

Note:

- The assignment is designed to practice class, fields, and methods only.
- Create a separate project for each question.
- Do not use getter/setter methods or constructors for these assignments.
- Define two classes: one class to implement the logic and another class to test it.

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:**
 - $$\text{monthlyPayment} = \text{principal} * (\text{monthlyInterestRate} * (1 + \text{monthlyInterestRate})^{\text{numberOfMonths}}) / ((1 + \text{monthlyInterestRate})^{\text{numberOfMonths}} - 1)$$
 - Where $\text{monthlyInterestRate} = \text{annualInterestRate} / 12 / 100$ and $\text{numberOfMonths} = \text{loanTerm} * 12$
 - 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 class LoanAmortizationCalculator with methods acceptRecord, calculateMonthlyPayment & printRecord and test the functionality in main method.

```
package ooj;
import java.util.*;
class Loanal{
    double principal;
    double annualIntrestRate;
    int loanTerm;

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

        System.out.println("Enter the loan amount (Principal): Rupees ");
        principal = sc.nextDouble();

        System.out.println("Enter the annual intrest rate (in %): ");
        annualIntrestRate = sc.nextDouble();

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

        sc.close();
    }
    double calculateMonthlypay() {
```

ASSIGNMENT NO.3

```

double monthlyIntrestRate = annualIntrestRate / 12 / 100;
int numberOfMonths = loanTerm * 12;
return (double) (principal * (monthlyIntrestRate * Math.pow(1 + monthlyIntrestRate,
numberOfMonths)) / (Math.pow(1 + monthlyIntrestRate, numberOfMonths) - 1));

    }
    void printRecord() {
        double monthlyPayment = calculateMonthllypay();
        double totalpayment = monthlyPayment * loanTerm * 12;

        System.out.printf("Monthly Payment: Rupees %.2f\n", monthlyPayment);
        System.out.printf("Total Payment over the life of loan: Rupees %.2f\n", totalpayment);
    }
}

```

```

public class Intrest {
    public static void main(String[] args) {
        Loanca1 cal= new Loanca1();
        cal.acceptRecord();
        cal.printRecord();
    }
}

```

```

1 package ooj;
2 import java.util.*;
3 class Loanca1{
4     double principal;
5     double annualIntrestRate;
6     int loanTerm;
7
8     void acceptRecord() {
9         Scanner sc = new Scanner(System.in);
10
11         System.out.println("Enter the loan amount (Principal): Rupees ");
12         principal = sc.nextDouble();
13
14         System.out.println("Enter the annual intrest rate (in %): ");
15         annualIntrestRate = sc.nextDouble();
16
17         System.out.println("Enter the loan term (in years): ");
18         loanTerm = sc.nextInt();
19
20         sc.close();
21     }
22     double calculateMonthllypay() {
23         double monthlyIntrestRate = annualIntrestRate / 12 / 100;
24         int numberOfMonths = loanTerm * 12;
25         return (double) (principal * (monthlyIntrestRate * Math.pow(1 + monthlyIntrestRate, numberOfMonths)) /
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

<terminated> Intrest [Java Application] C:\Users\Shreya\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.22.0.2\j220240802-1626\jre\bin\javaw.exe (7 Sept 2024, 6:07:32 pm - 6:07:50 pm) [pid: 29804]
Enter the loan amount (Principal): Rupees
100000
Enter the annual intrest rate (in %):
6
Enter the loan term (in years):
10
Monthly Payment: Rupees 1110.21
Total Payment over the life of loan: Rupees 133224.60

```

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:**

$$\text{futureValue} = \text{principal} * (1 + \text{annualInterestRate} / \text{numberOfCompounds})^{(\text{numberOfCompounds} * \text{years})}$$
 - **Total Interest Earned:** $\text{totalInterest} = \text{futureValue} - \text{principal}$
3. Display the future value and the total interest earned, in Indian Rupees (₹).

