**Assignment:3**

**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.

package com.example;

import java.util.Scanner;

public class ques1 {

public static void main(String[] args) {

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

double principal;

double annualInterestRate;

int loanTerm;

System.***out***.print("Enter the principal amount (loan amount) in INR: ");

principal = scanner.nextDouble();

System.***out***.print("Enter the annual interest rate (in %): ");

annualInterestRate = scanner.nextDouble();

System.***out***.print("Enter the loan term (in years): ");

loanTerm = scanner.nextInt();

double monthlyInterestRate = annualInterestRate / 12 / 100;

int numberOfMonths = loanTerm \* 12;

double numerator = monthlyInterestRate \* Math.*pow*((1 + monthlyInterestRate), numberOfMonths);

double denominator = Math.*pow*((1 + monthlyInterestRate), numberOfMonths) -

double monthlyPayment = principal \* (numerator / denominator);

double totalAmountPaid = monthlyPayment \* numberOfMonths;

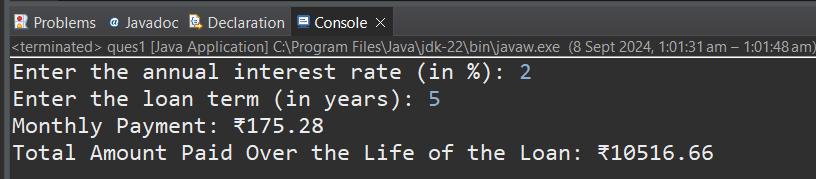
System.***out***.printf("Monthly Payment: ₹%.2f\n", monthlyPayment);

System.***out***.printf("Total Amount Paid Over the Life of the Loan: ₹%.2f\n", totalAmountPaid);

scanner.close();

}

}



**.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.

package org.example;

import java.util.Scanner;

public class ques2{

double principal;

double annualInterestRate;

int numberOfCompounds;

int years;

public void acceptRecord() {

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

System.*out*.print("Enter the initial investment amount (in INR): ");

principal = scanner.nextDouble();

System.*out*.print("Enter the annual interest rate (in %): ");

annualInterestRate = scanner.nextDouble();

System.*out*.print("Enter the number of times the interest is compounded per year: ");

numberOfCompounds = scanner.nextInt();

System.*out*.print("Enter the investment duration (in years): ");

years = scanner.nextInt();

}

public double calculateFutureValue() {

double ratePerCompound = annualInterestRate / 100 / numberOfCompounds;

it totalCompounds = numberOfCompounds \* years;

return principal \* Math.*pow*((1 + ratePerCompound), totalCompounds);

}

public void printRecord() {

double futureValue = calculateFutureValue();

double totalInterest = futureValue - principal;

System.*out*.printf("Future Value of Investment: ₹%.2f\n", futureValue);

System.*out*.printf("Total Interest Earned: ₹%.2f\n", totalInterest);

}

public static void main(String[] args) {

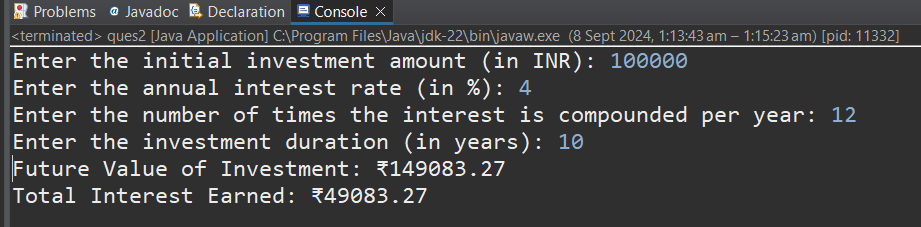
ques2 calculator = new ques2();

calculator.acceptRecord();

calculator.printRecord();

}

}



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

package com.present;

import java.util.Scanner;

public class ques3 {

double weight;

double height;

public void acceptRecord() {

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

System.***out***.print("Enter weight (in kilograms): ");

weight = scanner.nextDouble();

