**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.

**Code:**

package assignment4.org;

import java.util.Scanner;

class LoanAmortizationCalculator {

private double principalAmount;

private double annualInterestRate;

private double monthlyPayment;

private int loanterm;

private int numberMonths;

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

this.principalAmount = principal;

this.annualInterestRate = annualInterestRate;

this.loanterm = loanTerm;

}

public double getPrincipalAmount() {

return principalAmount;

}

public void setPrincipalAmount(double principalAmount) {

this.principalAmount = principalAmount;

}

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 calculateMonthlyAmount() {

double monthlyRoi = (annualInterestRate / 12) / 100;

numberMonths = loanterm \* 12;

monthlyPayment = principalAmount \* (monthlyRoi \* Math.*pow*(1 + monthlyRoi, numberMonths)) / (Math.*pow*(1 + monthlyRoi, numberMonths) - 1);

return monthlyPayment;

}

public double totAmount() {

double totalAmount = monthlyPayment \* numberMonths;

return totalAmount;

}

// @Override

// public String toString() {

// return "LoanAmortizationCalculator [Principal Amount= ₹" + principalAmount + ", Annual Interest Rate=" + annualInterestRate + "%, Loan Term=" + loanterm + " years]";

// }

}

class LoanAmortizationCalculatorUtil {

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

public LoanAmortizationCalculator acceptRecord() {

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

double principal = scanner.nextDouble();

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

double annualInterestRate = scanner.nextDouble();

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

int loanTerm = scanner.nextInt();

return new LoanAmortizationCalculator(principal, annualInterestRate, loanTerm);

}

public void printRecord(LoanAmortizationCalculator calculator) {

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

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

System.***out***.printf("Total Payment over the life of the loan: ₹%.2f%n", calculator.totAmount());

}

public void menuList() {

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

System.***out***.println("2. Display Loan Details and Calculate Payment");

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

}

}

class Program {

public static void main(String[] args) {

LoanAmortizationCalculatorUtil util = new LoanAmortizationCalculatorUtil();

LoanAmortizationCalculator calculator = null;

try (Scanner sc = new Scanner(System.***in***)) {

while (true) {

util.menuList();

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

int choice = sc.nextInt();

switch (choice) {

case 1:

calculator = util.acceptRecord();

break;

case 2:

if (calculator != null) {

util.printRecord(calculator);

} else {

System.***out***.println("No loan details entered yet.");

}

break;

case 3:

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

return;

default:

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

}

}

}

}

}

**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.

**Code:**

package ass4.org;

import java.util.Scanner;

class CompoundInterestCalculator {

private double initInvestment;

private double annualRoi;

private double numTimesRoiCompounded;

private double invesDuration;

private double futureValue;

private double totalInterest;

public CompoundInterestCalculator(double initInvestment, double annualRoi, double numTimesRoiCompounded, double invesDuration) {

this.initInvestment = initInvestment;

this.annualRoi = annualRoi;

this.numTimesRoiCompounded = numTimesRoiCompounded;

this.invesDuration = invesDuration;

}

public double getFutureValue() {

futureValue = initInvestment \* Math.*pow*(1 + (annualRoi / numTimesRoiCompounded), (numTimesRoiCompounded \* invesDuration));

return futureValue;

}

public double getTotalInterest() {

totalInterest = getFutureValue() - initInvestment;

return totalInterest;

}

*@Override*

public String toString() {

return "CompoundInterestCalculator [Initial Investment= ₹" + initInvestment + ", Annual Rate of Interest= " + annualRoi +

"%, Compounding Times per Year= " + numTimesRoiCompounded + ", Investment Duration= " + invesDuration + " years]";

}

}

class CompoundInterestCalculatorUtil {

public CompoundInterestCalculator acceptRecord() {

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

System.***out***.println("Initial Investment: ");

double initInvestment = sc.nextDouble();

System.***out***.println("Annual Rate Of Interest: ");

double annualRoi = sc.nextDouble();

System.***out***.println("Compounding Times per Year: ");

double numTimesRoiCompounded = sc.nextDouble();