Define class `CompoundInterestCalculator` with methods `acceptRecord` , `calculateFutureValue`, `printRecord` and test the functionality in `main` method.

package ooj;

import java.util.*;

```
public class Calculator {
    private double principal;
    private double annualIntrestRate;
    private int numberOfCompounds;
    private int years;
    private double futureValue;
    private double totalInterest;

    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Initial Interest Amount (in Rs): ");
        principal = sc.nextDouble();
        System.out.println("Annual Interest Rate(in %): ");
        annualIntrestRate = sc.nextDouble() / 100;
        System.out.println("Number of times the intrest is compounded per year: ");
        numberOfCompounds = sc.nextInt();
        System.out.println("Investment duration(in yrs): ");
        years = sc.nextInt();
        sc.close();
    }

    public void calculateFutureValue() {
        futureValue = principal * Math.pow((1 + annualIntrestRate /
        numberOfCompounds), numberOfCompounds * years);
        totalInterest = futureValue - principal;
    }
}
```

```

public void printRecord() {
    System.out.printf("\nFuture Value of Investment: ₹ %.2f\n", futureValue);
    System.out.printf("Total Interest Earned: ₹ %.2f\n", totalInterest);
}

public static class CompoundInterestCalculator {
    public static void main(String[] args) {
        Calculator ci = new Calculator();
        ci.acceptRecord();
        ci.calculateFutureValue();
        ci.printRecord();
    }
}
}

```

3. 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:** $BMI = \text{weight} / (\text{height} * \text{height})$
- 3. Classify the BMI into one of the following categories:**
 - **Underweight:** $BMI < 18.5$
 - **Normal weight:** $18.5 \leq BMI < 24.9$
 - **Overweight:** $25 \leq BMI < 29.9$
 - **Obese:** $BMI \geq 30$
- 4. Display the BMI value and its classification.**

Define class BMITracker with methods acceptRecord, calculateBMI, classifyBMI & printRecord and test the functionality in main method.

```
package ooj;
```

```
import java.util.Scanner;
```

```

class Bmicalc {
    private float weight;
    private float height;
    private float BMI;
    private String classification;

    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Weight(in kg): ");
        weight = sc.nextFloat();
        System.out.println("Enter Height(in meters): ");
        height = sc.nextFloat();
    }
}

```

```

        sc.close();
    }

    public void calculateBMI() {
        BMI = weight / (height * height);
    }

    public void classifyBMI() {
        if (BMI < 18.5f) {
            classification = "Underweight";
        } else if (BMI >= 18.5 && BMI < 24.9) {
            classification = "Normal Weight";
        } else if (BMI >= 25 && BMI < 29.9) {
            classification = "Overweight";
        } else if (BMI >= 30) {
            classification = "Obese";
        }
    }

    public void printRecord() {
        System.out.printf("BMI: %.2f%n", BMI);
        System.out.println("Classification : " + classification);
    }
}

public class BMITracker {
    public static void main(String[] args) {
        Bmicalc calculator = new Bmicalc();
        calculator.acceptRecord();
        calculator.calculateBMI();
        calculator.classifyBMI();
        calculator.printRecord();
    }
}

```

The screenshot shows an IDE with a project named 'BMI Tracker'. The 'Bmicalc' class is open, showing the following code:

```

5  class Bmicalc {
6      private float weight;
7      private float height;
8      private float BMI;
9      private String classification;
10
11     public void acceptRecord() {
12         Scanner sc = new Scanner(System.in);
13         System.out.println("Enter Weight(in kg): ");
14         weight = sc.nextFloat();
15         System.out.println("Enter Height(in meters): ");
16         height = sc.nextFloat();
17     }
18     sc.close();
19
20
21
22     public void calculateBMI() {
23         BMI = weight / (height * height);
24     }
25
26

```

The output console shows the following execution:

```

Enter Weight(in kg):
70
Enter Height(in meters):
1.9
BMI: 19.39
Classification : Normal Weight

```

The IDE also shows a list of files in the project, including 'Bmicalc.java', 'BMITracker.java', 'ArithmeticSwitch.java', 'ByteTest.java', 'Calculator.java', 'Conversion.java', 'DefaultValuesTest.java', 'DoubleTest.java', 'Employee.java', 'Employee1.java', 'FloatTest.java', 'Hello.java', 'Interest.java', 'IntTest.java', 'LongTest.java', 'package-info.java', 'ShortTest.java', 'Shreyas.java', and 'Test.java'.

