**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.question1;

import java.util.Scanner;

class LoanAmortizationCalculator {

private double principal;

private double annualInterestRate;

private int loanTerm;

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);

}

public double calculateTotalAmountPaid() {

return calculateMonthlyPayment() \* loanTerm \* 12;

}

*@Override*

public String toString() {

return String.*format*("Loan Amortization Calculator: [Principal=₹%.2f, Annual Interest Rate=%.2f%%, Loan Term=%d years]",

principal, annualInterestRate, loanTerm);

}

}

class LoanAmortizationCalculatorUtil {

public static LoanAmortizationCalculator 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();

return new LoanAmortizationCalculator(principal, annualInterestRate, loanTerm);

}

public static void printRecord(LoanAmortizationCalculator calculator) {

double monthlyPayment = calculator.calculateMonthlyPayment();

double totalAmountPaid = calculator.calculateTotalAmountPaid();

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

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

System.***out***.printf("Total Amount Paid: ₹%.2f%n", totalAmountPaid);

}

public static void menuList() {

System.***out***.println("1. Enter loan details");

System.***out***.println("2. Exit");

System.***out***.print("Choose an option: ");

}

}

public class Program{

public static void main(String[] args) {

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

LoanAmortizationCalculator calculator = null;

boolean running = true;

while (running) {

LoanAmortizationCalculatorUtil.*menuList*();

int choice = sc.nextInt();

switch (choice) {

case 1:

calculator = LoanAmortizationCalculatorUtil.*acceptRecord*();

LoanAmortizationCalculatorUtil.*printRecord*(calculator);

break;

case 2:

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

running = false;

break;

default:

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

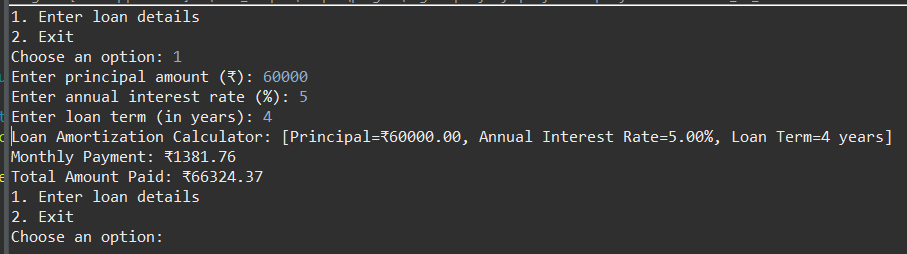
}

sc.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 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.question2;

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 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() {

double ratePerPeriod = annualInterestRate / numberOfCompounds / 100;

double exponent = numberOfCompounds \* years;

return principal \* Math.*pow*(1 + ratePerPeriod, exponent);

}

public double calculateTotalInterest() {

return calculateFutureValue() - principal;

}

*@Override*

public String toString() {

return String.*format*("Principal: ₹%.2f, Annual Interest Rate: %.2f%%, Compounded: %d times/year, Duration: %d years",

principal, annualInterestRate, numberOfCompounds, years);

}

}

class CompoundInterestCalculatorUtil {

public static CompoundInterestCalculator acceptRecord(Scanner scanner) {

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

double principal = scanner.nextDouble();

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

double annualInterestRate = scanner.nextDouble();

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

int numberOfCompounds = scanner.nextInt();

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

int years = scanner.nextInt();

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

}

public static void printRecord(CompoundInterestCalculator calculator) {

double futureValue = calculator.calculateFutureValue();

double totalInterest = calculator.calculateTotalInterest();

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

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

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

}

public static void menuList() {

System.***out***.println("1. Enter investment details");

System.***out***.println("2. Exit");

System.***out***.print("Choose an option: ");

}

}

public class Program {

public static void main(String[] args) {

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

CompoundInterestCalculator calculator = null;

boolean running = true;

while (running) {

CompoundInterestCalculatorUtil.*menuList*();

if (sc.hasNextInt()) {

int choice = sc.nextInt();

switch (choice) {

case 1:

calculator = CompoundInterestCalculatorUtil.*acceptRecord*(sc);

