**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.

package org.cdac;

public class LoanAmortizationCalculator {

private double principal;

private double annualInterestRate;

private int loanTerm; // in years

public LoanAmortizationCalculator(double principal, double annualInterestRate, int loanTerm) {

this.principal = principal;

this.annualInterestRate = annualInterestRate;

this.loanTerm = loanTerm;

}

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 getLoanTerm() {

return loanTerm;

}

public void setLoanTerm(int loanTerm) {

this.loanTerm = loanTerm;

}

public double calculateMonthlyPayment() {

double monthlyInterestRate = (annualInterestRate / 12) / 100;

int numberOfMonths = loanTerm \* 12;

return principal \* (monthlyInterestRate \* Math.*pow*(1 + monthlyInterestRate, numberOfMonths)) /

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

}

// calulate payment

public double calculateTotalPayment() {

return calculateMonthlyPayment() \* loanTerm \* 12;

}

*@Override*

public String toString() {

return "Principal: ₹" + principal + ", Annual Interest Rate: " + annualInterestRate +

"%, Loan Term: " + loanTerm + " years";

}

}

**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.

package org.cdac;

public class CompoundInterestCalculator {

private double principal;

private double annualInterestRate;

private int numberOfCompounds;

private int years;

public CompoundInterestCalculator(double principal, double annualInterestRate, int numberOfCompounds, int years) {

this.principal = principal;

this.annualInterestRate = annualInterestRate;

this.numberOfCompounds = numberOfCompounds;

this.years = years;

}

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;

}

public double calculateFutureValue() {

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

}

public double calculateTotalInterest() {

return calculateFutureValue() - principal;

}

*@Override*

public String toString() {

return String.*format*("Principal: ₹%.2f, Annual Interest Rate: %.2f%%, Compounds per Year: %d, Duration: %d years",

principal, annualInterestRate \* 100, numberOfCompounds, years);

}

}

package org.cdac;

import java.util.Scanner;

public class CompoundInterestCalculatorUtil {

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

public CompoundInterestCalculator acceptRecord() {

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

double principal = scanner.nextDouble();

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

double annualInterestRate = scanner.nextDouble() / 100;

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

int numberOfCompounds = scanner.nextInt();

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

int years = scanner.nextInt();

return new CompoundInterestCalculator(principal, annualInterestRate, numberOfCompounds, years);

}

public void printRecord(CompoundInterestCalculator calculator) {

double futureValue = calculator.calculateFutureValue();

double totalInterest = calculator.calculateTotalInterest();

System.***out***.println("\nInvestment Details:");

package org.cdac;

public class Programm {

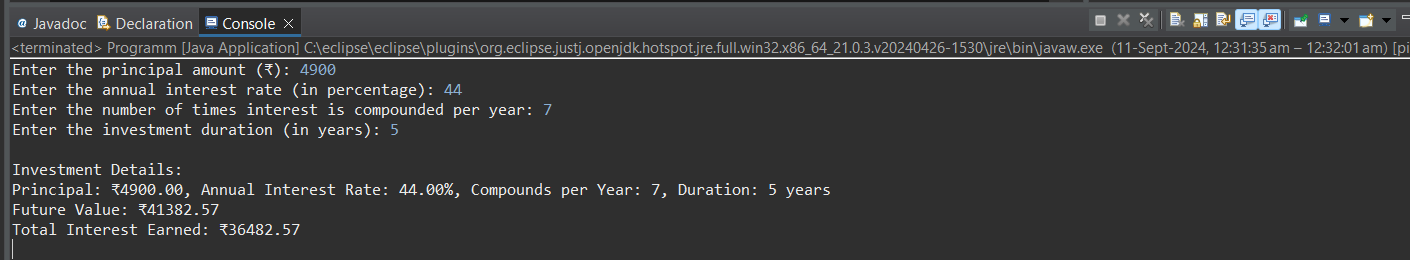
public static void main(String[] args) {

CompoundInterestCalculatorUtil util = new CompoundInterestCalculatorUtil();

util.menuList();

}

}



**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.

package org.cdac.c;

public class BMITracker {

private double weight; // in kilograms

private double height; // in meters

public BMITracker(double weight, double height) {

this.weight = weight;

this.height = height;

}

public double getWeight() {

return weight;

}

public void setWeight(double weight) {

this.weight = weight;

}

public double getHeight() {

return height;

}

public void setHeight(double height) {

this.height = height;

}

public double calculateBMI() {

return weight / (height \* height);

}

public String classifyBMI() {

double bmi = calculateBMI();

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";

}

}

*@*Override

public String toString() {

return "Weight: " + weight + " kg, Height: " + height + " meters, BMI: " + String.format("%.2f", calculateBMI()) +

", Classification: " + classifyBMI();

}

}

}

package org.cdac.c;

import java.util.Scanner;

public class BMITrackerUtil {

public static BMITracker acceptRecord() {

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

System.***out***.print("Enter Weight (kg): ");

double weight = sc.nextDouble();

