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:

1) LoanAmortizationCalculator

```
package Program.org;

public class LoanAmortizationCalculator {
	private double loanAmount;
	private double annualInterestRate;
	private int loanTerm;

	public double getLoanAmount() {
		return loanAmount;
	}
	public void setLoanAmount(double loanAmount) {
		this.loanAmount = loanAmount;
	}
	public double getAnnualInterestRate() {
		return annualInterest Rate;
	}
	public void setAnnualInterestRate(double annualInterestRate) {
```

```
		this.annualInterestRate = annualInterestRate;
	}
	public int getLoanTerm() {
		return loanTerm;
	}
	public void setLoanTerm(int loanTerm) {
		this.loanTerm = loanTerm;
	}

	@Override
	public String toString() {


		String res = "LoanAmortizationCalculator [loanAmount=" + loanAmount + ",
annualInterestRate=" + annualInterestRate
				+ ", loanTerm=" + loanTerm + "]";
		res = res + "monthly payment" + calculateMonthlyPayment(this.loanAmount,
this.annualInterestRate, this.loanTerm);
		return res;
	}

	public double calculateMonthlyPayment(double loanAmount, double
annualInterestRate, int loanTerm) {
		int numberOfMonths = loanTerm * 12;
		double monthlyInterestRate = annualInterestRate / 12 / 100;
		double monthlyPayment = (loanAmount * Math.pow(1 + monthlyInterestRate
, numberOfMonths)) / ((Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1));
		return monthlyPayment;
	}

}
```

2. LoanAmortizationCalculatorUtil.

```
package Program.org;

import java.util.Scanner;

public class LoanAmortizationCalculatorUtil {

	private Scanner sc;

	public LoanAmortizationCalculatorUtil() {
		sc = new Scanner(System.in);
	}

	public void menuList() {
		System.out.println("Enter choice 1. Accept and print record , 2. To exit");
		int choice = sc.nextInt();
		switch (choice) {
		case 1:
```

```
                acceptRecord();
                break;
            case 2:
                return;
            default:
                System.out.println("wrong choice");
                break;
            }
        }

        private void acceptRecord() {
            LoanAmortizationCalculator loan = new LoanAmortizationCalculator();
            System.out.println("enter loan amount, interest rate, and term");
            double amount = sc.nextDouble();
            loan.setLoanAmount(amount);
            double rate = sc.nextDouble();
            loan.setAnnualInterestRate(rate);
            int term = sc.nextInt();
            loan.setLoanTerm(term);
            System.out.println(loan.toString());

        }
}
```

3.Program.java

```
package Program.org;

public class Program {

        public static void main(String[] args) {
            LoanAmortizationCalculatorUtil loan = new
LoanAmortizationCalculatorUtil();
            loan.menuList();
        }

}
```

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> Program [Java Application] D:\ESCLIPSE\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (11-Sept-2024, 11:27:28 am – 11:28:28 am) [pid: 3512]
Enter choice 1. Accept and print record , 2. To exit

1
enter loan amount, interest rate, and term

9000
7.5
2
LoanAmortizationCalculator [loanAmount=9000.0, annualInterestRate=7.5, loanTerm=2]monthly payment64799.41341833845
```

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:

1. Compound Interest Calculator

```
package exe.java;

class CompoundInterestCalculator {

    private double principal;

    private double annualInterestRate;

    private int numberOfCompounds;

    private int years;

    // Default Constructor

    public CompoundInterestCalculator() {}

    // Parameterized 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) {
    this.numberOfCompounds = numberOfCompounds;
}
public int getYears() {
    return years;


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

```
    }

    // Method to calculate future value

    public double calculateFutureValue() {

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



    // Method to calculate total interest earned

    public double calculateTotalInterest() {

        return calculateFutureValue() - principal;

    }

    @Override

    public String toString() {

        return String.format("Investment Details:\nPrincipal: ₹%.2f\nAnnual Interest Rate:
%.2f%%\n" +

                        "Compounds per Year: %d\nInvestment Duration: %d years",

                        principal, annualInterestRate, numberOfCompounds, years);

    }

}
```

2. CompoundInterestUtil.java

```
Package exe.java;

import java.util.Scanner;

class CompoundInterestCalculatorUtil {
            Scanner scanner = new Scanner(System.in);
    private CompoundInterestCalculator compoundInterestCalculator;

    public void acceptRecord() {
        System.out.print("Enter Initial Investment Amount (in ₹): ");
        double principal = scanner.nextDouble();
        System.out.print("Enter Annual Interest Rate (in %): ");
        double annualInterestRate = scanner.nextDouble();
        System.out.print("Enter Number of Compounds per Year: ");
```

```
        int numberOfCompounds = scanner.nextInt();
        System.out.print("Enter Investment Duration (in years): ");
        int years = scanner.nextInt();

        compoundInterestCalculator = new CompoundInterestCalculator(principal,
annualInterestRate, numberOfCompounds, years);
    }

    public void printRecord() {

            System.out.println(compoundInterestCalculator);  // Display investment details
            double futureValue = compoundInterestCalculator.calculateFutureValue();
            double totalInterest = compoundInterestCalculator.calculateTotalInterest();
            System.out.printf("Future Value: ₹%.2f\n", futureValue);
            System.out.printf("Total Interest Earned: ₹%.2f\n", totalInterest);
    }

    // Method to display the menu options
    public void menuList() {
        System.out.println("1. Enter Investment Details");
        System.out.println("2. Display Future Value and Total Interest");
        System.out.println("3. Exit");
    }
}
```