CompoundInterestCalculatorUtil.*printRecord*(calculator);

break;

case 2:

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

running = false;

break;

default:

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

}

} else {

System.***out***.println("Invalid input. Please enter a number.");

sc.next();

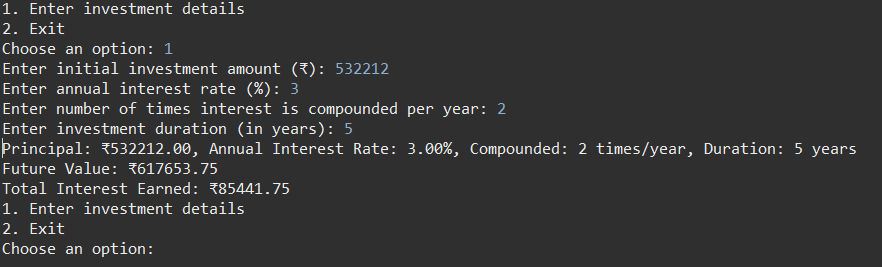
}

}

sc.close();

}

}



**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.question3;

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 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 < 24.9) {

return "Normal weight";

} else if (bmi < 29.9) {

return "Overweight";

} else {

return "Obese";

}

}

public String toString() {

double bmi = calculateBMI();

return String.*format*("BMI: %.2f, Classification: %s", bmi, classifyBMI());

}

}

class BMITrackerUtil {

public static BMITracker acceptRecord(Scanner scanner) {

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 static void printRecord(BMITracker tracker) {

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

}

public static void menuList() {

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

System.***out***.println("2. Exit");

System.***out***.print("Choose an option: ");

}

}

public class Program {

public static void main(String[] args) {

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

BMITracker tracker = null;

boolean running = true;

while (running) {

BMITrackerUtil.*menuList*();

if (sc.hasNextInt()) {

int choice = sc.nextInt();

switch (choice) {

case 1:

tracker = BMITrackerUtil.*acceptRecord*(sc);

BMITrackerUtil.*printRecord*(tracker);

break;

case 2:

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

running = false;

break;

default:

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

}

} else {

System.***out***.println("Invalid input. Please enter a number.");

sc.next();

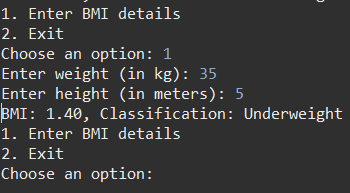
}

}

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 org.question4;

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 getOriginalPrice() {

return originalPrice;

}

public void setOriginalPrice(double originalPrice) {

this.originalPrice = originalPrice;

}

public double getDiscountRate() {

return discountRate;

}

public void setDiscountRate(double discountRate) {

this.discountRate = discountRate;

}

public double calculateDiscountAmount() {

return originalPrice \* (discountRate / 100);

}

public double calculateFinalPrice() {

return originalPrice - calculateDiscountAmount();

}

public String toString() {

double discountAmount = calculateDiscountAmount();

double finalPrice = calculateFinalPrice();

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

originalPrice, discountAmount, finalPrice);

}

}

class DiscountCalculatorUtil {

public static DiscountCalculator acceptRecord(Scanner scanner) {

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

double originalPrice = scanner.nextDouble();

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

double discountRate = scanner.nextDouble();

return new DiscountCalculator(originalPrice, discountRate);

}

public static void printRecord(DiscountCalculator calculator) {

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

}

public static void menuList() {

System.***out***.println("1. Enter discount details");

System.***out***.println("2. Exit");

System.***out***.print("Choose an option: ");

}

}

public class Program {

public static void main(String[] args) {

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

DiscountCalculator calculator = null;

boolean running = true;

while (running) {

DiscountCalculatorUtil.*menuList*();

if (sc.hasNextInt()) {

int choice = sc.nextInt();

switch (choice) {

case 1:

calculator = DiscountCalculatorUtil.*acceptRecord*(sc);

DiscountCalculatorUtil.*printRecord*(calculator);

break;

case 2:

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

running = false;

break;

default:

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

}

} else {

System.***out***.println("Invalid input. Please enter a number.");

sc.next();

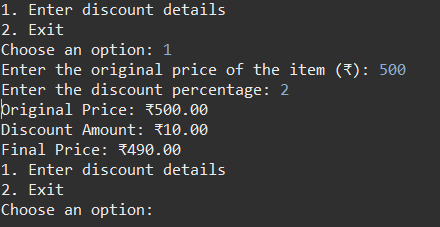
}

}

sc.close();

}

}



**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.question5;

import java.util.InputMismatchException;

import java.util.Scanner;

class TollBoothRevenueManager {

private double carTollRate;

private double truckTollRate;

private double motorcycleTollRate;

private int numCars;

private int numTrucks;

private int numMotorcycles;

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

this.carTollRate = carTollRate;

this.truckTollRate = truckTollRate;

this.motorcycleTollRate = motorcycleTollRate;

this.numCars = 0;

this.numTrucks = 0;

this.numMotorcycles = 0;

}

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 getNumCars() {

return numCars;

}

public void setNumCars(int numCars) {

this.numCars = numCars;

}

public int getNumTrucks() {

return numTrucks;

}

public void setNumTrucks(int numTrucks) {

this.numTrucks = numTrucks;

}

public int getNumMotorcycles() {

return numMotorcycles;

}

public void setNumMotorcycles(int numMotorcycles) {

this.numMotorcycles = numMotorcycles;

}

public double calculateTotalRevenue() {

return (numCars \* carTollRate) + (numTrucks \* truckTollRate) + (numMotorcycles \* motorcycleTollRate);

}

public int calculateTotalVehicles() {

return numCars + numTrucks + numMotorcycles;

}

public String toString() {

return String.*format*("Toll Booth Revenue Manager: [Car Toll Rate=₹%.2f, Truck Toll Rate=₹%.2f, Motorcycle Toll Rate=₹%.2f, " +

"Number of Cars=%d, Number of Trucks=%d, Number of Motorcycles=%d]",

carTollRate, truckTollRate, motorcycleTollRate, numCars, numTrucks, numMotorcycles);

}

}

class TollBoothRevenueManagerUtil {

public static TollBoothRevenueManager acceptRecord() {

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

double carTollRate = 50.00;

double truckTollRate = 100.00;

double motorcycleTollRate = 30.00;

System.***out***.println("Enter toll rates for different vehicle types (default values are set):");

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

carTollRate = sc.nextDouble();

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

truckTollRate = sc.nextDouble();

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

motorcycleTollRate = sc.nextDouble();

TollBoothRevenueManager manager = new TollBoothRevenueManager(carTollRate, truckTollRate, motorcycleTollRate);

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

manager.setNumCars(sc.nextInt());

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

manager.setNumTrucks(sc.nextInt());

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

manager.setNumMotorcycles(sc.nextInt());

return manager;

}

public static void printRecord(TollBoothRevenueManager manager) {

double totalRevenue = manager.calculateTotalRevenue();

int totalVehicles = manager.calculateTotalVehicles();

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

System.***out***.printf("Total Number of Vehicles: %d%n", totalVehicles);

System.***out***.printf("Total Revenue Collected: ₹%.2f%n", totalRevenue);

}

public static void menuList() {

System.***out***.println("1. Enter toll rates and vehicle details");

System.***out***.println("2. Exit");

System.***out***.print("Choose an option: ");

}

}

public class Program {

public static void main(String[] args) {

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

TollBoothRevenueManager manager = null;

boolean running = true;

while (running) {

TollBoothRevenueManagerUtil.*menuList*();

int choice = 0;

try {

choice = sc.nextInt();

} catch (InputMismatchException e) {

System.***out***.println("Invalid choice. Please enter a number.");

sc.next(); // Clear the invalid input

continue;

}

switch (choice) {

case 1:

manager = TollBoothRevenueManagerUtil.*acceptRecord*();

TollBoothRevenueManagerUtil.*printRecord*(manager);

break;

case 2:

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

running = false;

break;

default:

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

}

}

sc.close();

}

}