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

double height = sc.nextDouble();

return new BMITracker(weight, height);

}

public static void printRecord(BMITracker bmiTracker) {

System.***out***.println("\nBMI Details:");

System.***out***.println(bmiTracker.toString());

}

public static void menuList() {

System.***out***.println("\nBMI Tracker");

System.***out***.println("1. Enter BMI Details");

System.***out***.println("2. Display BMI Details");

System.***out***.println("3. Exit");

}

}

package org.cdac.c;

import java.util.Scanner;

public class program {

public static void main(String[] args) {

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

BMITracker bmiTracker = null;

int choice;

do {

BMITrackerUtil.*menuList*();

System.***out***.print("Enter your choice: ");

choice = sc.nextInt();

switch (choice) {

case 1:

bmiTracker = BMITrackerUtil.*acceptRecord*();

break;

case 2:

if (bmiTracker != null) {

BMITrackerUtil.*printRecord*(bmiTracker);

} else {

System.***out***.println("Please enter BMI details first!");

}

break;

case 3:

System.***out***.println("Exiting program...");

break;

default:

System.***out***.println("Invalid choice, please try again.");

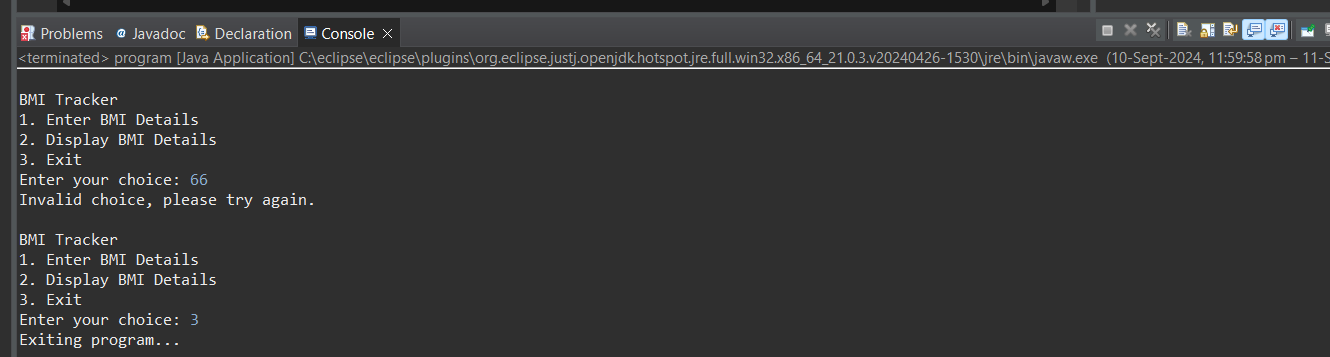
}

} while (choice != 3);

sc.close();

}

}



**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.

package com.cdac;

import java.util.Scanner;

public class Program {

public static void main(String[] args) {

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

DiscountCalculatorUtil util = new DiscountCalculatorUtil();

int choice;

do {

util.menuList();

System.***out***.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

util.acceptRecord();

break;

case 2:

util.printRecord();

break;

case 0:

System.***out***.println("Exiting...");

break;

default:

System.***out***.println("Invalid choice. Please try again.");

}

} while (choice != 0);

scanner.close();

}

}

package com.cdac;

import java.util.Scanner;

public class DiscountCalculatorUtil {

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

private DiscountCalculator discountCalculator;

// Method to accept input from the user

public void acceptRecord() {

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

double originalPrice = scanner.nextDouble();

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

double discountRate = scanner.nextDouble();

discountCalculator = new DiscountCalculator(originalPrice, discountRate);

}

public void printRecord() {

if (discountCalculator != null) {

System.***out***.println(discountCalculator);

} else {

System.***out***.println("No record to display.");

}

}

// Menu display

public void menuList() {

System.***out***.println("1. Accept Record");

System.***out***.println("2. Print Record");

System.***out***.println("0. Exit");

}

}package com.cdac;

public class DiscountCalculator {

private double originalPrice;

private double discountRate;

private double discountAmount;

private double finalPrice;

public DiscountCalculator(double originalPrice, double discountRate) {

this.originalPrice = originalPrice;

this.discountRate = discountRate;

calculateDiscount(); // Calculate discountAmount and finalPrice when object is created

}

// Getter and Setter methods

public double getOriginalPrice() {

return originalPrice;

}

public void setOriginalPrice(double originalPrice) {

this.originalPrice = originalPrice;

calculateDiscount(); // Recalculate on updating price

}

public double getDiscountRate() {

return discountRate;

}

public void setDiscountRate(double discountRate) {

this.discountRate = discountRate;

calculateDiscount(); // Recalculate on updating discount rate

}

public double getDiscountAmount() {

return discountAmount;

}

public double getFinalPrice() {

return finalPrice;

}

private void calculateDiscount() {

discountAmount = originalPrice \* (discountRate / 100);

finalPrice = originalPrice - discountAmount;

