**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:**
     + 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 class LoanAmortizationCalculator with methods acceptRecord, calculateMonthlyPayment & printRecord and test the functionality in main method.

Code-

package org.example;

import java.util.Scanner;

import java.math.\*;

class LoanAmortizationCalculator {

double Principle;

double AnnIntrestRate;

double LoanYear;

double monthlyPayment;

double totalPayment;

void acceptRecord() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Principle Amount :");

this.Principle = sc.nextDouble();

System.out.print("Enter Annual Interest Rate :");

this.AnnIntrestRate = sc.nextDouble();

System.out.print("Enter Loan Year :");

this.LoanYear = sc.nextDouble();

}

void calculateMonthlyPayment() {

double monthlyInterestRate = AnnIntrestRate / (12 \* 100);

double numberOfMonths = LoanYear \* 12;

monthlyPayment = (Principle \* ((monthlyInterestRate \* (Math.pow(1 + monthlyInterestRate, numberOfMonths)))

/ ((Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1))));

totalPayment = (monthlyPayment \* numberOfMonths);

}

void printRecord() {

System.out.printf("Monthly Payment : %.2f", this.monthlyPayment);

System.out.printf("\nTotal Payment : %.2f", this.totalPayment);

;

}

}

public class LoanAmortization {

public static void main(String[] args) {

// TODO Auto-generated method stub

LoanAmortizationCalculator lac = new LoanAmortizationCalculator();

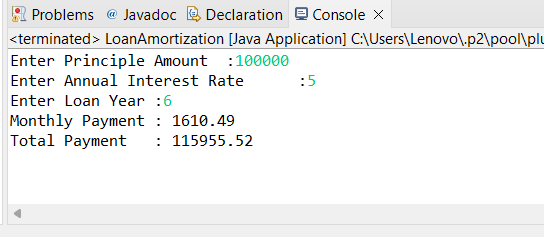
lac.acceptRecord();

lac.calculateMonthlyPayment();

lac.printRecord();

}

}



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

**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.

Code:

package org.example;

import java.util.Scanner;

class BMITracker {

float Height;

float Weight;

float BMIIndex;

//acceptRecord, calculateBMI, classifyBMI & printRecord

void acceptRecord() {

Scanner sc= new Scanner(System.in);

System.out.print("Enter your Weight (in KG) :");

this.Weight=sc.nextFloat();

System.out.print("Enter your Height (in Meter) :");

this.Height=sc.nextFloat();

}

void calculateBMI() {

BMIIndex= Weight / (Height \* Height);

}

void classifyBMI() {

if (BMIIndex<18.5) {

System.out.println("UnderWeight");

} else if (BMIIndex >=18.5 && BMIIndex<25) {

System.out.println("Normal Weight");

}else if (BMIIndex>=25 && BMIIndex<30) {

System.out.println("o Overweight");

}else if (BMIIndex>=30) {

System.out.println("Obese");

}

}

void printRecord() {

System.out.printf("BMI Index is : %.2f" ,this.BMIIndex);

System.out.println();

}

}

public class BMI {

public static void main(String[] args) {

BMITracker bmi= new BMITracker();

bmi.acceptRecord();

bmi.calculateBMI();

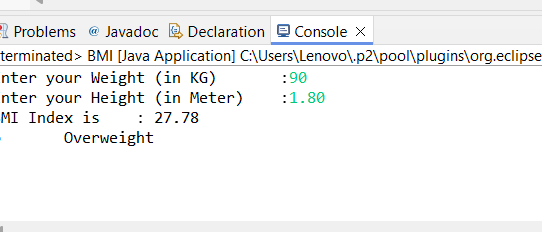
bmi.printRecord();

bmi.classifyBMI();

}

}

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



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

Code:

package org.example;

import java.util.Scanner;

class DiscountCalculator {

float Price;

float DisPercentage;

float discountAmount;

float finalPrice;

//acceptRecord, calculateDiscount & printRecord

void acceptRecord() {

Scanner sc= new Scanner(System.in);

System.out.print("Enter Price of Product :");

this.Price=sc.nextFloat();

System.out.print("Enter Discount Percentage :");

this.DisPercentage=sc.nextFloat();

}

void calculateDiscount() {

/\* Discount Amount Calculation: discountAmount = originalPrice \* (discountRate / 100)

Final Price Calculation: finalPrice = originalPrice - discountAmount \*/

discountAmount = Price \* (DisPercentage / 100);

finalPrice = Price - discountAmount;

}

void printRecord() {

System.out.printf("Discount Amount is : %.2f",this.discountAmount);

System.out.printf("\nFinal Price of Product :%.2f",this.finalPrice);

}

}

public class DiscountCalculation {

public static void main(String[] args) {

DiscountCalculator dc=new DiscountCalculator();

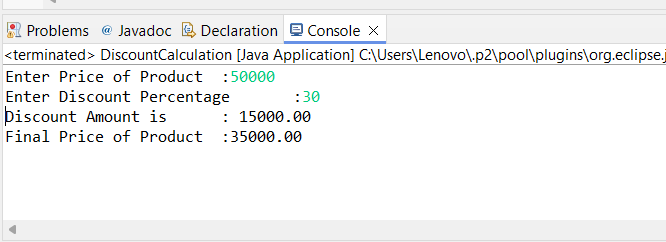
dc.acceptRecord();

dc.calculateDiscount();

dc.printRecord();

}

}



**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.

Code:

**package** org.example;

**import** java.util.Scanner;

**class** TBRM {

Scanner sc = **new** Scanner(System.***in***);

**float** TruckRate;

**float** CarRate;

**float** MotorcycleRate;

**int** TruckCount;

**int** CarCount;

**int** MotorCyclecount;

**float** totalRevenue;

// methods acceptRecord, setTollRates, calculateRevenue & printRecord

**void** acceptRecord() {

System.***out***.println("Enter The Count of Truck : ");

**this**.TruckCount = sc.nextInt();

System.***out***.println("Enter The Count of Car : ");

**this**.CarCount = sc.nextInt();

System.***out***.println("Enter The Count of MotorCycle : ");

**this**.MotorCyclecount = sc.nextInt();

}

**void** setTollRates() {

System.***out***.println("Enter The Rate of Truck : ");

**this**.TruckRate = sc.nextFloat();

System.***out***.println("Enter The Rate of Car : ");

**this**.CarRate = sc.nextFloat();

System.***out***.println("Enter The Rate of MotorCycle : ");

**this**.MotorcycleRate = sc.nextFloat();

}

**void** calculateRevenue() {

totalRevenue = (CarCount \* CarRate) + (TruckCount \* TruckRate) + (MotorCyclecount \* MotorcycleRate);

}

**void** printRecord() {

System.***out***.println("No of Trucks :" + TruckCount);

System.***out***.println("No of Cars :" + CarCount);

System.***out***.println("No of MotorCycle :" + MotorCyclecount);

System.***out***.println("totalRevenue generated :" + totalRevenue);

System.***out***.println("No of Trucks :" + TruckCount);

}

}

**public** **class** TollBoothRevenueManager {

**public** **static** **void** main(String[] args) {

TBRM manager = **new** TBRM();

manager.acceptRecord();

manager.setTollRates();

manager.calculateRevenue();

manager.printRecord();

}

}

