Assignment 4

09 September 2024 10:57

1. Loan Amortization Calculator

Implement a system to calculate and display the monthly payments for a mortgage loan. The system should:
Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
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
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 org.example;

import java.util.Scanner;

class Account{
    private int principal;
    private int annualInterestRate;
    private int loanTerm;
    private int monthlyPayment;
    private int monthlyInterestRate;
    private int numberOfMonths;

    void accept_data() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter principle amount          :       ");
        setPrincipal(sc.nextInt());
        System.out.print("Enter anual interest rate  :       ");
        setAnnualInterestRate(sc.nextInt());
        System.out.print("Enter loan term                    :       ");
        setLoanTerm(sc.nextInt());
    }


    public int getPrincipal() {
        return this.principal;
    }

    public void setPrincipal(int principal) {
        this.principal = principal;
    }

    public int getAnnualInterestRate() {
        return this.annualInterestRate;
    }

    public void setAnnualInterestRate(int annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }

    public int getLoanTerm() {
        return this.loanTerm;
```

```
    }

    public void setLoanTerm(int loanTerm) {
        this.loanTerm = loanTerm;
    }

    public int getMonthlyPayment() {
        return this.monthlyPayment;
    }

    public void setMonthlyPayment(int monthlyPayment) {
        this.monthlyPayment = monthlyPayment;
    }


    void calculate_payment() {
        monthlyInterestRate = annualInterestRate / 12 / 100;
        numberOfMonths = loanTerm * 12;
        monthlyPayment = principal * (monthlyInterestRate * (1 + monthlyInterestRate)^(numberOfMonths)) / ((1 +
        monthlyInterestRate)^(numberOfMonths) - 1);

    }

    void print_data() {
        System.out.print("Monthly payment                    :        "+ getMonthlyPayment());
    }
}


public class LoanAmortizationCalculator {

    public static void main(String[] args) {
        Account acc = new Account();

        acc.accept_data();
        acc.calculate_payment();
        acc.print_data();

    }

}
```

Output:

```
Enter principle amount          :       120002
Enter anual interest rate       :       12
Enter loan term                 :       5
Monthly payment                 :       124140
```

2. Compound Interest Calculator for Investment

Develop a system to compute the future value of an investment with compound interest. The system should:
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.

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

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 org.example;

import java.util.Scanner;

class CICalculator{
    int principal;
    int annualInterestRate;
    int numberOfCompounds;
    int years;
    int futureValue;
    int totalInterest;

    void acceptRecord(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Principal       :       ");
        setPincipal(sc.nextInt());
        System.out.print("number Of Compounds    :       ");
        setNumberOfCompunds(sc.nextInt());
        System.out.print("years   :       ");
        setYears(sc.nextInt());
        sc.nextLine();
        System.out.print("annual Interest Rate       :       ");
        setAnnualInterestRate(sc.nextInt());
    }

    public int getPincipal() {
        return this.principal;
    }

    public void setPincipal(int principal) {
        this.principal = principal;
    }

    public int getNumberOfCompunds(){
        return this.numberOfCompounds;
    }

    public void setNumberOfCompunds(int numberOfCompounds) {
        this.numberOfCompounds = numberOfCompounds;
    }

    public int getYears() {
        return this.years;
    }

    public void setYears(int years) {
        this.years=years;
    }

    public int getAnnualInterestRate() {
        return this.annualInterestRate;
    }

    public void setAnnualInterestRate(int annualInterestRate) {
        this.annualInterestRate=annualInterestRate;
    }

    public int getFutureValue() {
        return this.futureValue;
    }
```

```
    public int getTotalInterest() {
        return this.totalInterest;
    }


    void calculateFutureValue(){

        futureValue = principal * (1 + annualInterestRate / numberOfCompounds)^(numberOfCompounds * years);

        totalInterest = futureValue - principal;

    }

    void printRecord(){
        System.out.println("future Value       :       "+ getFutureValue());
        System.out.print("total Interest:       "+ getTotalInterest());
    }

}

public class CompoundInterestCalculator {
    public static void main(String[] args) {
        CICalculator cic = new CICalculator();

        cic.acceptRecord();
        cic.calculateFutureValue();
        cic.printRecord();

    }
}
```

Output:

```
Principal       :       12000
number Of Compounds     :       12
years   :       12
annual Interest Rate    :       12
future Value    :       23888
total Interest  :       11888
```

3. BMI (Body Mass Index) Tracker

Create a system to calculate and classify Body Mass Index (BMI). The system should:
Accept weight (in kilograms) and height (in meters) from the user.

Calculate the BMI using the formula:
BMI Calculation: BMI = weight / (height * height)
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

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 org.example;
import java.util.Scanner;
```

```
class BMITracker{
	private float height;
	private float weight;
	private float BMI;
	private float calculateBMI;
	private float classifyBMI;

	void acceptRecord(){
		Scanner sc = new Scanner(System.in);
		System.out.print("height in meter	:	");
		setHeight(sc.nextFloat());
		System.out.print("weight in kg	:	");
		setWeight(sc.nextFloat());

	}

	public float getHeight() {
		return this.height;
	}

	public void setHeight(float height) {
		this.height=height;
	}

	public float getWeight() {
		return this.weight;
	}

	public void setWeight(float weight) {
		this.weight=weight;
	}

	public float getBMI() {
		return this.BMI;
	}



	void calculateBMI(){
		BMI = weight/(height * height);
		System.out.println("bmi	:	" + BMI);
	}

	void classifyBMI(){
		if(BMI<18.5) {
			System.out.println("Underweight");
		}
		else if(BMI>18.5 && BMI<24.9){
			System.out.println("Normal weight");
		}
		else if(BMI>25 && BMI<29.9){
			System.out.println("Overweight");
		}
		else{
			System.out.println("Obese");
		}
	}

	void printRecord(){
		this.calculateBMI();
		this.classifyBMI();
	}
}

public class BodyMassIndexTracker{
```

```
    public static void main(String[] args){
        BMITracker bmi = new BMITracker();

        bmi.acceptRecord();
        bmi.printRecord();



