {
      System.out.println("Enter the original price of the item (₹): ");
      double originalPrice = scanner.nextDouble();
      System.out.println("Enter the discount percentage: ");
      double discountRate = scanner.nextDouble();
      return new DiscountCalculator(originalPrice, discountRate);
  }
  // Method to print the discount details
  public static void printRecord(DiscountCalculator calculator) {
      System.out.println(calculator);
  }
  // Method to display menu options
  public static void menuList() {
      System.out.println("1. Enter item details");
      System.out.println("2. Print discount record");
      System.out.println("3. Exit");
  }
  // Getter for the Scanner instance
  public static Scanner getScanner() {
      return scanner;
  }
}


package com.org.Discount;
public class Program {
            public static void main(String[] args) {
      DiscountCalculator calculator = null;
      boolean running = true;

      // program running until the user decides to exit
      while (running) {
          // Display the menu options
          DiscountCalculatorUtil.menuList();

          // Read the
          int choice = DiscountCalculatorUtil.getScanner().nextInt();

          // Process
          switch (choice) {
              case 1:
                  // Accept details for the discount
                  calculator = DiscountCalculatorUtil.acceptRecord();
                  break;
```

```java
                case 2:
                    if (calculator != null) {
                        DiscountCalculatorUtil.printRecord(calculator);
                    } else {
                        System.out.println("No discount record available. Please enter details first.");
                    }
                    break;
                case 3:
                    // Exit the loop and terminate the program
                    running = false;
                    System.out.println("Exiting...");
                    break;
                default:
                    // Handle invalid choices
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```



```
rogram (3) [Java Application] D:\Eclipse\eclipse\plugins\org.eclip
1. Enter item details
2. Print discount record
3. Exit
1
Enter the original price of the item (₹):
5000
Enter the discount percentage:
10
1. Enter item details
2. Print discount record
3. Exit
2
Original Price: ₹5000.00
Discount Rate: 10.00%
Discount Amount: ₹500.00
Final Price: ₹4500.00
```

5. Toll Booth Revenue Management

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

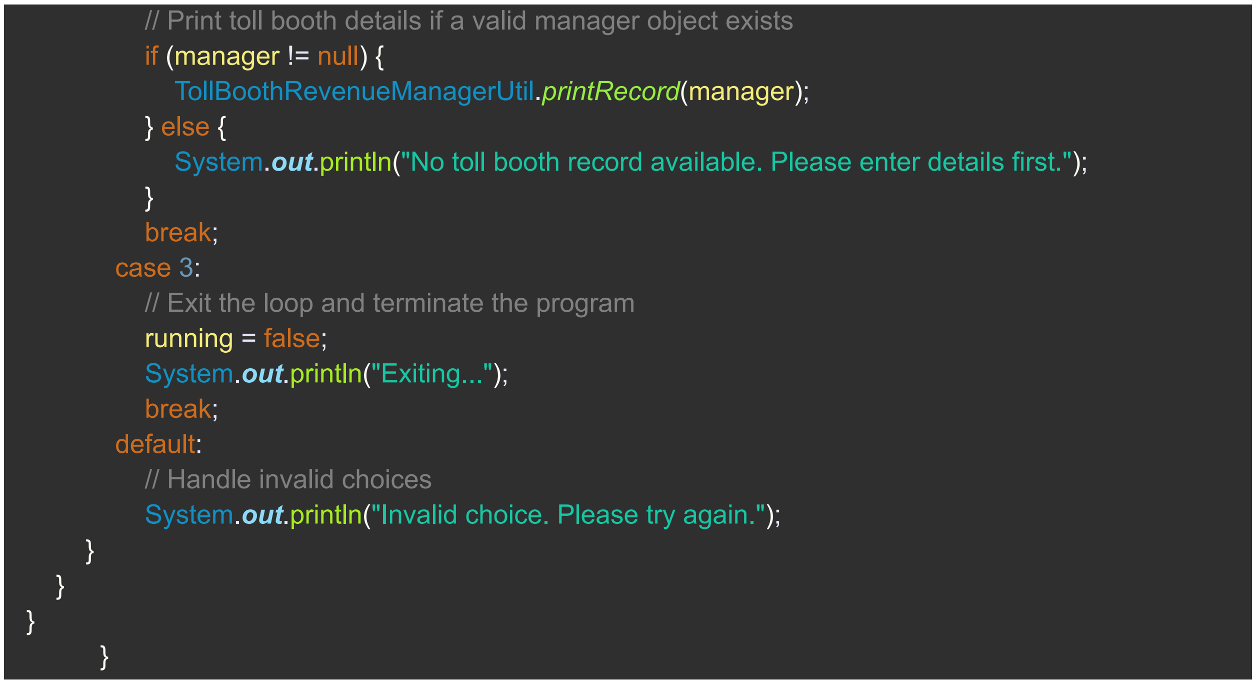
1. Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.
2. Accept the number of vehicles of each type passing through the toll booth.
3. Calculate the total revenue based on the toll rates and number of vehicles.
4. Display the total number of vehicles and the total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

