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:
    - 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.

Solution:

Class File 1:

```
package com.loanamortization;
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;
    }

    public String toString() {
        return "Principal: ₹" + principal +
                ", Annual Interest Rate: " + annualInterestRate + "%" +
                ", Loan Term: " + loanTerm + " years";
    }
}
```

Class File 2:

```
package com.loanamortization;
import java.util.Scanner;

public class LoanAmortizationCalculatorUtil {
    private LoanAmortizationCalculator calculator;

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

        System.out.print("Enter loan amount (₹): ");
        double principal = sc.nextDouble();
        System.out.print("Enter annual interest rate (%): ");
        double annualInterestRate = sc.nextDouble();
        System.out.print("Enter loan term (in years): ");
        int loanTerm = sc.nextInt();

        calculator = new LoanAmortizationCalculator(principal,
annualInterestRate, loanTerm);
    }
```

```
    public void printRecord() {
        if (calculator != null) {
            double monthlyPayment = calculator.calculateMonthlyPayment();
            double totalPayment = calculator.calculateTotalPayment();
            System.out.println("\nLoan Details:");
            System.out.println(calculator.toString());
            System.out.printf("Monthly Payment: ₹%.2f\n", monthlyPayment);
            System.out.printf("Total Payment over %d years: ₹%.2f\n",
calculator.getLoanTerm(), totalPayment);
        } else {
            System.out.println("No loan record found. Please enter loan details
first.");
        }
    }

    public void menuList() {
        Scanner sc = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\n=== Loan Amortization Calculator Menu ===");
            System.out.println("1. Enter Loan Details");
            System.out.println("2. Display Loan Information");
            System.out.println("0. Exit");
            System.out.print("Enter choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    acceptRecord();
                    break;
                case 2:
                    printRecord();
                    break;
                case 0:
                    System.out.println("Thank you for using the Loan Amortization
Calculator. Goodbye!");
                    break;
```

```
            default:
                System.out.println("Invalid choice, please try again.");
        }
    } while (choice != 0);
    sc.close();
  }
}
```

Class File 3:

```
package com.loanamortization;

public class Program {
  public static void main(String[] args) {
    LoanAmortizationCalculatorUtil util = new
LoanAmortizationCalculatorUtil();
    util.menuList();
  }
}
```

```
=== Loan Amortization Calculator Menu ===
1. Enter Loan Details
2. Display Loan Information
0. Exit
Enter choice: 1
Enter loan amount (₹): 500000
Enter annual interest rate (%): 7.5
Enter loan term (in years): 3

=== Loan Amortization Calculator Menu ===
1. Enter Loan Details
2. Display Loan Information
0. Exit
Enter choice: 2

Loan Details:
Principal: ₹500000.0, Annual Interest Rate: 7.5%, Loan Term: 3 years
Monthly Payment: ₹15553.11
Total Payment over 3 years: ₹559911.93
```

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.

Solution:

Class File 1:

```
package com.compoundinterest;

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

   public double calculateTotalInterest() {
      return calculateFutureValue() - principal;
   }
```

```
    public String toString() {
        return "Principal: ₹" + principal +
                "\nAnnual Interest Rate: " + annualInterestRate + "%" +
                "\nNumber of Compounds per Year: " + numberOfCompounds +
                "\nInvestment Duration: " + 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) {
this.numberOfCompounds = numberOfCompounds; }

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

Class File 2:

```
package com.compoundinterest;

import java.util.Scanner;

public class CompoundInterestCalculatorUtil {
    private CompoundInterestCalculator calculator;

    public void acceptRecord(Scanner sc) {
        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 the interest is compounded per
year: ");
        int numberOfCompounds = sc.nextInt();

        System.out.print("Enter investment duration (in years): ");
        int years = sc.nextInt();

        calculator = new CompoundInterestCalculator(principal,
annualInterestRate, numberOfCompounds, years);

        sc.nextLine();
    }

    public void printRecord() {

        if (calculator != null) {
            double futureValue = calculator.calculateFutureValue();
            double totalInterest = calculator.calculateTotalInterest();
            System.out.println("\nInvestment Details:");
            System.out.println(calculator.toString());
            System.out.printf("Future Value: ₹%.2f\n", futureValue);
            System.out.printf("Total Interest Earned: ₹%.2f\n", totalInterest);
        }
        else {
            System.out.println("No investment record found. Please enter
investment details first.");
        }
    }

    public void menuList() {
        Scanner sc = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\n=== Compound Interest Calculator Menu ===");
            System.out.println("1. Enter Investment Details");
```

```
            System.out.println("2. Display Investment Information");
            System.out.println("0. Exit");
            System.out.print("Enter choice: ");

            while (!sc.hasNextInt()) {
               System.out.println("Invalid input. Please enter a valid choice.");
               sc.next();
            }

            choice = sc.nextInt();

            sc.nextLine();

            switch (choice) {
               case 1:
                  acceptRecord(sc);
                  break;
               case 2:
                  printRecord();
                  break;
               case 0:
                  System.out.println("Exiting...");
                  break;
               default:
                  System.out.println("Invalid choice, try again.");
            }
         } while (choice != 0);
         sc.close();
      }
}
```

Class File 3:

```
package com.compoundinterest;

public class Program {
   public static void main(String[] args) {
```

```
        CompoundInterestCalculatorUtil util = new
CompoundInterestCalculatorUtil();
        util.menuList();
    }
}
```

```
=== Compound Interest Calculator Menu ===
1. Enter Investment Details
2. Display Investment Information
0. Exit
Enter choice: 1
Enter initial investment amount (₹): 700000
Enter annual interest rate (%): 11.5
Enter number of times the interest is compounded per year: 4
Enter investment duration (in years): 10

=== Compound Interest Calculator Menu ===
1. Enter Investment Details
2. Display Investment Information
0. Exit
Enter choice: 2

Investment Details:
Principal: ₹700000.0
Annual Interest Rate: 11.5%
Number of Compounds per Year: 4
Investment Duration: 10 years
Future Value: ₹23766559328384297000000000000.00
Total Interest Earned: ₹23766559328384297000000000000.00
```

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.

Soultion:

Class File 1:

```
package com.bmicalculator;

public class BMITracker {
    private double weight;
    private double height;

    public BMITracker(double weight, double height) {
        this.weight = weight;
        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 < 24.9) {
            return "Normal weight";
        } else if (bmi < 29.9) {
            return "Overweight";
        } else {
            return "Obese";
        }
    }

    public String toString() {
        return "Weight: " + weight + " kg\n" +
               "Height: " + height + " meters\n" +
               "BMI: " + String.format("%.2f", calculateBMI()) + "\n" +
               "Classification: " + classifyBMI();
```

```
    }

    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; }
}
```

Class File 2:

```
package com.bmicalculator;

import java.util.Scanner;

public class BMITrackerUtil {
    private BMITracker tracker;

    public void acceptRecord() {

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter weight (in kilograms): ");
        double weight = sc.nextDouble();
        System.out.print("Enter height (in meters): ");
        double height = sc.nextDouble();
        tracker = new BMITracker(weight, height);
    }

    public void printRecord() {

        if (tracker != null) {
            System.out.println("\nBMI Details:");
            System.out.println(tracker.toString());
        }
        else {
            System.out.println("No BMI record found. Please enter your details
first.");
        }
```

```
}

public void menuList() {

   Scanner sc = new Scanner(System.in);
   int choice;

   do {
      System.out.println("\n=== BMI Tracker Menu ===");
      System.out.println("1. Enter Details");
      System.out.println("2. Display BMI Information");
      System.out.println("0. Exit");
      System.out.print("Enter choice: ");

      while (!sc.hasNextInt()) {
         System.out.println("Invalid input. Please enter a valid choice.");
         sc.next();
      }

      choice = sc.nextInt();
      sc.nextLine();

      switch (choice) {
         case 1:
            acceptRecord();
            break;
         case 2:
            printRecord();
            break;
         case 0:
            System.out.println("Exiting...");
            break;
         default:
            System.out.println("Invalid choice, try again.");
      }
   } while (choice != 0);
   sc.close();
```

```
    }
}
```

Class file 3:

```
package com.bmicalculator;

public class Program {
    public static void main(String[] args) {
        BMITrackerUtil util = new BMITrackerUtil();
        util.menuList();
    }
}
```

```
=== BMI Tracker Menu ===
1. Enter Details
2. Display BMI Information
0. Exit
Enter choice: 1
Enter weight (in kilograms): 45
Enter height (in meters): 1.55

=== BMI Tracker Menu ===
1. Enter Details
2. Display BMI Information
0. Exit
Enter choice: 2

BMI Details:
Weight: 45.0 kg
Height: 1.55 meters
BMI: 18.73
Classification: Normal weight

=== BMI Tracker Menu ===
1. Enter Details
2. Display BMI Information
0. Exit
Enter choice: 0
Exiting...
```

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.

Solution:

Class File 1:

```
package com.discountcalculator;
public class DiscountCalculator {
   private double originalPrice;
   private double discountRate;
   private double finalPrice;

   public DiscountCalculator(double originalPrice, double discountRate) {
      this.originalPrice = originalPrice;
      this.discountRate = discountRate;
      calculateFinalPrice();
   }

   public double getOriginalPrice() {
      return originalPrice;
   }

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

   public double getDiscountRate() {
      return discountRate;
   }

   public void setDiscountRate(double discountRate) {
```

```
        this.discountRate = discountRate;
        calculateFinalPrice();
    }

    public double getFinalPrice() {
        return finalPrice;
    }

    private void calculateFinalPrice() {
        double discountAmount = originalPrice * (discountRate / 100);
        this.finalPrice = originalPrice - discountAmount;
    }

    public String toString() {
        double discountAmount = originalPrice * (discountRate / 100);
        return String.format("Original Price: ₹%.2f\nDiscount Rate:
%.2f%%\nDiscount Amount: ₹%.2f\nFinal Price: ₹%.2f",
                originalPrice, discountRate, discountAmount, finalPrice);
    }
}
```

Class File 2:

```
package com.discountcalculator;
import java.util.Scanner;

public class DiscountCalculatorUtil {
    private Scanner scanner = new Scanner(System.in);
    private DiscountCalculator discountCalculator;

    public void acceptRecord() {
        System.out.print("Enter the original price of the item: ₹");
        double originalPrice = scanner.nextDouble();
        System.out.print("Enter the discount rate (%): ");
        double discountRate = scanner.nextDouble();
        discountCalculator = new DiscountCalculator(originalPrice, discountRate);
    }
```

```
    public void printRecord() {
        if (discountCalculator != null) {
            System.out.println(discountCalculator.toString());
        } else {
            System.out.println("No record found. Please enter the details first.");
        }
    }

    public void menuList() {
        System.out.println("Menu:");
        System.out.println("1. Enter Item Details");
        System.out.println("2. Show Discount and Final Price");
        System.out.println("3. Exit");
    }
}
```

```
package com.discountcalculator;
import java.util.Scanner;

public class Program {
    public static void main(String[] args) {
        DiscountCalculatorUtil util = new DiscountCalculatorUtil();
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            util.menuList();
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    util.acceptRecord();
                    break;
                case 2:
```

```
                util.printRecord();
                break;
            case 3:
                System.out.println("Done..");
                break;
            default:
                System.out.println("Invalid choice! Please try again.");
          }
      }while (choice != 3);
      scanner.close();
   }
}
```

```
Menu:
1. Enter Item Details
2. Show Discount and Final Price
3. Exit
Enter your choice: 1
Enter the original price of the item: ₹2560
Enter the discount rate (%): 15
Menu:
1. Enter Item Details
2. Show Discount and Final Price
3. Exit
Enter your choice: 2
Original Price: ₹2560.00
Discount Rate: 15.00%
Discount Amount: ₹384.00
Final Price: ₹2176.00
Menu:
1. Enter Item Details
2. Show Discount and Final Price
3. Exit
Enter your choice: 3
Done..
```

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.

Solution:

Class File 1:

```
package org.tollrevenue;

public class TollBoothRevenueManager {
   private double carRate;
   private double truckRate;
   private double motorcycleRate;
   private int numCars;
   private int numTrucks;
   private int numMotorcycles;

   public TollBoothRevenueManager(double carRate, double truckRate, double
motorcycleRate) {
      this.carRate = carRate;
      this.truckRate = truckRate;
      this.motorcycleRate = motorcycleRate;
   }

   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;
    }

    public double calculateTotalRevenue() {
        return (numCars * carRate) + (numTrucks * truckRate) + (numMotorcycles
* motorcycleRate);
    }

    public int calculateTotalVehicles() {
        return numCars + numTrucks + numMotorcycles;
    }

    public String toString() {
        return String.format("Total Vehicles: %d\nTotal Revenue: ₹%.2f",
                        calculateTotalVehicles(), calculateTotalRevenue());
    }
}
```

Class File 2:

```
package org.tollrevenue;

import java.util.Scanner;

public class TollBoothRevenueManagerUtil {

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

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

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

        System.out.print("Enter the toll rate for Motorcycle (₹): ");
```

```
        double motorcycleRate = scanner.nextDouble();

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

        System.out.print("Enter the number of Cars: ");
        manager.setNumCars(scanner.nextInt());

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

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

        return manager;
    }

    public static void printRecord(TollBoothRevenueManager manager) {
        System.out.println(manager.toString());
    }

    public static void menuList() {
        System.out.println("Toll Booth Revenue Management Menu:");
        System.out.println("1. Enter new toll rates and vehicle counts");
        System.out.println("2. Exit");
    }
}
```

Class file 3:

```
package org.tollrevenue;

import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```
        int choice;

        do {
            TollBoothRevenueManagerUtil.menuList();
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    TollBoothRevenueManager manager =
TollBoothRevenueManagerUtil.acceptRecord();
                    TollBoothRevenueManagerUtil.printRecord(manager);
                    break;
                case 2:
                    System.out.println("Exiting the program.");
                    break;
                default:
                    System.out.println("Invalid choice. Please enter again.");
                    break;
            }
        } while (choice != 2);

        scanner.close();
    }
}
```

```
Toll Booth Revenue Management Menu:
1. Enter new toll rates and vehicle counts
2. Exit
Enter your choice: 1
Enter the toll rate for Car (₹): 100
Enter the toll rate for Truck (₹): 200
Enter the toll rate for Motorcycle (₹): 50
Enter the number of Cars: 260
Enter the number of Trucks: 71
Enter the number of Motorcycles: 410
Total Vehicles: 741
Total Revenue: ₹60700.00
Toll Booth Revenue Management Menu:
1. Enter new toll rates and vehicle counts
2. Exit
Enter your choice: 2
Exiting the program.
```