System.***out***.print("Enter height (in meters): ");

height = scanner.nextDouble(); }

public double calculateBMI() {

return weight / (height \* height);

}

public String classifyBMI(double bmi) {

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 void printRecord() {

double bmi = calculateBMI();

String classification = classifyBMI(bmi);

System.***out***.printf("BMI: %.2f\n", bmi);

System.***out***.println("Classification: " + classification);

}

public static void main(String[] args) {

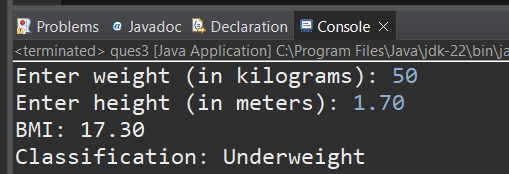
ques3 tracker = new ques3();

tracker.acceptRecord();

tracker.printRecord();

}

}



**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

package in.cdac;

import java.util.Scanner;

public class ques4 {

double originalPrice;

double discountRate;

public void acceptRecord() {

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

System.***out***.print("Enter the original price (in INR): ");

originalPrice = scanner.nextDouble();

System.***out***.print("Enter the discount percentage: ");

discountRate = scanner.nextDouble();

}

public double calculateDiscount() {

return originalPrice \* (discountRate / 100);

}

public void printRecord() {

double discountAmount = calculateDiscount();

double finalPrice = originalPrice - discountAmount;

System.***out***.printf("Discount Amount: ₹%.2f\n", discountAmount);

System.***out***.printf("Final Price: ₹%.2f\n", finalPrice);

}

public static void main(String[] args) {

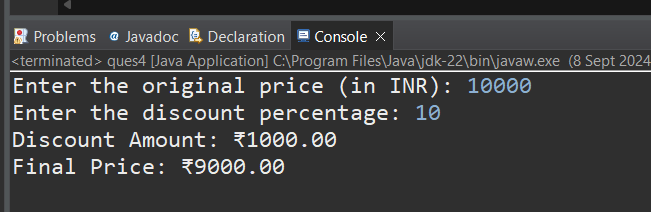
ques4 calculator = new ques4();

calculator.acceptRecord();

calculator.printRecord();

}

}



**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 in.cdac;

import java.util.Scanner;

public class ques5 {

private double carRate;

private double truckRate;

private double motorcycleRate;

private int carCount;

private int truckCount;

private int motorcycleCount;

public void setTollRates() {

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

System.***out***.print("Enter toll rate for Car (in INR): ");

carRate = scanner.nextDouble();

System.***out***.print("Enter toll rate for Truck (in INR): ");

truckRate = scanner.nextDouble();

System.***out***.print("Enter toll rate for Motorcycle (in INR): ");

motorcycleRate = scanner.nextDouble();

}

public void acceptRecord() {

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

System.***out***.print("Enter the number of Cars: ");

carCount = scanner.nextInt();

System.***out***.print("Enter the number of Trucks: ");

truckCount = scanner.nextInt();

System.***out***.print("Enter the number of Motorcycles: ");

motorcycleCount = scanner.nextInt();

}

public double calculateRevenue() {

return (carCount \* carRate) + (truckCount \* truckRate) + (motorcycleCount \* motorcycleRate);

}

public void printRecord() {

int totalVehicles = carCount + truckCount + motorcycleCount;

double totalRevenue = calculateRevenue();

System.***out***.println("\n----- Toll Booth Summary -----");

System.***out***.println("Total number of vehicles: " + totalVehicles);

System.***out***.printf("Total revenue collected: ₹%.2f\n", totalRevenue);

}

public static void main(String[] args) {

ques5 tollBoothManager = new ques5();

tollBoothManager.setTollRates();

tollBoothManager.acceptRecord();

tollBoothManager.printRecord();

}

}