3. Program.java (Main Method)

```
package exe.java;

import java.util.Scanner;
public class Program {
    public static void main(String[] args) {
        CompoundInterestCalculatorUtil util = new CompoundInterestCalculatorUtil();
        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();   // Accept investment details from user
                    break;
                case 2:
                    util.printRecord();    // Display future value and total interest
                    break;
                case 3:
                    System.out.println("Exiting...");
```

```
                break;
            default:
                System.out.println("Invalid choice! Please select a valid option.");
        }

    } while (choice != 3);  // Repeat menu until user selects "Exit"
    scanner.close();
  }
}
```

Output:

```
1. Enter Investment Details
2. Display Future Value and Total Interest
3. Exit
Enter your choice: 1
Enter Initial Investment Amount (in ₹): 5600
Enter Annual Interest Rate (in %): 2.5
Enter Number of Compounds per Year: 1
Enter Investment Duration (in years): 1
1. Enter Investment Details
2. Display Future Value and Total Interest
3. Exit
Enter your choice: 2
Investment Details:
Principal: ₹5600.00
Annual Interest Rate: 2.50%
Compounds per Year: 1
Investment Duration: 1 years
Future Value: ₹19600.00
Total Interest Earned: ₹14000.00
1. Enter Investment Details
2. Display Future Value and Total Interest
3. Exit
Enter your choice: 3
Exiting...
```

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.
Solution:

1. BMI (Body Mass Index) Tracker

```
package exe.java;

import java.util.Scanner;
public class Program {

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

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

            switch (choice) {
                case 1:
                    tracker = util.acceptRecord();  // Accept new record
                    util.printRecord(tracker);      // Display the calculated BMI
                    break;
                case 2:

                        util.printRecord(tracker);  // Display the last BMI record
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice, please try again.");
            }
        } while (choice != 3);
        sc.close();
    }
}
```

2. BMITracker.util

```
package exe.java;

import java.util.Scanner;
```

```
public class BMITrackerrUtil {
    private Scanner sc = new Scanner(System.in);

    public BMITrackerr acceptRecord() {
        System.out.print("Enter weight : ");
        double weight = sc.nextDouble();
        System.out.print("Enter height : ");
        double height = sc.nextDouble();
        return new BMITrackerr(weight, height);
    }

    public void printRecord(BMITrackerr tracker) {
        System.out.println(tracker);
    }

    // Method to display the menu
    public void menuList() {
        System.out.println("1. Calculate BMI");
        System.out.println("2. Display Last BMI Record");
        System.out.println("3. Exit");
    }
}
```

3. Program.java

```
package exe.java;

import java.util.Scanner;
public class Program {

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

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

            switch (choice) {
                case 1:
                    tracker = util.acceptRecord();  // Accept new record
```

```
                util.printRecord(tracker);      // Display the calculated BMI
                break;
            case 2:

                    util.printRecord(tracker);  // Display the last BMI record
                break;
            case 3:
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice, please try again.");
        }
    } while (choice != 3);
    sc.close();
  }
}
```

Output:

```
1. Calculate BMI
2. Display Last BMI Record
3. Exit
Enter your choice: 1
Enter weight : 65
Enter height : 5.6
BMI: 2.07
Underweight
1. Calculate BMI
2. Display Last BMI Record
3. Exit
Enter your choice: 2
BMI: 2.07
Underweight
1. Calculate BMI
2. Display Last BMI Record
3. Exit
Enter your choice: 3
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:

1. DiscountCalculator

```
package program.java;

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

    // Constructor
    public DiscountCalculator(double originalPrice, double discountRate) {
        this.originalPrice = originalPrice;
        this.discountRate = discountRate;
        calculateDiscount();
    }


    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;
    }

    //logic to calculate discount and final price
```

```
    private void calculateDiscount() {
        discountAmount = originalPrice * (discountRate / 100);
        finalPrice = originalPrice - discountAmount;
    }

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

2. DiscountCalculatorUtil.

```
package program.java;

import java.util.Scanner;

public class DiscountCalculatorUtil {

    private static DiscountCalculator lastRecord;

    public static void acceptRecord(Scanner scanner) {
        System.out.print("Enter original price: ₹");
        double originalPrice = scanner.nextDouble();
        System.out.print("Enter discount percentage: ");
        double discountRate = scanner.nextDouble();

        lastRecord = new DiscountCalculator(originalPrice, discountRate);  //
Create new record

        System.out.println("Discount calculation completed!");
    }

    public static void printRecord() {
            System.out.println(lastRecord);  // Use toString() of
DiscountCalculator
    }
    // Method to display menu options
    public static void menuList() {
        System.out.println("Discount Calculator Menu:");
        System.out.println("1. Calculate Discount");
        System.out.println("2. Display Last Discount");
        System.out.println("3. Exit");
    }
```

```
}
```