- **Toll Rate Examples:**
    - Car: ₹50.00
    - Truck: ₹100.00
    - Motorcycle: ₹30.00

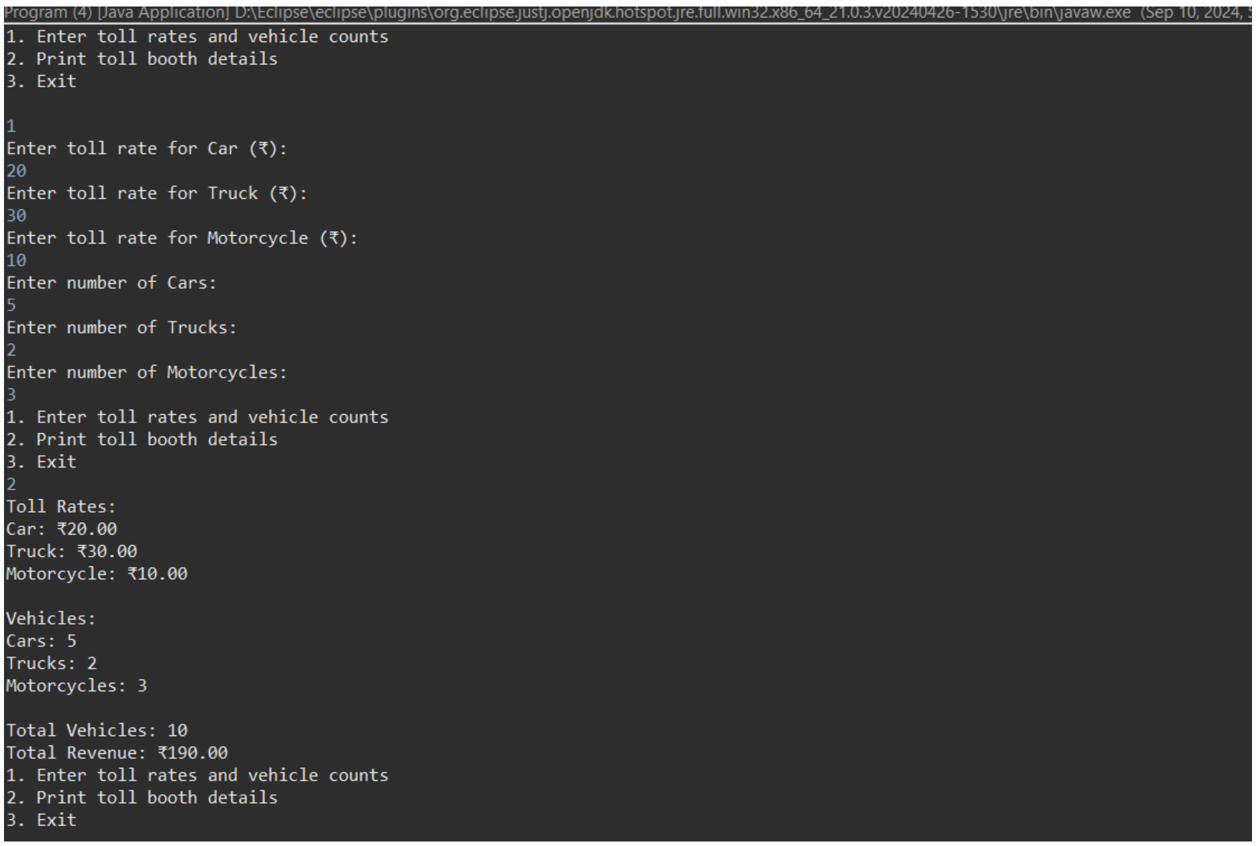
Define the class `TollBoothRevenueManager` with fields, an appropriate constructor, getter and setter methods, a `toString` method, and business logic methods. Define the class `TollBoothRevenueManagerUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with a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

```
 package com.org.TollBoothRevenueManager;
```

```
public class TollBoothRevenueManager {
            private double carRate;
  private double truckRate;
  private double motorcycleRate;
  private int numCars;
  private int numTrucks;
  private int numMotorcycles;
  // Constructor
  public TollBoothRevenueManager(double carRate, double truckRate, double motorcycleRate) {
     this.carRate = carRate;
     this.truckRate = truckRate;
     this.motorcycleRate = motorcycleRate;
     this.numCars = 0;
     this.numTrucks = 0;
     this.numMotorcycles = 0;
  }
  // Getters and Setters
  public double getCarRate() {
     return carRate;
  }
  public void setCarRate(double carRate) {
     this.carRate = carRate;
  }
  public double getTruckRate() {
     return truckRate;
  }
  public void setTruckRate(double truckRate) {
     this.truckRate = truckRate;
  }
  public double getMotorcycleRate() {
      return motorcycleRate;
  }
  public void setMotorcycleRate(double motorcycleRate) {
      this.motorcycleRate = motorcycleRate;
  }
  public int getNumCars() {
      return numCars;
  }
  public void setNumCars(int numCars) {
      this.numCars = numCars;
  }
  public int getNumTrucks() {
      return numTrucks;
  }
  public void setNumTrucks(int numTrucks) {
      this.numTrucks = numTrucks;
  }
  public int getNumMotorcycles() {
      return numMotorcycles;
  }
  public void setNumMotorcycles(int numMotorcycles) {
      this.numMotorcycles = numMotorcycles;
  }
  // Method to calculate total revenue
  public double calculateTotalRevenue() {
      return (numCars * carRate) + (numTrucks * truckRate) + (numMotorcycles * motorcycleRate);
  }
  // Method to calculate total number of vehicles
  public int calculateTotalVehicles() {
      return numCars + numTrucks + numMotorcycles;
  }
  // toString method to display the toll booth details

  public String toString() {
      return String.format("Toll Rates:\nCar: ₹%.2f\nTruck: ₹%.2f\nMotorcycle: ₹%.2f\n\n" +
                      "Vehicles:\nCars: %d\nTrucks: %d\nMotorcycles: %d\n\n" +
                      "Total Vehicles: %d\nTotal Revenue: ₹%.2f",
                      carRate, truckRate, motorcycleRate,
                      numCars, numTrucks, numMotorcycles,
                      calculateTotalVehicles(), calculateTotalRevenue());
  }
}
package com.org.TollBoothRevenueManager;
import java.util.Scanner;
public class TollBoothRevenueManagerUtil {
        private static Scanner scanner = new Scanner(System.in);
  // accept toll rates and vehicle counts from the user
  public static TollBoothRevenueManager acceptRecord() {
      System.out.println("Enter toll rate for Car (₹): ");
      double carRate = scanner.nextDouble();
