OOPJ
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 com.example.q1;

import java.util.Scanner;

class LoanAmortizationCalculator {

	private double principal;

	private double annualInterestRate;

	private double loanTerm;

	public LoanAmortizationCalculator() {
		this.principal = 0.0;
		this.annualInterestRate = 0.0;
```

```
		this.loanTerm = 0.0;
	}

	public LoanAmortizationCalculator(double principal, double annualInterestRate, double loanTerm) {
		this.principal = principal;
		this.annualInterestRate = annualInterestRate;
		this.loanTerm = loanTerm;
	}

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

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

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

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

	public double getLoanTerm() {
		return this.loanTerm;
	}

	public void setLoanTerm(double loanTerm) {
		this.loanTerm = loanTerm;
	}
}
class LoanAmortizationCalculatorUtil {

	double monthlyInterestRate;
	double numberOfMonths;
	double monthlyPayment;
	double totalAmount;

	private LoanAmortizationCalculator lac = new LoanAmortizationCalculator();

	public LoanAmortizationCalculator getLac() {
		return lac;
	}

	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {

		System.out.print("Enter Principal: ");
		lac.setPrincipal( sc.nextDouble() );
		System.out.print("Enter Annual Interest Rate: ");
		lac.setAnnualInterestRate( sc.nextDouble() );
		System.out.print("Enter Loan Term: ");
		lac.setLoanTerm( sc.nextDouble() );
	}
```

```
	public double calculateMonthlyPayment() {

		monthlyInterestRate = lac.getAnnualInterestRate() / 12 / 100;
		numberOfMonths = lac.getLoanTerm() * 12;
		monthlyPayment = lac.getPrincipal() * (monthlyInterestRate * Math.pow((1 + monthlyInterestRate),
(numberOfMonths))) / (Math.pow((1 + monthlyInterestRate), (numberOfMonths)) - 1);
		return monthlyPayment;
	}

	public double calculateTotalAmount() {

		totalAmount = calculateMonthlyPayment() * numberOfMonths;
		return totalAmount;
	}

	public void printRecord() {
		System.out.printf("Monthly Payment: %.2f%n", this.calculateMonthlyPayment());
		System.out.printf("Total amount paid: %.2f%n", this.calculateTotalAmount());
	}

	public static int menuList( ) {
		System.out.println("0. Exit");
		System.out.println("1. Accept Record");
		System.out.println("2. Print Record");
		System.out.print("Enter choice: ");
		int choice = sc.nextInt( );
		return choice;
	}
}

public class Program {
	public static void main(String[] args) {

		LoanAmortizationCalculatorUtil util = new LoanAmortizationCalculatorUtil();

		int choice;
		while ( (  choice = LoanAmortizationCalculatorUtil.menuList( ) ) != 0 ) {
			switch( choice ) {
			case 1:
				util.acceptRecord();
				break;
			case 2:
				util.printRecord();
				break;
			default:
				System.out.println("Invalid Choice");
			}

