ASSIGNMENT NO. 04

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.

CODE:

```
package org.assignment4;

import java.util.Scanner;

class LoanAmortizationCalculator{
	double principal;
	double annualInterestRate;
	int loanTerm;
	double monthlyPayment;
	double totalAmountPaid;

	LoanAmortizationCalculator(double principle, double annualInterestRate, int loanTerm){
		this.principal = principle;
		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 void calculateMonthlyPayment() {
      double monthlyInterestRate = (annualInterestRate / 12) / 100;

		int numberOfMonths = loanTerm * 12;

		monthlyPayment = principal * (monthlyInterestRate * Math.pow(1 +
monthlyInterestRate, numberOfMonths)) / (Math.pow(1 + monthlyInterestRate,
numberOfMonths) - 1);

		totalAmountPaid = monthlyPayment * loanTerm * 12;
   }

	@Override
	public String toString() {
		return "LoanAmortizationCalculator [principal=" + principal + ",
annualInterestRate=" + annualInterestRate
				+ ", loanTerm=" + loanTerm + ", monthlyPayment=" +
monthlyPayment + ", totalAmountPaid="
					+ totalAmountPaid + "]";
	}

}

class LoanAmortizationCalculatorUtil{
	private String monthlyPayment;
	private String totalAmountPaid;

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

      System.out.println("Enter the principal amount (₹): ");
      double principal = sc.nextDouble();
```

```
        System.out.println("Enter the annual interest rate (in percentage): ");
        double annualInterestRate = sc.nextDouble();

        System.out.println("Enter the loan term (in years): ");
        int loanTermYears = sc.nextInt();

        sc.close();

        return new LoanAmortizationCalculator(principal, annualInterestRate,
loanTermYears);
    }


    public void menuList() {
        System.out.println("1. Calculate Loan Amortization");
        System.out.println("2. Exit");
    }


        public void printRecord(LoanAmortizationCalculator calculator) {
                System.out.println("Monthly Payment: ₹" + monthlyPayment);
                System.out.println("Total Payment: ₹ "+totalAmountPaid );
        }
}

public class Program1 {

        public static void main(String[] args) {

                 LoanAmortizationCalculatorUtil util = new
LoanAmortizationCalculatorUtil();
            Scanner scanner = new Scanner(System.in);

            while (true) {
                util.menuList();
                System.out.print("Choose an option: ");
                int choice = scanner.nextInt();

                switch (choice) {
                    case 1:
                        LoanAmortizationCalculator calculator = util.acceptRecord();
                        util.printRecord(calculator);
                        break;
                    case 2:
                        System.out.println("Exiting...");
                        scanner.close();
                        return;
                    default:
                        System.out.println("Invalid option! Please try again.");
                }
```

```
            }
        }
}
```

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 (₹).

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.