3. Program.java (main program)

```
package program.java;

import java.util.Scanner;

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

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

            switch (choice) {
                case 1:
                    DiscountCalculatorUtil.acceptRecord(scanner);
                    break;
                case 2:
                    DiscountCalculatorUtil.printRecord();
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice! Please try again.");
                    break;
            }
        } while (choice != 3);

        scanner.close();
    }
}
```


Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> Program (2) [Java Application] D:\ESCLIPSE\ecl
Discount Calculator Menu:
1. Calculate Discount
2. Display Last Discount
3. Exit
Enter your choice: 1
Enter original price: ₹70000
Enter discount percentage: 7
Discount calculation completed!
Discount Calculator Menu:
1. Calculate Discount
2. Display Last Discount
3. Exit
Enter your choice: 2
Original Price: ₹70000.00
Discount Rate: 7.00%
Discount Amount: ₹4900.00
Final Price: ₹65100.00
Discount Calculator Menu:
1. Calculate Discount
2. Display Last Discount
3. Exit
Enter your choice: 3
Exiting...
```

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:

1. TollBooth

```
package exe.java;

public class TollBooth{
    private double carRate;
    private double truckRate;
    private double motorcycleRate;
    private int carCount;
```

```
private int truckCount;
private int motorcycleCount;

// Constructor
public TollBooth(double carRate, double truckRate, double motorcycleRate) {
    this.carRate = carRate;
    this.truckRate = truckRate;
    this.motorcycleRate = motorcycleRate;
    this.carCount = 0;
    this.truckCount = 0;
    this.motorcycleCount = 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 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;
    }

    // Method to calculate total revenue
    public double calculateTotalRevenue() {
        return (carCount * carRate) + (truckCount * truckRate) + (motorcycleCount *
motorcycleRate);
    }

    // Updated toString method to display details on new lines
    @Override
    public String toString() {
        return "TollBoothRevenueManager Details:\n" +
                "Car Rate: ₹" + carRate + "\n" +
                "Truck Rate: ₹" + truckRate + "\n" +
                "Motorcycle Rate: ₹" + motorcycleRate + "\n" +
                "Number of Cars: " + carCount + "\n" +
                "Number of Trucks: " + truckCount + "\n" +
                "Number of Motorcycles: " + motorcycleCount + "\n" +
                "Total Revenue: ₹" + calculateTotalRevenue();
    }
}
```

2. TollBooyhUtil.java

```
package exe.java;

import java.util.Scanner;

public class TollBoothUtil {

    private static Scanner scanner = new Scanner(System.in); // Single Scanner
instance

    public static TollBooth acceptRecord() {
        System.out.print("Enter toll rate for Car : ");
        double carRate = scanner.nextDouble();

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

```
        System.out.print("Enter toll rate for Motorcycle : ");
        double motorcycleRate = scanner.nextDouble();

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

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

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

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

        return manager;
    }

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

    public static void menuList() {
        System.out.println("Toll Booth Revenue Management System");
        System.out.println("1. Accept Toll Rates and Vehicle Counts");
        System.out.println("2. Display Toll Booth Details");
        System.out.println("3. Exit");
    }
}
```

3. Program.java

```
package exe.java;

import java.util.Scanner;

public class Program {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TollBooth manager = null;
        while (true) {
            TollBoothUtil.menuList();
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            switch (choice) {

                case 1:
                    manager = TollBoothUtil.acceptRecord();
                    break;
```

```
            case 2:
                if (manager != null) {
                    TollBoothUtil.printRecord(manager);
                } else {
                    System.out.println("Please enter toll rates and vehicle counts first.");
                }
                break;
            case 3:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }

  }
}
```

Output:

```
<terminated> Program (5) [Java Application] D:\ESCLIPSE\eclipse\plugins\org.
Toll Booth Revenue Management System
1. Accept Toll Rates and Vehicle Counts
2. Display Toll Booth Details
3. Exit
Enter your choice: 1
Enter toll rate for Car : 700
Enter toll rate for Truck : 1400
Enter toll rate for Motorcycle : 400
Enter number of Cars: 7
Enter number of Trucks: 9
Enter number of Motorcycles: 4
Toll Booth Revenue Management System
1. Accept Toll Rates and Vehicle Counts
2. Display Toll Booth Details
3. Exit
Enter your choice: 2
TollBoothRevenueManager Details:
Car Rate: ₹700.0
Truck Rate: ₹1400.0
Motorcycle Rate: ₹400.0
Number of Cars: 7
Number of Trucks: 9
Number of Motorcycles: 4
Total Revenue: ₹19100.0
Toll Booth Revenue Management System
1. Accept Toll Rates and Vehicle Counts
2. Display Toll Booth Details
3. Exit
```