		}
	}
}
```

Output

```
0. Exit
1. Accept Record
2. Print Record
Enter choice: 1
Enter Principal: 1000000
Enter Annual Interest Rate: 10
Enter Loan Term: 10
0. Exit
1. Accept Record
2. Print Record
Enter choice: 2
Monthly Payment: 13215.07
Total amount paid: 1585808.84
```

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 com.example.a4q2;

import java.util.Scanner;

class CompoundInterestCalculator {

	private double principal;
	private double annualInterestRate;
	private double numberOfCompounds;
	private double years;

	public CompoundInterestCalculator() {

		this.principal = 0.0;
```

```
		this.annualInterestRate = 0.0;
		this.numberOfCompounds = 0.0;
		this.years = 0.0;
	}

	public CompoundInterestCalculator(double principal, double annualInterestRate, double
numberOfCompounds, double 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 double getNumberOfCompounds() {
		return numberOfCompounds;
	}

	public void setNumberOfCompounds(double numberOfCompounds) {
		this.numberOfCompounds = numberOfCompounds;
	}

	public double getYears() {
		return years;
	}

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

class CompoundInterestCalculatorUtil {

	private double futureValue;
	private double totalInterest;

	private CompoundInterestCalculator cic = new CompoundInterestCalculator();
	public CompoundInterestCalculator getCic() {
		return cic;
	}

	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {
```

```
                System.out.print("Enter Principal: ");
                this.cic.setPrincipal( sc.nextDouble() );
                System.out.print("Enter Interest Rate: ");
                this.cic.setAnnualInterestRate( sc.nextDouble() );
                System.out.print("Enter number of compounds: ");
                this.cic.setNumberOfCompounds( sc.nextDouble() );
                System.out.print("Enter duration: ");
                this.cic.setYears( sc.nextDouble() );
        }

        public double calculateFutureValue() {
                futureValue = cic.getPrincipal() * Math.pow((1 + cic.getAnnualInterestRate() /
cic.getNumberOfCompounds() / 100), (cic.getNumberOfCompounds() * cic.getYears()));
                return futureValue;
        }

        public double calculateTotalInterest() {
                totalInterest = futureValue - cic.getPrincipal();
                return totalInterest;
        }

        public void printRecord() {
                calculateFutureValue();
                calculateTotalInterest();
                System.out.printf("Future Value: %.2f%n", futureValue);
                System.out.printf("Total Interest: %.2f%n", totalInterest);
        }

        public static int menuList() {
                System.out.println("0. Exit");
                System.out.println("1. Accept Record");
                System.out.println("2. Print Record");
                System.out.print("Enter choice:  ");
                int choice = sc.nextInt( );
                return choice;
        }
}

public class Program {
        public static void main(String[] args) {

                int choice;
                CompoundInterestCalculatorUtil util = new CompoundInterestCalculatorUtil();

                while ( ( choice = CompoundInterestCalculatorUtil.menuList( ) ) != 0 ) {
                        switch( choice ) {
                        case 1:
                                util.acceptRecord();
                                break;
                        case 2:
                                util.printRecord();
                                break;
                        }
                }
        }
}
```

Output

```
0. Exit
1. Accept Record
2. Print Record
Enter choice:   1
Enter Principal: 100000
Enter Interest Rate: 8
Enter number of compounds: 2
Enter duration: 3
0. Exit
1. Accept Record
2. Print Record
Enter choice:   2
Future Value: 126531.90
Total Interest: 26531.90
0. Exit
1. Accept Record
2. Print Record
Enter choice:
```

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 com.example.a4q3;
import java.util.Scanner;

class BMITracker {

	private double weight;
	private double height;

	public BMITracker() {
		this.weight = 0.0;
```

```
		this.height = 0.0;
	}

	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;
	}

	@Override
	public String toString() {
		return "BMITracker [weight=" + weight + ", height=" + height + "]";
	}
}

class BMITrackerUtil {

	private double bmi;

	private BMITracker bmit = new BMITracker();

	public BMITracker getBmit() {
		return bmit;
	}

	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {
		System.out.print("Enter weight (in kg): ");
		this.bmit.setWeight( sc.nextDouble() );
		System.out.print("Enter height (in m): ");
		this.bmit.setHeight( sc.nextDouble() );
	}

	public double calculateBMI() {
		 bmi = bmit.getWeight() / Math.pow(bmit.getHeight(), 2);
		 return bmi;
	}

	public String classifyBMI() {
		if(bmi > 30) {
			return "Obese";
		} else if (bmi > 25) {
```

```
                return "Overweight";
            } else if (bmi > 18.5) {
                return "Normal weight";
            } else {
                return "Under weight";
            }
        }

        public void printRecord() {
            System.out.printf("BMI: %.2f%n", calculateBMI() );
            System.out.printf("Classification: %s%n", classifyBMI());
            System.out.println(bmit.toString());
        }

        public static int menuList() {
            System.out.println("0. Exit");
            System.out.println("1. Accept Record");
            System.out.println("2. Print Record");
            System.out.print("Enter choice: ");
            int choice = sc.nextInt();
            return choice;
        }
}

public class Program {
        public static void main(String[] args) {

            BMITrackerUtil util = new BMITrackerUtil();

            int choice;
            while(( choice = BMITrackerUtil.menuList())  != 0 ) {
                switch( choice ) {
                    case 1:
                        util.acceptRecord();
                        break;
                    case 2:
                        util.printRecord();
                        break;
                    default:
                        System.out.println("Invalid Input");
                }
            }
        }
}
```

Output