CODE:

```
package org.prob4;

public class CompoundInterestCalculator {
    private double principal;
    private double annualInterestRate;
    private int numberOfCompounds;
    private int years;
    private double futureValue;
    private double totalInterest;


    public CompoundInterestCalculator(double principal, double annualInterestRate, int numberOfCompounds, int years) {
        this.principal = principal;
        this.annualInterestRate = annualInterestRate / 100;
        this.numberOfCompounds = numberOfCompounds;
        this.years = years;
        this.futureValue = calculateFutureValue();
        this.totalInterest = futureValue - principal;
    }
```

```
    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 / 100;
    }

    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 getFutureValue() {
        return futureValue;
    }

    public double getTotalInterest() {
        return totalInterest;
    }


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


    @Override
    public String toString() {
```

```
        return String.format("Future Value: ₹%.2f, Total Interest Earned: ₹%.2f", futureValue,
totalInterest);
    }
}

package org.prob4;

import java.util.Scanner;

public class CompoundInterestCalculatorUtil {
    private static Scanner sc = new Scanner(System.in);


    public static CompoundInterestCalculator acceptRecord() {
        System.out.print("Enter the initial investment amount (₹): ");
        double principal = sc.nextDouble();
        System.out.print("Enter the annual interest rate (in %): ");
        double annualInterestRate = sc.nextDouble();
        System.out.print("Enter the number of times interest is compounded per year: ");
        int numberOfCompounds = sc.nextInt();
        System.out.print("Enter the investment duration (in years): ");
        int years = sc.nextInt();

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


    public static void printRecord(CompoundInterestCalculator calculator) {
        System.out.println(calculator);
    }


    public static void menuList() {
        System.out.println("1. Calculate Future Value");
        System.out.println("2. Exit");
    }
}

package org.prob4;

import java.util.Scanner;

public class Problem4 {
            public static void main(String[] args) {
                    int choice;
        Scanner sc = new Scanner(System.in);

        do {
            CompoundInterestCalculatorUtil.menuList();
```

```
            System.out.print("Enter choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    CompoundInterestCalculator calculator =
CompoundInterestCalculatorUtil.acceptRecord();
                    CompoundInterestCalculatorUtil.printRecord(calculator);
                    break;

                case 2:
                    System.out.println("Exiting...");
                    break;

                default:
                    System.out.println("Invalid choice, please try again.");
            }
        } while (choice != 2);
        sc.close();
            }
}
```

OUTPUT:

```
1. Calculate Future Value
2. Exit
Enter choice: 1
Enter the initial investment amount (₹): 25000
Enter the annual interest rate (in %): 2.3
Enter the number of times interest is compounded per year: 4
Enter the investment duration (in years): 7
Future Value: ₹29353.59, Total Interest Earned: ₹4353.59
1. Calculate Future Value
2. Exit
Enter choice:
```

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.

CODE:

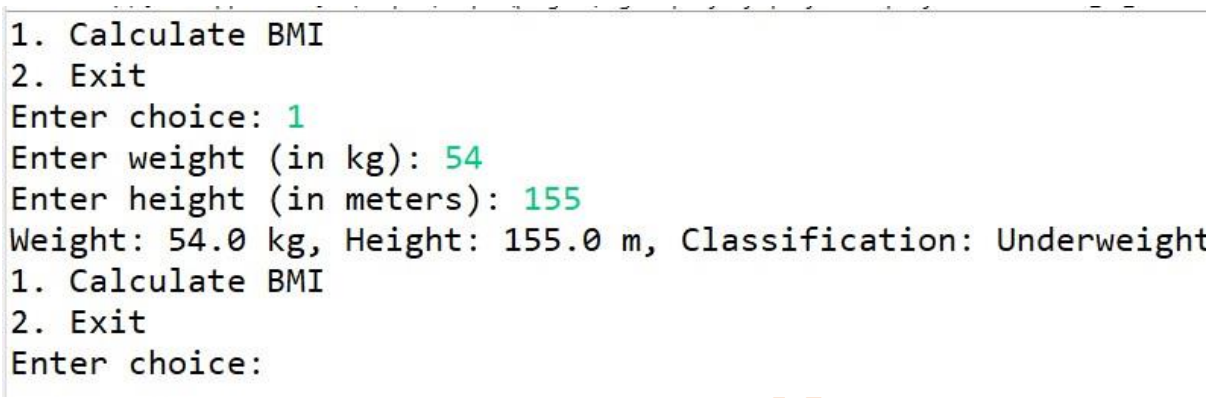
```
package org.prob3;

public class BMITracker {

	private float weight;
	private float height;
	private float bmi;

	public BMITracker(float weight, float height) {
    this.weight = weight;
    this.height = height;
    this.bmi = calculateBMI();
  }

	public float getWeight() {
		return weight;
	}
	public void setWeight(float weight) {
		this.weight = weight;
	}
	public float getHeight() {
		return height;
	}
	public void setHeight(float height) {
		this.height = height;
	}
	public float getBmi() {
    return bmi;
  }
	private float calculateBMI() {
    return weight / (height * height);
  }

  public String classifyBMI() {
    if (bmi < 18.5) {
      return "Underweight";
    } else if (bmi >= 18.5 && bmi < 24.9) {
      return "Normal weight";
```

```
        } else if (bmi >= 25 && bmi < 29.9) {
            return "Overweight";
        } else {
            return "Obese";
        }
    }
        @Override
        public String toString() {
                return "BMITracker [weight=" + weight + ", height=" + height + "]";
        }

}

package org.prob3;

public class BMITracker {

        private float weight;
        private float height;
        private float bmi;

        public BMITracker(float weight, float height) {
        this.weight = weight;
        this.height = height;
        this.bmi = calculateBMI();
    }

        public float getWeight() {
                return weight;
        }
        public void setWeight(float weight) {
                this.weight = weight;
        }
        public float getHeight() {
                return height;
        }
        public void setHeight(float height) {
                this.height = height;
        }
        public float getBmi() {
        return bmi;
    }
        private float calculateBMI() {
        return weight / (height * height);
    }
```

```
    public String classifyBMI() {
        if (bmi < 18.5) {
            return "Underweight";
        } else if (bmi >= 18.5 && bmi < 24.9) {
            return "Normal weight";
        } else if (bmi >= 25 && bmi < 29.9) {
            return "Overweight";
        } else {
            return "Obese";
        }
    }
            @Override
            public String toString() {
                    return "Weight: " + weight + " kg, Height: " + height + " m"+
               ", Classification: " + classifyBMI();
            }

}

package org.prob3;

import java.util.Scanner;

public class Problem3 {

                    public static void main(String[] args) {
                            int choice;
                Scanner sc = new Scanner(System.in);

                do {
                    BMITrackerUtil.menuList();
                    System.out.print("Enter choice: ");
                    choice = sc.nextInt();

                    switch (choice) {
                        case 1:
                            BMITracker tracker = BMITrackerUtil.acceptRecord();
                            BMITrackerUtil.printRecord(tracker);
                            break;

                        case 2:
                            System.out.println("Exiting...");
                            break;

                        default:
                            System.out.println("Invalid choice, please try again.");
```

```
            }
        } while (choice != 2);
        sc.close();
    }
}
```