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:** $\text{discountAmount} = \text{originalPrice} * (\text{discountRate} / 100)$
 - **Final Price Calculation:** $\text{finalPrice} = \text{originalPrice} - \text{discountAmount}$
3. Display the discount amount and the final price of the item, in Indian Rupees (₹).

Define class DiscountCalculator with methods acceptRecord, calculateDiscount & printRecord and test the functionality in main method.

```
package org.oopj;
```

```
import java.util.Scanner;
```

```
class Discount {
    private float discountAmount;
    private float originalPrice;
    private float discountRate;
    private float finalPrice;

    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Original Price (in ₹) : ");
        originalPrice = sc.nextFloat();
        System.out.println("Discount Percent : ");
        discountRate = sc.nextInt();
        sc.close();
    }

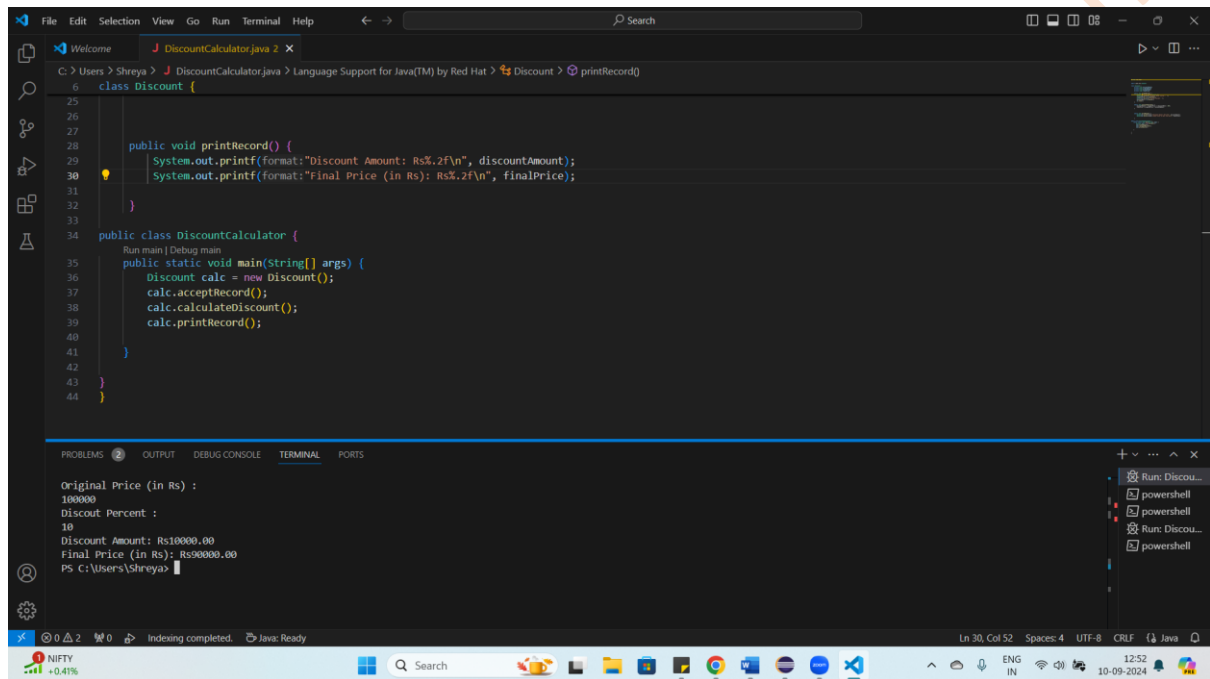
    public void calculateDiscount() {
        discountAmount = originalPrice * (discountRate / 100);
        finalPrice = originalPrice - discountAmount;
    }

    public void printRecord() {
        System.out.printf("Discount Amount: ₹%.2f\n", discountAmount);
        System.out.printf("Final Price (in ₹): ₹%.2f\n", finalPrice);
    }
}

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

ASSIGNMENT NO.3

```
        Discount calc = new Discount();
        calc.acceptRecord();
        calc.calculateDiscount();
        calc.printRecord();
    }
}
}
```



The screenshot shows an IDE with a Java file named `DiscountCalculator.java`. The code defines a `Discount` class with methods `printRecord()` and `acceptRecord()`, and a `DiscountCalculator` class with a `main` method. The `main` method creates a `Discount` object, calls `acceptRecord()`, `calculateDiscount()`, and `printRecord()`. The terminal output shows the execution results: Original Price (in Rs) : 100000, Discount Percent : 10, Discount Amount: Rs10000.00, Final Price (in Rs): Rs90000.00. The terminal prompt is `PS C:\Users\Shreya>`.