```
0. Exit
1. Accept Record
2. Print Record
Enter choice: 1
Enter weight (in kg): 73
Enter height (in m): 1.75
0. Exit
1. Accept Record
2. Print Record
Enter choice: 2
BMI: 23.84
Classification: Normal weight
BMITracker [weight=73.0, height=1.75]
```

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.example.a4q4;

import java.util.Scanner;

class DiscountCalculator{

	private float originalPrice;
	private float discountRate;

	public DiscountCalculator() {
		this.originalPrice = 0.0f;
		this.discountRate = 0.0f;
	}

	public DiscountCalculator(float originalPrice, float discountRate) {
		this.originalPrice = originalPrice;
		this.discountRate = discountRate;
	}

	public float getOriginalPrice() {
		return originalPrice;
	}

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

	public float getDiscountRate() {
		return discountRate;
	}

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

```
	@Override
	public String toString() {
		return "DiscountCalculator [originalPrice=" + originalPrice + ", discountRate=" + discountRate + "]";
	}
}

class DiscountCalculatorUtil {

	private float discountAmount;
	private float finalPrice;

	private DiscountCalculator dc = new DiscountCalculator();

	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {

		System.out.print("Enter Original Price: ");
		this.dc.setOriginalPrice( sc.nextFloat() );
		System.out.print("Enter dicount percent: ");
		this.dc.setDiscountRate( sc.nextFloat() );
	}

	public float calculateDiscount() {
		discountAmount = dc.getOriginalPrice() * (dc.getDiscountRate() / 100);
		return discountAmount;
	}

	public float calculateFinalPrice() {
		finalPrice = dc.getOriginalPrice() - discountAmount;
		return finalPrice;
	}

	public void printRecord() {
		System.out.printf("Discount Amount: %.2f%n", calculateDiscount());
		System.out.printf("Final Amount: %.2f%n", calculateFinalPrice());
		System.out.println(dc.toString());
	}

	public static int menuList() {
		System.out.println("0.Exit.");
		System.out.println("1.Accept Record.");
		System.out.println("2.Print Record.");
		System.out.print("Enter choice: ");
		int choice = sc.nextInt( );
		return choice;
	}
}

public class Program {
	public static void main(String[] args) {

		DiscountCalculatorUtil util = new DiscountCalculatorUtil();

		int choice;
		while ( (  choice = DiscountCalculatorUtil.menuList( ) ) != 0 ) {
			switch( choice ) {
			case 1:
```

```
                    util.acceptRecord();
                    break;
                case 2:
                    util.printRecord();
                    break;
                default:
                    System.out.println("Invalid Input");
                }
            }
        }
}
```

Output

```
0.Exit.
1.Accept Record.
2.Print Record.
Enter choice: 1
Enter Original Price: 1000
Enter dicount percent: 15
0.Exit.
1.Accept Record.
2.Print Record.
Enter choice: 2
Discount Amount: 150.00
Final Amount: 850.00
DiscountCalculator [originalPrice=1000.0, discountRate=15.0]
```

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 com.example.a4q5;

import java.util.Scanner;

class TollBoothRevenueManager {

	private float carTollRate;
	private float truckTollRate;
	private float motorcycleTollRate;
	private int numberOfCar;
	private int numberOfTruck;
	private int numberOfMotorcycle;

	public TollBoothRevenueManager() {
		this.carTollRate = 0.0f;
		this.truckTollRate = 0.0f;
		this.motorcycleTollRate = 0.0f;
		this.numberOfCar = 0;
		this.numberOfTruck = 0;
		this.numberOfMotorcycle = 0;
	}

	public TollBoothRevenueManager(float carTollRate, float truckTollRate, float motorcycleTollRate, int
numberOfCar,
			int numberOfTruck, int numberOfMotorcycle) {
		this.carTollRate = carTollRate;
		this.truckTollRate = truckTollRate;
		this.motorcycleTollRate = motorcycleTollRate;
		this.numberOfCar = numberOfCar;
		this.numberOfTruck = numberOfTruck;
		this.numberOfMotorcycle = numberOfMotorcycle;
	}

	public float getCarTollRate() {
		return carTollRate;
	}

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

	public float getTruckTollRate() {
		return truckTollRate;
	}

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

	public float getMotorcycleTollRate() {
		return motorcycleTollRate;
	}

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