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 (₹).

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.

CODE:

```
package org.assign4;

public class DiscountCalculator {
    private double originalPrice;
    private double discountRate;
    private double discountAmount;
    private double finalPrice;
```

```
    public DiscountCalculator(double originalPrice, double discountRate) {
        this.originalPrice = originalPrice;
        this.discountRate = discountRate;
        this.discountAmount = calculateDiscountAmount();
        this.finalPrice = calculateFinalPrice();
    }


    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 getDiscountAmount() {
        return discountAmount;
    }

    public double getFinalPrice() {
        return finalPrice;
    }


    private double calculateDiscountAmount() {
        return originalPrice * (discountRate / 100);
    }


    private double calculateFinalPrice() {
        return originalPrice - discountAmount;
    }


    @Override
    public String toString() {
        return String.format("Discount Amount: ₹%.2f, Final Price: ₹%.2f", discountAmount,
finalPrice);
    }
}
```

```
package org.assign4;

import java.util.Scanner;

public class DiscountCalculatorUtil {
   private static Scanner sc = new Scanner(System.in);

   // Method to accept the original price and discount rate from the user
   public static DiscountCalculator acceptRecord() {
      System.out.print("Enter the original price (₹): ");
      double originalPrice = sc.nextDouble();
      System.out.print("Enter the discount rate (in %): ");
      double discountRate = sc.nextDouble();

      return new DiscountCalculator(originalPrice, discountRate);  // Create
DiscountCalculator object
   }

   // Method to display the discount amount and final price
   public static void printRecord(DiscountCalculator calculator) {
      System.out.println(calculator);  // Calls toString method of DiscountCalculator
   }

   // Method to display menu
   public static void menuList() {
      System.out.println("1. Calculate Discount and Final Price");
      System.out.println("2. Exit");
   }
}

package org.assign4;

import java.util.Scanner;

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

      do {
         DiscountCalculatorUtil.menuList();  // Display menu
         System.out.print("Enter choice: ");
         choice = sc.nextInt();

         switch (choice) {
            case 1:

               DiscountCalculator calculator = DiscountCalculatorUtil.acceptRecord();
               DiscountCalculatorUtil.printRecord(calculator);
```

```
                break;

            case 2:
                System.out.println("Exiting...");
                break;

            default:
                System.out.println("Invalid choice, please try again.");
        }
    } while (choice != 2);
    sc.close();
  }
}
```