System.***out***.println("Investment Duration in Years: ");

double invesDuration = sc.nextDouble();

return new CompoundInterestCalculator(initInvestment, annualRoi, numTimesRoiCompounded, invesDuration);

}

public void printRecord(CompoundInterestCalculator calculator) {

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

System.***out***.printf("Future Value : ₹%-20.2f%n", calculator.getFutureValue());

System.***out***.printf("Total Interest : ₹%-20.2f%n", calculator.getTotalInterest());

}

public void menuList() {

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

System.***out***.println("2. Display Compound Interest and Calculate Future Value");

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

}

}

public class Program {

public static void main(String[] args) {

CompoundInterestCalculatorUtil util = new CompoundInterestCalculatorUtil();

CompoundInterestCalculator calculator = null;

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

while (true) {

util.menuList();

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

int choice = sc.nextInt();

switch (choice) {

case 1:

calculator = util.acceptRecord();

break;

case 2:

if (calculator != null) {

util.printRecord(calculator);

} else {

System.***out***.println("No investment details entered yet.");

}

break;

case 3:

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

return;

default:

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

}

}

}

}

**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.

**Code:**

package ass4Q3;

import java.util.Scanner;

class BMITracker {

private double weight;

private double height;

private double bmi;

private String classification;

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 getBmi() {

return bmi;

}

public String getClassification() {

return classification;

}

public void calculateBMI() {

bmi = weight / (height \* height);

classifyBMI();

}

private void classifyBMI() {

if (bmi < 18.5) {

classification = "Underweight";

} else if (bmi >= 18.5 && bmi < 24.9) {

classification = "Normal weight";

} else if (bmi >= 25 && bmi < 29.9) {

classification = "Overweight";

} else {

classification = "Obese";

}

}

*@Override*

public String toString() {

return "Weight: " + weight + " kg, Height: " + height + " m, BMI: " + String.*format*("%.2f", bmi) +

", Classification: " + classification;

}

}

class BMITrackerUtil {

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

public BMITracker acceptRecord() {

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

double weight = scanner.nextDouble();

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

double height = scanner.nextDouble();

return new BMITracker(weight, height);

}

public void printRecord(BMITracker tracker) {

tracker.calculateBMI();

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

}

public void menuList() {

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

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

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

}

}

public class Program {

public static void main(String[] args) {

BMITrackerUtil util = new BMITrackerUtil();

BMITracker tracker = null;

try (Scanner sc = new Scanner(System.***in***)) {

while (true) {

util.menuList();

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

int choice = sc.nextInt();

switch (choice) {

case 1:

tracker = util.acceptRecord();

break;

case 2:

if (tracker != null) {

util.printRecord(tracker);

} else {

System.***out***.println("No BMI details entered yet.");

}

break;

case 3:

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

return;

default:

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

}

}

}

}

}

**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.

**Code:**

package ass4Q4;

import java.util.Scanner;

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;

this.discountAmount = calculateDiscountAmount();

this.finalPrice = calculateFinalPrice();

}

public double getOriginalPrice() {

return originalPrice;

}

public void setOriginalPrice(double originalPrice) {

this.originalPrice = originalPrice;

this.discountAmount = calculateDiscountAmount();

this.finalPrice = calculateFinalPrice();

}

public double getDiscountRate() {

return discountRate;

}

public void setDiscountRate(double discountRate) {

this.discountRate = discountRate;

this.discountAmount = calculateDiscountAmount();

this.finalPrice = calculateFinalPrice();

}

public double getDiscountAmount() {

return discountAmount;

}

public double getFinalPrice() {

return finalPrice;

}

private double calculateDiscountAmount() {

return originalPrice \* (discountRate / 100);

}

private double calculateFinalPrice() {

return originalPrice - discountAmount;

}

*@Override*

public String toString() {

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

originalPrice, discountRate, discountAmount, finalPrice);

}

}

class DiscountCalculatorUtil {

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

public DiscountCalculator acceptRecord() {

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

double originalPrice = scanner.nextDouble();

