МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут прикладної математики і фундаментальних наук

Кафедра прикладної математики

Лабораторна робота №6
з курсу "Програмування настільних застосунків"
Тема: "Інтерфейси та абстрактні класи."

Виконав студент групи ПМ-33
Венгринюк Олег
Прийняла
Терендій О. В.

Завдання для лабораторної роботи:

1. Розробити, відповідно до варіанту, два інтерфейси, зв'язаних наслідуванням, базовий абстрактний клас та конкретні класи, розмістивши кожен з них в окремому пакеті. Набір полів і методів, необхідних для забезпечення функціональної зручності класів, визначити самостійно.
2. Створити кілька об'єктів кожного з класів, оголосивши їх:
   а) екземплярами інтерфейсу;
   б) екземплярами абстрактного класу.
3. Продемонструвати можливості класів.

Класи, що потрібно реалізувати:
- базовий клас ДЕПОЗИТНИЙ РАХУНОК
- похідний клас РАХУНОК З ВІДСОТКАМИ В КІНЦІ ТЕРМІНУ
- похідний клас КРАХУНОК З ПРОГРЕСИВНОЮ СТАВКОЮ

Для введених даних про відкриті депозитні рахунки відсортувати їх за зростанням прибутку (суми отриманих відсотків) і визначити найвигідніші.

Програмний код:

Main.java

```
import deposit.Deposit;
import deposit.DepositInterfaceAdvanced;
import depositStatic.DepositStatic;
import depositDynamic.DepositDynamic;
import rate.RateYear;
import rate.RateQuarter;
import java.util.*;
public class Main {
    public static void main(String[] args) {
        double rateYearConst = 0.4;
        double[] ratesQuarters = {0.2, 0.3, 0.4, 0.5};
        RateYear rateYear = new RateYear(rateYearConst);
        RateQuarter rateQuarter = new RateQuarter(ratesQuarters);
        Deposit depositStatic = new DepositStatic("Deposit static", 1000, rateYear, 12);
        DepositInterfaceAdvanced depositDynamic = new DepositDynamic("Deposit dynamic",
1000, rateQuarter, 4);
        testDeposit(depositStatic);
        testDeposit(depositDynamic);
        DepositInterfaceAdvanced[] deposits = {depositStatic, depositDynamic};
        outputDeposits(deposits);
    }
    static void testDeposit(DepositInterfaceAdvanced deposit) {
        System.out.println("Client's name: " + deposit.getName());
        System.out.println("Deposit sum: " + deposit.getSum());
        System.out.println("Deposit rate : " + deposit.getRate());
        System.out.println("Deposit profit is: " + deposit.calcProfit());
        System.out.println();
    }
    static void outputDeposits(DepositInterfaceAdvanced[] deposits) {
        HashMap<String, Double> depositMap = new HashMap();
        for (DepositInterfaceAdvanced deposit : deposits) {
            depositMap.put(deposit.getName(), 0.);
        }
        for (DepositInterfaceAdvanced deposit : deposits) {
            double profit = depositMap.get(deposit.getName());
            profit += deposit.calcProfit();
            depositMap.put(deposit.getName(), profit);
        }
        HashMap sorted = sortHashMapByValues(depositMap);
        System.out.println("Profit: " + sorted);
```

```
        System.out.println("Profit sorted");
        for (Object key : sorted.keySet()) {
            System.out.println(key + " " +sorted.get(key));
        }
    }
    static public LinkedHashMap<String, Double> sortHashMapByValues(
            HashMap<String, Double> passedMap) {
        List<String> mapKeys = new ArrayList<>(passedMap.keySet());
        List<Double> mapValues = new ArrayList<>(passedMap.values());
        Collections.sort(mapValues, Collections.reverseOrder());
        Collections.sort(mapKeys, Collections.reverseOrder());
        LinkedHashMap<String, Double> sortedMap =
                new LinkedHashMap<>();
        Iterator<Double> valueIt = mapValues.iterator();
        while (valueIt.hasNext()) {
            Double val = valueIt.next();
            Iterator<String> keyIt = mapKeys.iterator();
            while (keyIt.hasNext()) {
                String key = keyIt.next();
                Double comp1 = passedMap.get(key);
                Double comp2 = val;
                if (comp1.equals(comp2)) {
                    keyIt.remove();
                    sortedMap.put(key, val);
                    break;
                }
            }
        }
        return sortedMap;
    }
}
```