```
	public int getNumberOfCar() {
		return numberOfCar;
	}

	public void setNumberOfCar(int numberOfCar) {
		this.numberOfCar = numberOfCar;
	}

	public int getNumberOfTruck() {
		return numberOfTruck;
	}

	public void setNumberOfTruck(int numberOfTruck) {
		this.numberOfTruck = numberOfTruck;
	}

	public int getNumberOfMotorcycle() {
		return numberOfMotorcycle;
	}

	public void setNumberOfMotorcycle(int numberOfMotorcycle) {
		this.numberOfMotorcycle = numberOfMotorcycle;
	}

	@Override
	public String toString() {
		return "TollBoothRevenueManager [carTollRate=" + carTollRate + ", truckTollRate=" + truckTollRate
				+ ", motorcycleTollRate=" + motorcycleTollRate + ", numberOfCar=" + numberOfCar +
", numberOfTruck="
				+ numberOfTruck + ", numberOfMotorcycle=" + numberOfMotorcycle + "]";
	}
}

class TollBoothRevenueManagerUtil {

	private float totalRevenue;
	private int totalVehicle;

	private TollBoothRevenueManager tbrm = new TollBoothRevenueManager();
	public TollBoothRevenueManager getTbrm() {
		return tbrm;
	}

	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {

		System.out.print("Enter number of Cars: ");
		this.tbrm.setNumberOfCar( sc.nextInt() );
		System.out.print("Enter number of Trucks: ");
		this.tbrm.setNumberOfTruck( sc.nextInt() );
		System.out.print("Enter number of Motorcycles: ");
		this.tbrm.setNumberOfMotorcycle( sc.nextInt() );
		System.out.print("Enter toll rate for Cars: ");
		this.tbrm.setCarTollRate( sc.nextFloat() );
		System.out.print("Enter toll rate for Trucks: ");
		this.tbrm.setTruckTollRate( sc.nextFloat() );
		System.out.print("Enter toll rate for Motorcycles: ");
```

```
                this.tbrm.setMotorcycleTollRate( sc.nextFloat() );
        }

        public float calculateRevenue() {
                totalRevenue = (tbrm.getNumberOfCar() * tbrm.getCarTollRate()) + (tbrm.getNumberOfTruck() *
tbrm.getTruckTollRate()) + (tbrm.getNumberOfMotorcycle() * tbrm.getMotorcycleTollRate());
                return totalRevenue;
        }

        public int calculateTotalVehicle() {
                totalVehicle = tbrm.getNumberOfCar() + tbrm.getNumberOfTruck() + tbrm.getNumberOfMotorcycle();
                return totalVehicle;
        }

        public void printRecord() {
                System.out.println("Total Vehicle: " + calculateTotalVehicle());
                System.out.printf("Total Revenue: %.2f%n", calculateRevenue());
                System.out.println(tbrm.toString());
        }

        public static int menuList( ) {
                System.out.println("0. Exit");
                System.out.println("1. Accept Record");
                System.out.println("2. Print Record");
                System.out.print("Enter choice:  ");
                int choice = sc.nextInt( );
                return choice;
        }
}

public class Program {
        public static void main(String[] args) {

                TollBoothRevenueManagerUtil util = new TollBoothRevenueManagerUtil();

                int choice;
                while ( (  choice = TollBoothRevenueManagerUtil.menuList( ) ) != 0 ) {
                        switch( choice ) {
                        case 1:
                                util.acceptRecord();
                                break;
                        case 2:
                                util.printRecord();
                                break;
                        default:
                                System.out.println("Invalid Input");
                        }
                }
        }
}
```

Output

```
0. Exit
1. Accept Record
2. Print Record
Enter choice:   1
Enter number of Cars: 300
Enter number of Trucks: 800
Enter number of Motorcycles: 500
Enter toll rate for Cars: 50
Enter toll rate for Trucks: 100
Enter toll rate for Motorcycles: 30
0. Exit
1. Accept Record
2. Print Record
Enter choice:   2
Total Vehicle: 1600
Total Revenue: 110000.00
```