**Assignment:4**

**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.solution;

import java.util.Scanner;

class LoanCalculator {

private double principal;

private double annualInterestRate;

private int loanTerm;

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

this.principal = principal;

this.annualInterestRate = annualInterestRate;

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);

}

public double calculateTotalPayment() {

return calculateMonthlyPayment() \* loanTerm \* 12;

}

}

class LoanAmortizationCalculatorUtil {

private LoanCalculator loan;

public void acceptRecord() {

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

System.***out***.print("Enter Principal Amount: ");

double principal = sc.nextDouble();

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

double annualInterestRate = sc.nextDouble();

System.***out***.print("Enter Loan Term (in years): ");

int loanTerm = sc.nextInt();

loan = new LoanCalculator(principal, annualInterestRate, loanTerm);

}

public void printRecord() {

if (loan != null) {

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

System.***out***.printf("Total Payment: ₹%.2f%n", loan.calculateTotalPayment());

} else {

System.***out***.println("No loan record available.");

}

}

public void menuList() {

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

int choice;

do {

System.***out***.println("1. Accept Loan Details");

System.***out***.println("2. Print Loan Details");

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

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

choice = sc.nextInt();

switch (choice) {

case 1:

acceptRecord();

break;

case 2:

printRecord();

break;

}

} while (choice != 3);

}

}

public class ass4qs1 {

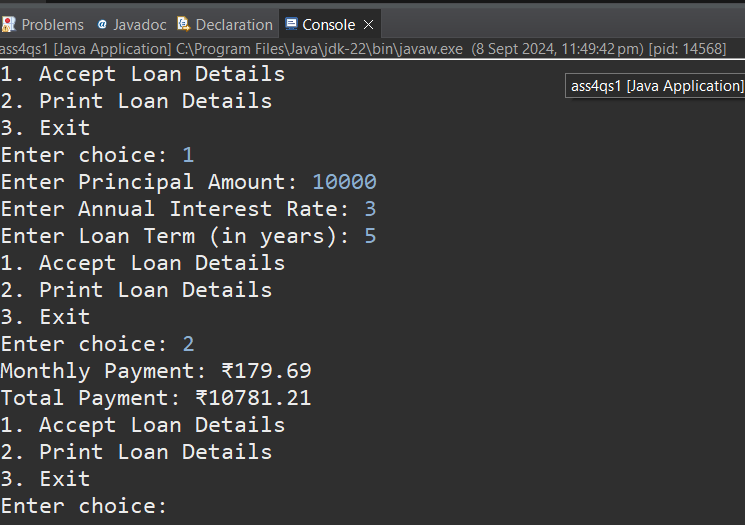
public static void main(String[] args) {

LoanAmortizationCalculatorUtil util = new LoanAmortizationCalculatorUtil();

util.menuList();

}

}



**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.solution;

import java.util.Scanner;

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 calculateFutureValue() {

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

}

public double calculateTotalInterest() {

return calculateFutureValue() - principal;

}

// Getters and Setters

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;

}

@Override

public String toString() {

return String.format("Principal: ₹%.2f%nAnnual Interest Rate: %.2f%%%nNumber of Compounds per Year: %d%nInvestment Duration: %d years%n",

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

}

}

class CompoundInterestCalculatorUtil {

private CompoundInterestCalculator investment;

public void acceptRecord() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Initial Investment Amount: ");

double principal = sc.nextDouble();

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

double annualInterestRate = sc.nextDouble();

System.out.print("Enter Number of Times Interest is Compounded Per Year: ");

int numberOfCompounds = sc.nextInt();

System.out.print("Enter Investment Duration (in years): ");

int years = sc.nextInt();

investment = new CompoundInterestCalculator(principal, annualInterestRate / 100, numberOfCompounds, years);

}

public void printRecord() {

if (investment != null) {

System.out.println(investment);

System.out.printf("Future Value: ₹%.2f%n", investment.calculateFutureValue());

System.out.printf("Total Interest Earned: ₹%.2f%n", investment.calculateTotalInterest());

} else {

System.out.println("No investment record available.");

}

}

public void menuList() {

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("1. Accept Investment Details");

System.out.println("2. Print Investment Details");

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

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

choice = sc.nextInt();

switch (choice) {

case 1:

acceptRecord();

break;

case 2:

printRecord();

break;

}

} while (choice != 3);