OUTPUT:

```
1. Calculate Discount and Final Price
2. Exit
Enter choice: 1
Enter the original price (₹): 3000
Enter the discount rate (in %): 15
Discount Amount: ₹450.00, Final Price: ₹2550.00
1. Calculate Discount and Final Price
2. Exit
Enter choice:
```

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.

CODE:

```
package org.prob5;

public class TollBoothRevenueManager {
   private double carRate;
   private double truckRate;
   private double motorcycleRate;

   private int numCars;
   private int numTrucks;
   private int numMotorcycles;
   private double totalRevenue;


   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;
      this.totalRevenue = 0;
   }


   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 getTotalRevenue() {
        return totalRevenue;
    }


    public void calculateTotalRevenue() {
        totalRevenue = (numCars * carRate) + (numTrucks * truckRate) + (numMotorcycles *
motorcycleRate);
    }


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


    @Override
    public String toString() {
```

```
        return String.format("Total Vehicles: %d, Total Revenue: ₹%.2f",
calculateTotalVehicles(), totalRevenue);
    }
}
```

```
package org.prob5;

import java.util.Scanner;

public class TollBoothRevenueManagerUtil {
    private static Scanner sc = new Scanner(System.in);

    public static TollBoothRevenueManager acceptRecord() {
        System.out.print("Enter the toll rate for cars ₹: ");
        double carRate = sc.nextDouble();
        System.out.print("Enter the toll rate for trucks ₹: ");
        double truckRate = sc.nextDouble();
        System.out.print("Enter the toll rate for motorcycles ₹: ");
        double motorcycleRate = sc.nextDouble();


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

        System.out.print("Enter the number of cars: ");
        manager.setNumCars(sc.nextInt());
        System.out.print("Enter the number of trucks: ");
        manager.setNumTrucks(sc.nextInt());
        System.out.print("Enter the number of motorcycles: ");
        manager.setNumMotorcycles(sc.nextInt());

        manager.calculateTotalRevenue();
        return manager;
    }


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


    public static void menuList() {
        System.out.println("1. Set Toll Rates and Calculate Revenue");
        System.out.println("2. Exit");
    }
}
```

```
package org.prob5;

import java.util.Scanner;

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

      do {
         TollBoothRevenueManagerUtil.menuList();  // Display menu
         System.out.print("Enter choice: ");
         choice = sc.nextInt();

         switch (choice) {
            case 1:
               TollBoothRevenueManager manager =
TollBoothRevenueManagerUtil.acceptRecord();
               TollBoothRevenueManagerUtil.printRecord(manager);
               break;

            case 2:
               System.out.println("Exiting...");
               break;

            default:
               System.out.println("Invalid choice, please try again.");
         }
      } while (choice != 2);
      sc.close();
   }
}
```

OUTPUT:

```
Program (3) [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotsp
1. Set Toll Rates and Calculate Revenue
2. Exit
Enter choice: 1
Enter the toll rate for cars ₹: 60
Enter the toll rate for trucks ₹: 90
Enter the toll rate for motorcycles ₹: 40
Enter the number of cars: 5
Enter the number of trucks: 3
Enter the number of motorcycles: 7
Total Vehicles: 15, Total Revenue: ₹850.00
1. Set Toll Rates and Calculate Revenue
2. Exit
Enter choice:
```