deposit.Deposit

```
 package deposit;
import rate.Rate;
public abstract class Deposit implements DepositInterfaceAdvanced{
    protected String name;
    protected double sum;
    protected Rate rate;
    protected double term;
    protected Deposit(String name, double sum, Rate rate, double term){
        this.name = name;
        this.sum = sum;
        this.rate = rate;
        this.term = term;
    }
    public String getName(){
        return name;
    }
    public double getSum(){
        return sum;
    }
    public Rate getRate(){
        return rate;
    }
    public double getTerm(){
        return term;
    }
    public void setName(String name){
        this.name = name;
    }
    public void setSum(double sum){
        this.sum = sum;
    }
    public void setRate(Rate rate){
        this.rate = rate;
    }
```

```
    public void setTerm(double term){
        this.term = term;
    }
}
```

deposit.DepositInterface

```
package deposit;
import rate.Rate;
public interface DepositInterface {
    public String getName();
    public double getSum();
    public Rate getRate();
    public double getTerm();
    public void setName(String name);
    public void setSum(double sum);
    public void setRate(Rate rate);
    public void setTerm(double term);
}
```

deposit.DepositInterfaceAdvanced

```
package deposit;
public interface DepositInterfaceAdvanced extends DepositInterface {
    public double calcProfit();
}
```

depositDynamic.DepositDynamic

```
package depositDynamic;
import deposit.Deposit;
import rate.RateQuarter;
public class DepositDynamic extends Deposit{
    private int termQuarters;
    public DepositDynamic(String name, double sum, RateQuarter rate, int termQuarters){
        super(name, sum, rate, termQuarters*3);
        this.termQuarters = termQuarters;
    }
    public double calcProfit(){
        RateQuarter rate = (RateQuarter) this.rate;
        double total = sum;
        for(int i=0; i<termQuarters; ++i){
            total += (1+rate.getRate(i))*total;
        }
        return total;
    }
}
```

depositStatic.DepositStatic

```
package depositStatic;
import rate.Rate;
import deposit.Deposit;
public class DepositStatic extends Deposit{
    private double monthRate;
    public DepositStatic(String name, double sum, Rate rate, double term){
        super(name, sum, rate, term);
        monthRate = rate.getRate()/12;
    }
    public double calcProfit(){
        return (1+monthRate*term)*sum;
    }
}
```

rate.Rate

```
package rate;
public abstract class Rate{
```

```
    double rate;
    public Rate(){
        rate = 0;
    }
    public Rate(double rate){
        this.rate = rate;
    }
    public double getRate(){
        return rate;
    }
}
```

rate.RateQuater

```
package rate;
public class RateQuarter extends Rate {
    private double[] rate;
    public RateQuarter(double[] rate){
        this.rate = rate;
    }
    public double getRate(int quater){
        return this.rate[quater];
    }
    public String toString(){
        return String.format("1: %.2f, 2: %.2f, 3: %.2f, 4: %.2f", rate[0], rate[1],
rate[2], rate[3]);
    }
}
```

rate.RateYear

```
package rate;
public class RateYear extends Rate {
    public RateYear(double rate){
        super(rate);
    }
    public String toString(){
        return String.format("%.2f", rate);
    }
}
```

Результат виконання роботи:

```
Client's name: Deposit static
Deposit sum: 1000.0
Deposit rate : 0.40
Deposit profit is: 1400.0

Client's name: Deposit dynamic
Deposit sum: 1000.0
Deposit rate : 1: 0.20, 2: 0.30, 3: 0.40, 4: 0.50
Deposit profit is: 30360.0

Profit: {Deposit dynamic=30360.0, Deposit static=1400.0}
Profit sorted
Deposit dynamic 30360.0
Deposit static 1400.0
```

Висновок: в ході виконання роботи було освоєно абстрактні класи та закріплено знання інтерфейсів. Було виконано поставлення завдання.