}

}

public class ass4qs2 {

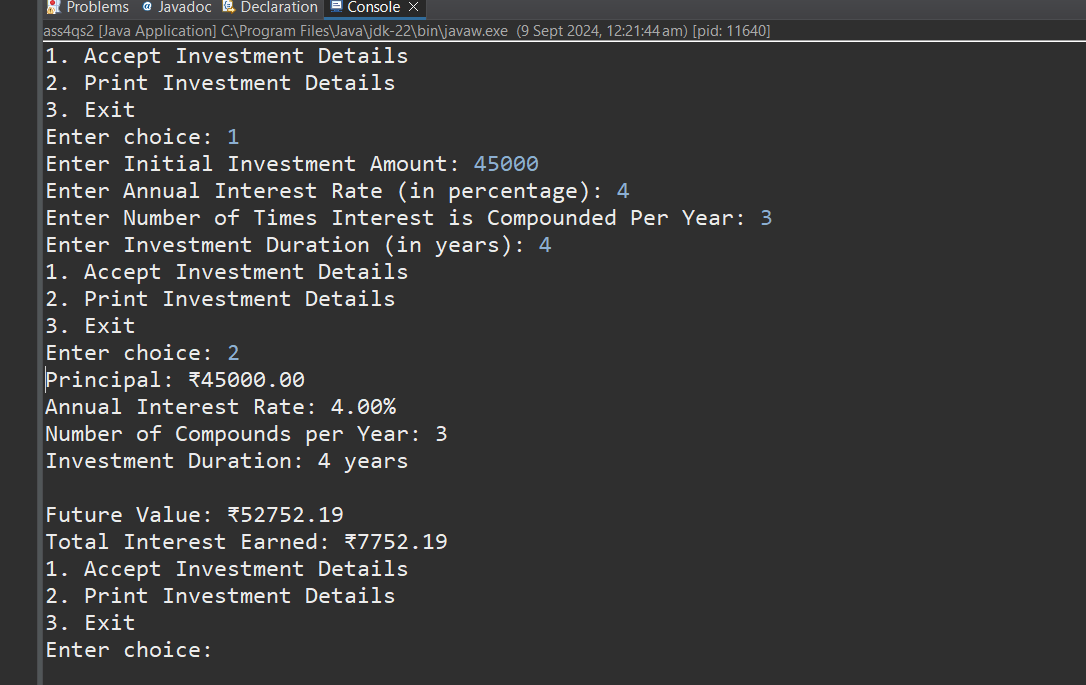
public static void main(String[] args) {

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.solution;

import java.util.Scanner;

class BMITracker {

private double weight;

private double height;

public BMITracker(double weight, double height) {

this.weight = weight;

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 < 24.9) {

return "Normal weight";

} else if (bmi < 29.9) {

return "Overweight";

} else {

return "Obese";

}

}

// Getters and Setters

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;

}

@Override

public String toString() {

return String.format("Weight: %.2f kg%nHeight: %.2f m%nBMI: %.2f%nClassification: %s",

weight, height, calculateBMI(), classifyBMI());

}

}

class BMITrackerUtil {

private BMITracker bmiTracker;

public void acceptRecord() {

Scanner sc = new Scanner(System.in);

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

double weight = sc.nextDouble();

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

double height = sc.nextDouble();

bmiTracker = new BMITracker(weight, height);

}

public void printRecord() {

if (bmiTracker != null) {

System.out.println(bmiTracker);

} else {

System.out.println("No BMI record available.");

}

}

public void menuList() {

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("1. Accept BMI Details");

System.out.println("2. Print BMI Details");

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

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

choice = sc.nextInt();

switch (choice) {

case 1:

acceptRecord();

break;

case 2:

printRecord();

break;

}

} while (choice != 3);

}

}

public class Program {

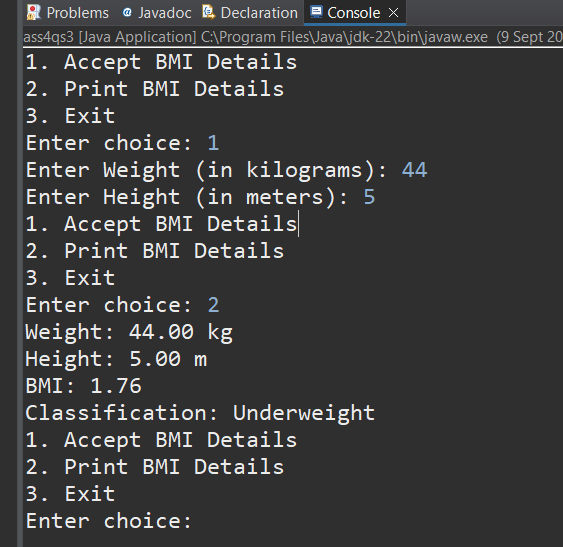
public static void main(String[] args) {

BMITrackerUtil util = new BMITrackerUtil();

util.menuList();

}

}



**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 org.solution;

import java.util.Scanner;

class DiscountCalculator {

private double originalPrice;

private double discountRate;

public DiscountCalculator(double originalPrice, double discountRate) {

this.originalPrice = originalPrice;

this.discountRate = discountRate;

}

public double calculateDiscountAmount() {

return originalPrice \* (discountRate / 100);

}

public double calculateFinalPrice() {

return originalPrice - calculateDiscountAmount();

}

// Getters and Setters

public double getOriginalPrice() {

return originalPrice;

}

public void setOriginalPrice(double originalPrice) {

this.originalPrice = originalPrice;