      System.out.println("Enter toll rate for Truck (₹): ");
      double truckRate = scanner.nextDouble();
      System.out.println("Enter toll rate for Motorcycle (₹): ");
      double motorcycleRate = scanner.nextDouble();
      TollBoothRevenueManager manager = new TollBoothRevenueManager(carRate, truckRate,
motorcycleRate);
      System.out.println("Enter number of Cars: ");
      int numCars = scanner.nextInt();
      manager.setNumCars(numCars);
      System.out.println("Enter number of Trucks: ");
      int numTrucks = scanner.nextInt();
      manager.setNumTrucks(numTrucks);
      System.out.println("Enter number of Motorcycles: ");
      int numMotorcycles = scanner.nextInt();
      manager.setNumMotorcycles(numMotorcycles);
      return manager;
   }
   // print the toll booth details
   public static void printRecord(TollBoothRevenueManager manager) {
      System.out.println(manager);
   }
   // display menu options
   public static void menuList() {
      System.out.println("1. Enter toll rates and vehicle counts");
      System.out.println("2. Print toll booth details");
      System.out.println("3. Exit");
   }
   // Getter for the Scanner instance
   public static Scanner getScanner() {
      return scanner;
   }
}
package com.org.TollBoothRevenueManager;
public class Program {
            public static void main(String[] args) {
                        TollBoothRevenueManager manager = null;
      boolean running = true;
      while (running) {
         TollBoothRevenueManagerUtil.menuList();
         int choice = TollBoothRevenueManagerUtil.getScanner().nextInt();

         switch (choice) {
            case 1:
               // Accept toll rates and vehicle counts
               manager = TollBoothRevenueManagerUtil.acceptRecord();
               break;
            case 2:
```



```java
                // Print toll booth details if a valid manager object exists
                if (manager != null) {
                    TollBoothRevenueManagerUtil.printRecord(manager);
                } else {
                    System.out.println("No toll booth record available. Please enter details first.");
                }
                break;
            case 3:
                // Exit the loop and terminate the program
                running = false;
                System.out.println("Exiting...");
                break;
            default:
                // Handle invalid choices
                System.out.println("Invalid choice. Please try again.");
        }
    }
}
      }
```



```
Program (4) [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (Sep 10, 2024,
1. Enter toll rates and vehicle counts
2. Print toll booth details
3. Exit

1
Enter toll rate for Car (₹):
20
Enter toll rate for Truck (₹):
30
Enter toll rate for Motorcycle (₹):
10
Enter number of Cars:
5
Enter number of Trucks:
2
Enter number of Motorcycles:
3
1. Enter toll rates and vehicle counts
2. Print toll booth details
3. Exit
2
Toll Rates:
Car: ₹20.00
Truck: ₹30.00
Motorcycle: ₹10.00

Vehicles:
Cars: 5
Trucks: 2
Motorcycles: 3

Total Vehicles: 10
Total Revenue: ₹190.00
1. Enter toll rates and vehicle counts
2. Print toll booth details
3. Exit
```