ASSIGNMENT NO.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 assignment_5;

import java.util.Scanner;

class Cal{

    private float rate;

    private float term;

    private double monthlyPayment;

    private double principal;
private double totalAmountPaid;

public double getPrincipal() {

        return principal;

}

public void setPrincipal(double principal) {

        this.principal = principal;

}


public float getRate() {

        return rate;

}

public void setRate(float rate) {

        this.rate = rate;

}

public float getTerm() {

        return term;

}

public void setTerm(float term) {

        this.term = term;

}

public double getMonthlyPayment() {

        return monthlyPayment;

}

public double getTotalAmountPaid() {

        return totalAmountPaid;
        }

        public void cal() {

                monthlyPayment = principal *

                                ((rate/1200) *

                                Math.pow((1 + (rate/1200)),((float)term)*12))

                                / (Math.pow((1 + (rate/1200)),((float)term)*12) - 1);

                totalAmountPaid = monthlyPayment * term * 12;

        }


}

class CalUtill{

        Cal c=new Cal();

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

        public void acceptRecord() {

                c.setPrincipal(sc.nextFloat());

                c.setRate(sc.nextFloat());

                c.setTerm(sc.nextFloat());

        }

        public void printRecord() {

                c.cal();

                System.out.println(c.getMonthlyPayment()+"
"+c.getTotalAmountPaid());

        }

        public int choice() {

                System.out.println("enter 1 for setting values");

                System.out.println("enter 2 for printing values");
                System.out.println("enter 0 for to exit ");

                int choice =sc.nextInt();

                return choice;


        }

}

public class ass5 {


        public static void main(String[] args) {

                CalUtill c1=new CalUtill();

                int choice ;

                while((choice = c1.choice())!=0) {

                        switch(choice) {

                        case 1:c1.acceptRecord();

                        break;

                        case 2:c1.printRecord();

                        break;

                        }

                }

                CalUtill.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 ass5;


import compoundUtill.CompoundUtill;


public class Program {


    public static void main(String[] args) {

        // TODO Auto-generated method stub

    CompoundUtill c1=new CompoundUtill();

    int choice ;

    while((choice = CompoundUtill.menuList())!=0) {

      switch(choice) {

      case 1:c1.acceptRecord();

      break;
```

```
            case 2:c1.printRecord();
            break;
            }
          }
          CompoundUtill.sc.close();
            }

    }
    public int getNumberOfCompounds() {
            return numberOfCompounds;
    }

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

    public int getYears() {
            return years;
    }
    public double getTotalInterest() {
            return totalInterest;
    }
    public void setTotalInterest(double totalInterest) {
            this.totalInterest = totalInterest;
```

```
}

public void setYears(int years) {

        this.years = years;

}

public String toString() {

        return this.principal+" "+this.annualInterestRate+"
"+this.numberOfCompounds;

}

public double getFutureValue() {

        return futureValue;

}

public void setFutureValue(double futureValue) {

        this.futureValue = futureValue;

}

public void cal() {

        futureValue = principal * (1 + annualInterestRate
/Math.pow(numberOfCompounds, numberOfCompounds*years));

        totalInterest = futureValue - principal;

}



}

package compoundUtill;



import java.util.Scanner;
import demo.Compound;


public class CompoundUtill {

private Compound c=new Compound();

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

public void printRecord() {

        c.cal();

        System.out.println(c.getTotalInterest()+" "+c.getFutureValue());

}

public void acceptRecord() {

        c.setPrincipal(sc.nextFloat());

        c.setAnnualInterestRate(sc.nextDouble());

        c.setNumberOfCompounds(sc.nextInt());

        c.setYears(sc.nextInt());

}

public static int menuList() {

        System.out.println("enter 0 to exit");

        System.out.println("enter 1 to acceptRecord");

        System.out.println("enter 2 to printRecord");

        return sc.nextInt();

}

}
```

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 ass5;

import compoundUtill.BmiUtill;
public class Program {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	//CompoundUtill c1=new CompoundUtill();
		BmiUtill b=new BmiUtill();
	int choice ;
	while((choice = BmiUtill.menuList())!=0) {
	  switch(choice) {
	  case 1:b.acceptRecord();
	  break;
	  case 2:b.printRecord();
	  break;
	  }
	}
	BmiUtill.sc.close();
	  }

}

package demo;

public class Bmi {
private double bmi;
private int height;
private int weight;
public double getBmi() {
        return bmi;
}
public void setBmi(double bmi) {
        this.bmi = bmi;
}
public int getHeight() {
        return height;
}
public void setHeight(int height) {
        this.height = height;
}
public int getWeight() {
        return weight;
}
public void setWeight(int weight) {
        this.weight = weight;
}
public void cal() {
        bmi=(weight*10000)/Math.pow(height, 2);
        if(bmi<20)
                System.out.println("underweight");
        else if(bmi>=20||bmi<=25)
                System.out.println("noraml");
        else if(bmi>=26||bmi<=29)
                System.out.println("overweight");
        else if(bmi>30)
                System.out.println("obese");
}
public String toString() {
        return this.bmi+" "+this.height+" "+this.weight;


}

}

package compoundUtill;

import java.util.Scanner;

import demo.Bmi;

public class BmiUtill {
private Bmi b=new Bmi();
public static Scanner sc=new Scanner(System.in);
public void acceptRecord() {
        b.setHeight(sc.nextInt());
        b.setWeight(sc.nextInt());
}
public void printRecord() {
        b.cal();

}
public static  int menuList() {
        System.out.println("enter 0 to exit");
        System.out.println("enter 1 to accpetRecord");
        System.out.println("enter 2 to printRecord");
        return sc.nextInt();
}
}
```

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 ass5;

//import compoundUtill.BmiUtill;
import compoundUtill.DiscountUtill;
public class Program {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
     //CompoundUtill c1=new CompoundUtill();
                //BmiUtill b=new BmiUtill();
                DiscountUtill d=new DiscountUtill();
     int choice ;
     while((choice = DiscountUtill.menuList())!=0) {
            switch(choice) {
            case 1:d.acceptRecord();
            break;
            case 2:d.printRecord();
            break;
            }
        }
        DiscountUtill.sc.close();
            }

}
package demo;

public class Dsicount {
private double discountAmount;
private double originalPrice;
private double discountRate;
private double finalPrice;

public double getDiscountAmount() {
            return discountAmount;
}

public void setDiscountAmount(double discountAmount) {
            this.discountAmount = discountAmount;
}

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 getFinalPrice() {
            return finalPrice;
}

public void setFinalPrice(double finalPrice) {
            this.finalPrice = finalPrice;
}
```

```
public void cal() {
        discountAmount = originalPrice * (discountRate / 100);
        finalPrice = originalPrice - discountAmount;

}
public String toString() {
        return this.discountAmount+" "+this.discountRate+" "+this.finalPrice;
}
}

package compoundUtill;

import java.util.Scanner;

import demo.Dsicount;

public class DiscountUtill {
private Dsicount d=new Dsicount();
public static Scanner sc=new Scanner(System.in);
public void acceptRecord() {
        d.setOriginalPrice(sc.nextDouble());;
        d.setDiscountRate(sc.nextDouble());;

}
public void printRecord() {
        d.cal();
        System.out.println(d.getDiscountAmount()+" "+d.getFinalPrice());
}
public static  int menuList(){
        System.out.println("enter o to exit");
        System.out.println("enter 1 to acceptRecord");
        System.out.println("enter 2 to printRecord");
        return sc.nextInt();
}
}
```

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.