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

double discountRate = scanner.nextDouble();

return new DiscountCalculator(originalPrice, discountRate);

}

public void printRecord(DiscountCalculator calculator) {

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

}

public void menuList() {

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

System.***out***.println("2. Display Discount Details and Calculate Final Price");

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

}

}

public class Program {

public static void main(String[] args) {

DiscountCalculatorUtil util = new DiscountCalculatorUtil();

DiscountCalculator calculator = null;

try (Scanner sc = new Scanner(System.***in***)) {

while (true) {

util.menuList();

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

int choice = sc.nextInt();

switch (choice) {

case 1:

calculator = util.acceptRecord();

break;

case 2:

if (calculator != null) {

util.printRecord(calculator);

} else {

System.***out***.println("No discount details entered yet.");

}

break;

case 3:

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

return;

default:

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

}

}

}

}

}

**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.

**Code:**

package ass4Q5;

import java.util.Scanner;

class TollBoothRevenueManager {

private double carTollRate;

private double truckTollRate;

private double motorcycleTollRate;

private int numberOfCars;

private int numberOfTrucks;

private int numberOfMotorcycles;

private double totalRevenue;

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

this.carTollRate = carTollRate;

this.truckTollRate = truckTollRate;

this.motorcycleTollRate = motorcycleTollRate;

}

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 getNumberOfCars() {

return numberOfCars;

}

public void setNumberOfCars(int numberOfCars) {

this.numberOfCars = numberOfCars;

}

public int getNumberOfTrucks() {

return numberOfTrucks;

}

public void setNumberOfTrucks(int numberOfTrucks) {

this.numberOfTrucks = numberOfTrucks;

}

public int getNumberOfMotorcycles() {

return numberOfMotorcycles;

}

public void setNumberOfMotorcycles(int numberOfMotorcycles) {

this.numberOfMotorcycles = numberOfMotorcycles;

}

public double calculateTotalRevenue() {

totalRevenue = (carTollRate \* numberOfCars) + (truckTollRate \* numberOfTrucks) + (motorcycleTollRate \* numberOfMotorcycles);

return totalRevenue;

}

*@Override*

public String toString() {

return String.*format*("Cars: %d, Trucks: %d, Motorcycles: %d, Total Revenue: ₹%.2f",

numberOfCars, numberOfTrucks, numberOfMotorcycles, totalRevenue);

}

}

class TollBoothRevenueManagerUtil {

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

public TollBoothRevenueManager acceptRecord() {

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

double carTollRate = scanner.nextDouble();

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

double truckTollRate = scanner.nextDouble();

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

double motorcycleTollRate = scanner.nextDouble();

return new TollBoothRevenueManager(carTollRate, truckTollRate, motorcycleTollRate);

}

public void collectVehicleData(TollBoothRevenueManager manager) {

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

int numberOfCars = scanner.nextInt();

manager.setNumberOfCars(numberOfCars);

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

int numberOfTrucks = scanner.nextInt();

manager.setNumberOfTrucks(numberOfTrucks);

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

int numberOfMotorcycles = scanner.nextInt();

manager.setNumberOfMotorcycles(numberOfMotorcycles);

}

public void printRecord(TollBoothRevenueManager manager) {

manager.calculateTotalRevenue();

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

}

public void menuList() {

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

System.***out***.println("2. Enter Vehicle Data and Calculate Revenue");

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

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

}

}

public class Program {

public static void main(String[] args) {

TollBoothRevenueManagerUtil util = new TollBoothRevenueManagerUtil();

TollBoothRevenueManager manager = null;

try (Scanner sc = new Scanner(System.***in***)) {

while (true) {

util.menuList();

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

int choice = sc.nextInt();

switch (choice) {

case 1:

manager = util.acceptRecord();

break;

case 2:

if (manager != null) {

util.collectVehicleData(manager);

} else {

System.***out***.println("Please set the toll rates first.");

}

break;

case 3:

if (manager != null) {

util.printRecord(manager);

} else {

System.***out***.println("No toll data available.");

}

break;

case 4:

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

return;

default:

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

}

}

}

}

}

**Output:**