Assignment 3.2

**Name:** Yash Bandu Dhole

**Center:** Juhu.

**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 problem1;

import java.util.Scanner;

class LoanAmortizationCalculator {

// Fields to store loan details

double principal;

double annualInterestRate;

int loanTermYears;

// Method to accept user input for loan details

public void acceptRecord() {

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

System.***out***.print("Enter the loan amount (Principal): ₹");

principal = sc.nextDouble();

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

annualInterestRate = sc.nextDouble();

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

loanTermYears = sc.nextInt();

}

// Method to calculate the monthly payment

public double calculateMonthlyPayment() {

double monthlyInterestRate = annualInterestRate / 12 / 100;

int numberOfMonths = loanTermYears \* 12;

// Applying the standard mortgage formula

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

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

return monthlyPayment;

}

// Method to display the monthly payment and total amount paid

public void printRecord(double monthlyPayment) {

int numberOfMonths = loanTermYears \* 12; // Total number of months

double totalAmountPaid = monthlyPayment \* numberOfMonths;

System.***out***.printf("Your monthly payment is: ₹%.2f\n", monthlyPayment);

System.***out***.printf("Total amount paid over the life of the loan: ₹%.2f\n", totalAmountPaid);

}

}

// Test class to run the application

public class LoanAmortizationTest {

public static void main(String[] args) {

// Creating an instance of LoanAmortizationCalculator

LoanAmortizationCalculator calculator = new LoanAmortizationCalculator();

// Accepting user input

calculator.acceptRecord();

// Calculating the monthly payment

double monthlyPayment = calculator.calculateMonthlyPayment();

// Displaying the result

calculator.printRecord(monthlyPayment);

}

}

**o/p:**

Enter the loan amount (Principal): ₹90000

Enter the annual interest rate (in %): 8

Enter the loan term (in years): 3

Your monthly payment is: ₹2820.27

Total amount paid over the life of the loan: ₹101529.82

**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.

**Code:**

package problem2;

import java.util.Scanner;

class CompoundInterestCalculator {

// Fields to store investment details

double principal;

double annualInterestRate;

int numberOfCompounds;

int years;

// Method to accept user input for investment details

public void acceptRecord() {

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

System.***out***.print("Enter the initial investment amount (Principal): ₹");

principal = sc.nextDouble();

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

annualInterestRate = sc.nextDouble();

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

numberOfCompounds = sc.nextInt();

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

years = sc.nextInt();

}

// Method to calculate the future value of the investment

public double calculateFutureValue() {

double ratePerCompound = annualInterestRate / 100 / numberOfCompounds;

double futureValue = principal \* Math.*pow*(1 + ratePerCompound, numberOfCompounds \* years);

return futureValue;

}

// Method to display the future value and total interest earned

public void printRecord(double futureValue) {

double totalInterest = futureValue - principal;

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

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

}

}

// Test class to run the application

public class CompoundInterestTest {

public static void main(String[] args) {

CompoundInterestCalculator calculator = new CompoundInterestCalculator();

calculator.acceptRecord();

double futureValue = calculator.calculateFutureValue();

calculator.printRecord(futureValue);

}

}

**o/p:**

Enter the initial investment amount (Principal): ₹1000000

Enter the annual interest rate (in %): 7.5

Enter the number of times the interest is compounded per year: 2

Enter the investment duration (in years): 1

Future value of your investment: ₹1076406.25

Total interest earned: ₹76406.25

**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.

**Code:**

package problem3;

import java.util.Scanner;

class BMITracker {

// Fields to store weight and height

double weight;

double height;

// Method to accept user input for weight and height

public void acceptRecord() {

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

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

weight = sc.nextDouble();

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

height = sc.nextDouble();

}

// Method to calculate BMI

public double calculateBMI() {

return weight / (height \* height);

}

// Method to classify BMI based on the calculated value

public String classifyBMI(double bmi) {

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";

}

}

// Method to print the BMI and classification

public void printRecord(double bmi, String classification) {

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

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

}

}

// Test class to run the application

public class BMITrackerTest {

public static void main(String[] args) {

BMITracker tracker = new BMITracker();

tracker.acceptRecord();

double bmi = tracker.calculateBMI();

String classification = tracker.classifyBMI(bmi);

tracker.printRecord(bmi, classification);

}

}

**o/p:**

Enter your weight (in kilograms): 65

Enter your height (in meters): 1.52

Your BMI is: 28.13

BMI Classification: Overweight

**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 problem4;

import java.util.Scanner;

class DiscountCalculator {

double originalPrice;

double discountRate;

double discountAmount;

double finalPrice;

// Method to accept input from the user

void acceptRecord() {

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

System.***out***.print("Enter the original price of the item: ₹");

originalPrice = scanner.nextDouble();

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

discountRate = scanner.nextDouble();

}

// Method to calculate discount and final price

void calculateDiscount() {

discountAmount = originalPrice \* (discountRate / 100);

finalPrice = originalPrice - discountAmount;

}

// Method to print the discount amount and final price

void printRecord() {

System.***out***.println("Discount Amount: ₹" + discountAmount);

System.***out***.println("Final Price after Discount: ₹" + finalPrice);

}

}

public class Main {

public static void main(String[] args) {

DiscountCalculator calculator = new DiscountCalculator();

calculator.acceptRecord();

calculator.calculateDiscount();

calculator.printRecord();

}

}

**o/p:**

Enter the original price of the item: ₹15

Enter the discount percentage: 3

Discount Amount: ₹0.44999999999999996

Final Price after Discount: ₹14.55

**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 problem5;

import java.util.Scanner;

class TollBoothRevenueManager {

double carTollRate;

double truckTollRate;

double motorcycleTollRate;

int carCount;

int truckCount;

int motorcycleCount;

double totalRevenue;

void setTollRates() {

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

System.***out***.print("Enter the toll rate for Cars: ₹");

carTollRate = scanner.nextDouble();

System.***out***.print("Enter the toll rate for Trucks: ₹");

truckTollRate = scanner.nextDouble();

System.***out***.print("Enter the toll rate for Motorcycles: ₹");

motorcycleTollRate = scanner.nextDouble();

}

void acceptRecord() {

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

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

carCount = scanner.nextInt();

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

truckCount = scanner.nextInt();

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

motorcycleCount = scanner.nextInt();

}

void calculateRevenue() {

totalRevenue = (carCount \* carTollRate) + (truckCount \* truckTollRate) + (motorcycleCount \* motorcycleTollRate);

}

void printRecord() {

int totalVehicles = carCount + truckCount + motorcycleCount;

System.***out***.println("Total Vehicles: " + totalVehicles);

System.***out***.println("Total Revenue Collected: ₹" + totalRevenue);

}

}

public class TollBooth {

public static void main(String[] args) {

TollBoothRevenueManager tollBooth = new TollBoothRevenueManager();

tollBooth.setTollRates();

tollBooth.acceptRecord();

tollBooth.calculateRevenue();

tollBooth.printRecord();

}

}

**o/p:**

Enter the toll rate for Cars: ₹15

Enter the toll rate for Trucks: ₹280

Enter the toll rate for Motorcycles: ₹0

Enter the number of Cars passing: 15000

Enter the number of Trucks passing: 10000

Enter the number of Motorcycles passing: 54896

Total Vehicles: 79896

Total Revenue Collected: ₹3025000.0