}

public double getDiscountRate() {

return discountRate;

}

public void setDiscountRate(double discountRate) {

this.discountRate = discountRate;

}

*@Override*

public String toString() {

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

originalPrice, discountRate, calculateDiscountAmount(), calculateFinalPrice());

}

}

class DiscountCalculatorUtil {

private DiscountCalculator discountCalculator;

public void acceptRecord() {

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

System.***out***.print("Enter Original Price (in INR): ");

double originalPrice = sc.nextDouble();

System.***out***.print("Enter Discount Rate (in percentage): ");

double discountRate = sc.nextDouble();

discountCalculator = new DiscountCalculator(originalPrice, discountRate);

}

public void printRecord() {

if (discountCalculator != null) {

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

} else {

System.***out***.println("No discount record available.");

}

}

public void menuList() {

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

int choice;

do {

System.***out***.println("1. Accept Discount Details");

System.***out***.println("2. Print Discount Details");

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

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

choice = sc.nextInt();

switch (choice) {

case 1:

acceptRecord();

break;

case 2:

printRecord();

break;

}

} while (choice != 3);

}

}

public class ass4qs4{

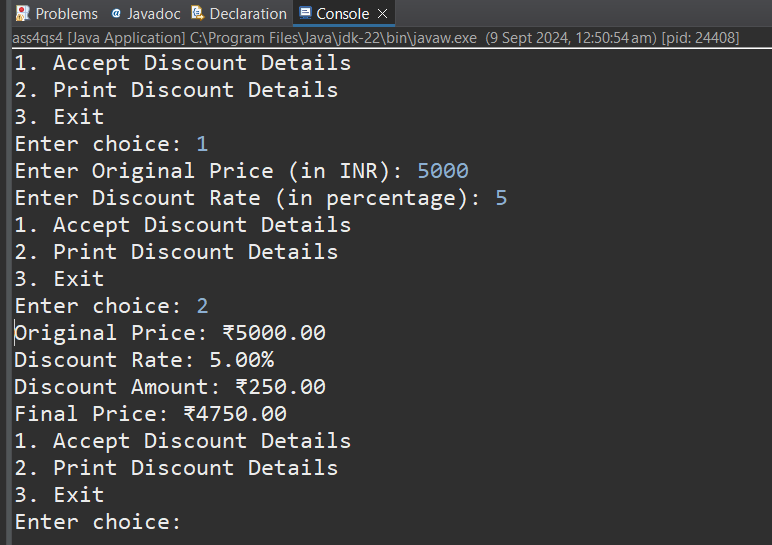
public static void main(String[] args) {

DiscountCalculatorUtil util = new DiscountCalculatorUtil();

util.menuList();1

}

}



**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 org.solution;

import java.util.Scanner;

public class ass4qs5 {

private double carRate;

private double truckRate;

private double motorcycleRate;

private int carCount;

private int truckCount;

private int motorcycleCount;

public ass4qs5(double carRate, double truckRate, double motorcycleRate) {

this.carRate = carRate;

this.truckRate = truckRate;

this.motorcycleRate = motorcycleRate;

}

public double calculateTotalRevenue() {

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

}

public int getTotalVehicleCount() {

return carCount + truckCount + motorcycleCount;

}

public double getCarRate() {

return carRate;

}

public void setCarRate(double carRate) {

this.carRate = carRate;

}

public double getTruckRate() {

return truckRate;

}

public void setTruckRate(double truckRate) {

this.truckRate = truckRate;

}

public double getMotorcycleRate() {

return motorcycleRate;

}

public void setMotorcycleRate(double motorcycleRate) {

this.motorcycleRate = motorcycleRate;

}

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;

}

*@Override*

public String toString() {

return String.*format*("Car Rate: ₹%.2f%nTruck Rate: ₹%.2f%nMotorcycle Rate: ₹%.2f%n" +

"Cars: %d%nTrucks: %d%nMotorcycles: %d%n" +

"Total Vehicles: %d%nTotal Revenue: ₹%.2f",

carRate, truckRate, motorcycleRate, carCount, truckCount, motorcycleCount,

getTotalVehicleCount(), calculateTotalRevenue());

}

public static void main(String[] args) {

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

System.***out***.print("Enter Toll Rate for Car: ₹");

double carRate = sc.nextDouble();

System.***out***.print("Enter Toll Rate for Truck: ₹");

double truckRate = sc.nextDouble();

System.***out***.print("Enter Toll Rate for Motorcycle: ₹");

double motorcycleRate = sc.nextDouble();

ass4qs5 tollManager = new ass4qs5(carRate, truckRate, motorcycleRate);

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

int carCount = sc.nextInt();

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

int truckCount = sc.nextInt();

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

int motorcycleCount = sc.nextInt();

tollManager.setCarCount(carCount);

tollManager.setTruckCount(truckCount);

tollManager.setMotorcycleCount(motorcycleCount);

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

}

}