```
class Discount {
    public void printRecord() {
        System.out.printf(format:"Discount Amount: Rs%.2f\n", discountAmount);
        System.out.printf(format:"Final Price (in Rs): Rs%.2f\n", finalPrice);
    }
}

public class DiscountCalculator {
    public static void main(String[] args) {
        Discount calc = new Discount();
        calc.acceptRecord();
        calc.calculateDiscount();
        calc.printRecord();
    }
}
```

Original Price (in Rs) :
100000
Discount Percent :
10
Discount Amount: Rs10000.00
Final Price (in Rs): Rs90000.00
PS C:\Users\Shreya>

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 class TollBoothRevenueManager with methods acceptRecord, setTollRates, calculateRevenue & printRecord and test the functionality in main method.

```
package org.oopj;
```

```
import java.util.Scanner;
```

```
class Revenue {
    private double carTollRate;
    private double TruckTollRate;
    private double MotorcycleTollRate;
    private int numberOfCars;
    private int numberOfTrucks;
    private int numberOfMotorcycles;

    private double totalRevenue;
    private int totalVehicles;

    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Number of Cars: ");
        int cars = sc.nextInt();

        System.out.println("Number of Trucks: ");
        int trucks = sc.nextInt();

        System.out.println("Number of Motorcycles: ");
        int motorcycle = sc.nextInt();

        sc.close();
    }

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

        System.out.print("Set toll rate for Cars (₹): ");
        carTollRate = sc.nextDouble();

        System.out.print("Set toll rate for Trucks (₹): ");
        TruckTollRate = sc.nextDouble();

        System.out.print("Set toll rate for Motorcycles (₹): ");
        MotorcycleTollRate = sc.nextDouble();
        sc.close();
    }
}
```


ASSIGNMENT NO.3

```

        public void calculateRevenue() {
            totalRevenue = (numberOfCars * carTollRate) +(numberOfTrucks *
TruckTollRate) +(numberOfMotorcycles * MotorcycleTollRate);
            totalVehicles = numberOfCars + numberOfTrucks + numberOfMotorcycles;
        }

        public void printRecord() {
            System.out.println("\nToll Booth Revenue Summary:");
            System.out.println("Total Vehicles: " + totalVehicles);
            System.out.printf("Total Revenue (₹): ₹%.2f\n", totalRevenue);
        }
    }

}

public class TollBoothRevenueManager {

    public static void main(String[] args) {
        Revenue rev = new Revenue();
        rev.acceptRecord();
        rev.setTollRates();
        rev.calculateRevenue();
        rev.printRecord();
    }
}

```

The screenshot shows an IDE with the following components:

- Package Explorer:** Shows a project named 'oj' with a source folder 'src' containing various Java files like 'ArithmeticSwitch.java', 'BMITracker.java', 'ByteTest.java', 'Calculator.java', 'Conversion.java', 'DefaultValuesTest.java', 'DoubleTest.java', 'Employee.java', 'Employee1.java', 'FloatTest.java', 'hello.java', 'Intrest.java', 'IntTest.java', 'LongTest.java', 'package-info.java', 'ShortTest.java', 'Shrey.java', 'Test.java', and 'TollBoothRevenueManager.java'.
- Editor:** Displays the code for 'TollBoothRevenueManager.java', which includes the 'main' method and the 'TollBoothRevenueManager' class. The code is identical to the one provided in the previous block.
- Console:** Shows the output of the program execution:


```

Number of Cars:
10
Number of Trucks:
50
Number of Motorcycles:
5
Set toll rate for Cars (₹): 250
Set toll rate for Trucks (₹): 700
Set toll rate for Motorcycles (₹): 200
            
```

sandeepkulange@gmail.com