}

// toString method to display the result

*@Override*

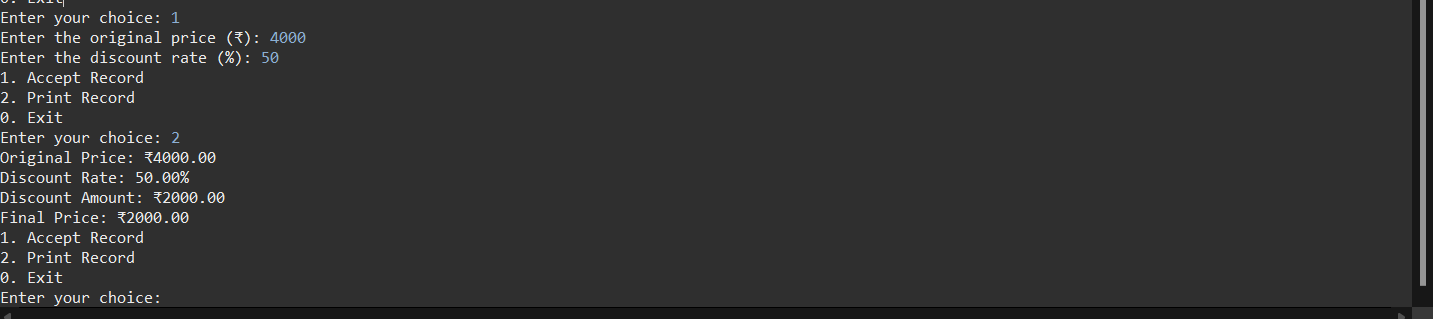
public String toString() {

return String.*format*("Original Price: ₹%.2f\nDiscount Rate: %.2f%%\nDiscount Amount: ₹%.2f\nFinal Price: ₹%.2f",

originalPrice, discountRate, discountAmount, finalPrice);

}

}



**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.

package cdac.org;

public class TollBoothRevenueManager {

private double carTollRate;

private double truckTollRate;

private double motorcycleTollRate;

private int carCount;

private int truckCount;

private int motorcycleCount;

public TollBoothRevenueManager(double carTollRate, double truckTollRate, double motorcycleTollRate) {

this.carTollRate = carTollRate;

this.truckTollRate = truckTollRate;

this.motorcycleTollRate = motorcycleTollRate;

this.carCount = 0;

this.truckCount = 0;

this.motorcycleCount = 0;

}

// Getter and Setter Methods

public double getCarTollRate() { return carTollRate; }

public void setCarTollRate(double carTollRate) { this.carTollRate = carTollRate; }

public double getTruckTollRate() { return truckTollRate; }

public void setTruckTollRate(double truckTollRate) { this.truckTollRate = truckTollRate; }

public double getMotorcycleTollRate() { return motorcycleTollRate; }

public void setMotorcycleTollRate(double motorcycleTollRate) { this.motorcycleTollRate = motorcycleTollRate; }

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; }

public double calculateTotalRevenue() {

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

}

public int getTotalVehicles() {

return carCount + truckCount + motorcycleCount;

}

*@Override*

public String toString() {

return "Total Vehicles: " + getTotalVehicles() + "\n" +

"Total Revenue: ₹" + calculateTotalRevenue();

}

}

package cdac.org;

import java.util.Scanner;

public class TollBoothRevenueManagerUtil {

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

public void acceptRecord(TollBoothRevenueManager manager) {

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

manager.setCarCount(scanner.nextInt());

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

manager.setTruckCount(scanner.nextInt());

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

manager.setMotorcycleCount(scanner.nextInt());

}

public void printRecord(TollBoothRevenueManager manager) {

System.***out***.println(manager);

}

public void menuList(TollBoothRevenueManager manager) {

int choice;

do {

System.***out***.println("\nToll Booth Revenue Management");

System.***out***.println("1. Set Toll Rates");

System.***out***.println("2. Accept Vehicle Count");

System.***out***.println("3. Display Revenue and Total Vehicles");

System.***out***.println("4. Exit");

System.***out***.print("Enter choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

setTollRates(manager);

break;

case 2:

acceptRecord(manager);

break;

case 3:

printRecord(manager);

break;

case 4:

System.***out***.println("Exiting...");

break;

default:

System.***out***.println("Invalid choice, please try again.");

}

} while (choice != 4);

}

private void setTollRates(TollBoothRevenueManager manager) {

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

manager.setCarTollRate(scanner.nextDouble());

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

manager.setTruckTollRate(scanner.nextDouble());

System.***out***.println("Enter Motorcycle Toll Rate:");

manager.setMotorcycleTollRate(scanner.nextDouble());

}

}

package cdac.org;

public class Program {

public static void main(String[] args) {

TollBoothRevenueManager manager = new TollBoothRevenueManager(50.0, 100.0, 30.0); // Default rates

TollBoothRevenueManagerUtil util = new TollBoothRevenueManagerUtil();

util.menuList(manager);

}

}