    }

}
```

Output:

```
height in meter :       2
weight in kg    :       61
bmi     :       15.25
Underweight
```

4. Discount Calculation for Retail Sales

Design a system to calculate the final price of an item after applying a discount. The system should:
Accept the original price of an item and the discount percentage from the user.

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

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 org.example;
import java.util.Scanner;

class DiscountCalculator{
    private double discountAmount;
    private double originalPrice;
    private double discountRate;
    private double finalPrice;


    void acceptRecord(){
        Scanner sc = new Scanner(System.in);
        System.out.print("original Price : ");
        setOriginalPrice(sc.nextDouble());
        System.out.print("discount Rate : ");
        setDiscountRate(sc.nextDouble());
    }



    public double getOriginalPrice() {
        return this.originalPrice;
    }

    public void setOriginalPrice(double originalPrice) {
        this.originalPrice=originalPrice;
    }

    public double getDiscountRate() {
```

```
        return this.discountRate;
    }

    public void setDiscountRate(double discountRate) {
        this.discountRate=discountRate;
    }

    public double getDiscountAmount() {
        return this.discountAmount;
    }

    public double getFinalPrice() {
        return this.finalPrice;
    }


    void calculateDiscount(){
        discountAmount = originalPrice*(discountRate / 100);
        finalPrice = originalPrice-discountAmount;
    }

    void printRecord() {
        System.out.println("discount Amount : " + this.getDiscountAmount());
        System.out.println("final price : " + this.getFinalPrice());
    }

}


public class DiscountCalculation {

    public static void main(String[] args) {
        DiscountCalculator dc = new DiscountCalculator();

        dc.acceptRecord();
        dc.calculateDiscount();
        dc.printRecord();
    }

}
```

Output:

```
original Price : 10000
discount Rate : 12
discount Amount : 1200.0
final price : 8800.0
```

5. Toll Booth Revenue Management

Develop a system to simulate a toll booth for collecting revenue. The system should:
Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.
Accept the number of vehicles of each type passing through the toll booth.
Calculate the total revenue based on the toll rates and number of vehicles.
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 org.example;
import java.util.Scanner;

class TollBoothRevenueManager {
   private int count;
   private float revenue;
   private float toll;
   private String vehicle;

   void acceptRecord() {
      Scanner sc = new Scanner(System.in);
      System.out.print("Vehicle passed is: ");
      vehicle = sc.nextLine();
   }

   public void setVehicle(String vehicle){
         this.vehicle=vehicle;
   }

   public String getVehicle(){
         return this.vehicle;
   }

   public int getCount(){
         return this.count;
   }

   public float getRevenue() {
         return this.revenue;
   }

   void setTollRates() {
      switch(vehicle.toLowerCase()) {
         case "car":
            System.out.println("Car passed the toll");
            toll = 50;
            count += 1;
            break;
         case "truck":
            System.out.println("Truck passed the toll");
            toll = 100;
            count += 1;
            break;
         case "motorcycle":
            System.out.println("Motorcycle passed the toll");
            toll = 30;
            count += 1;
            break;
         default:
            System.out.println("Invalid vehicle type");
            toll = 0;
            break;
      }
   }

   void calculateRevenue() {
      revenue += toll;
   }

   void printRecord() {
      System.out.println("Revenue: " + this.getRevenue());
```

```
        System.out.println("Number of vehicles: " + this.getCount());
        System.out.println();
    }

    static void menulist() {
        System.out.println("Press 0 to Exit");
        System.out.println("Press 1 for Accept Record");
        System.out.println("Press 2 for Print Record");
    }
}

public class TollBoothRevenueManagement {
    public static void main(String[] args) {
        TollBoothRevenueManager tbrm = new TollBoothRevenueManager();
        Scanner sc = new Scanner(System.in);
        int choice = -1;

        while(choice!=0) {

            TollBoothRevenueManager.menulist();
            choice = sc.nextInt();
            switch(choice) {
                    case 1:
                            tbrm.acceptRecord();
                            tbrm.setTollRates();
                            tbrm.calculateRevenue();
                            break;
                    case 2:
                            tbrm.printRecord();
                            break;
                    default:
                            System.out.println("Exit");
                            break;
            }
        }

    }
}
```

Output:

```
Press 0 to Exit
Press 1 for Accept Record
Press 2 for Print Record
1
Vehicle passed is: car
Car passed the toll
Press 0 to Exit
Press 1 for Accept Record
Press 2 for Print Record
1
Vehicle passed is: motorcycle
Motorcycle passed the toll
Press 0 to Exit
Press 1 for Accept Record
Press 2 for Print Record
1
Vehicle passed is: truck
Truck passed the toll
Press 0 to Exit
Press 1 for Accept Record
Press 2 for Print Record
1
Vehicle passed is: abcd
Invalid vehicle type
Press 0 to Exit
Press 1 for Accept Record
Press 2 for Print Record
```

```
Invalid vehicle type
Press 0 to Exit
Press 1 for Accept Record
Press 2 for Print Record
2
Revenue: 180.0
Number of vehicles: 3

Press 0 to Exit
Press 1 for Accept Record
Press 2 for Print Record
0
Exit
```