МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут прикладної математики і фундаментальних наук

Кафедра прикладної математики

Лабораторна робота №5
з курсу "Програмування настільних застосунків"
Тема: "Наслідування. Створення та використання ієрархії класів"

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

Завдання для лабораторної роботи:

1. Розробити ієорархію класів відповідно до варіанту.
2. Створити базовий, похідні класи.
3. Використати public, protected наслідування.
4. Виконати перевантаження функцій в базовому класі, перевизначити їх в похідних.
5. Включити в звіт Uml-діаграму розробленої ієрархії класів.
6. Продемонструвати можливості класів.

Класи, що потрібно реалізувати:

- Базовий кредит
- Кредит, при якому сума ділиться рівними платежами;
- Кредит, при якому нараховується відсоток від суми залишку;
- Пільговий кредит, при якому держава компенсовує частину відсотків по кредиту.

Класи повинні мати повний набір методів для роботи з ними. Кожен клас обов'язково повинен вміти обчислити суму платежу в заданий місяць, суму виплачену до заданого місяця, суму, яка буде виплачена за весь період.

Програмний код:

Main.java

```
import Credit.Credit;
import Credit.CreditPercentage;
import Credit.CreditSpecial;
import Credit.CreditUniform;

import java.util.Calendar;
import java.util.Date;


public class Main {
   public static void main(String [] args){
      testCredit(10000, "11-2019" , "10-2020");
      testCreditPercentage(10000, "12-2019" , "12-2020");
      testCreditUniform(10000, "11-2019" , "11-2020");
      testCreditSpecial(10000, "11-2019" , "11-2020");
   }

   static private void testCredit(int sum, String dateTake, String  dateGive){

      Credit credit = new Credit(sum, dateTake, dateGive);

      int paymentMonth = 6;
      int totalPaymentMonth = 8;
      int numberPayMonth = 12;

      System.out.print("Sum: ");
      System.out.println(sum);

      System.out.print("Interest rate: ");
      System.out.println(credit.getInterestRate());

      System.out.print("Total sum with interest rate: ");
      System.out.println(credit.getTotal());
```

```
        System.out.print("Date when credit taken: ");
        System.out.println(dateTake);
        System.out.print("Date when credit given: ");
        System.out.println(dateGive);

        System.out.print("Payment history: ");
        System.out.println(credit.getPaymentHistory());
        System.out.print("Payment Schedule: ");
        System.out.println(credit.getPaymentSchedule());

        double moneyToPay = credit.getPaymentSchedule(dateTake);
        System.out.println(String.format("You should pay %f money in %d month: ",moneyToPay,
paymentMonth));

        credit.pay(dateTake);

        System.out.print(String.format("Total paid unit %d month: ", totalPaymentMonth));
        System.out.println(credit.getTotalPaid(dateGive));
        System.out.print("Total sum with interest rate: ");
        System.out.println(credit.getTotal());
        System.out.print("Ovepay: ");
        System.out.println(credit.getOverpay());
    }

    static private void testCreditUniform(int sum, String dateTake, String  dateGive){

        CreditUniform credit = new CreditUniform(sum, dateTake, dateGive);

        int paymentMonth = 6;
        int totalPaymentMonth = 8;
        int numberPayMonth = 12;

        System.out.print("Sum: ");
        System.out.println(sum);

        System.out.print("Interest rate: ");
        System.out.println(credit.getInterestRate());

        System.out.print("Total sum with interest rate: ");
        System.out.println(credit.getTotal());

        System.out.print("Date when credit taken: ");
        System.out.println(dateTake);
        System.out.print("Date when credit given: ");
        System.out.println(dateGive);

        credit.pay("10-2019");
        credit.pay("11-2019");
        credit.pay("12-2019");


        System.out.print("Payment history: ");
        System.out.println(credit.getPaymentHistory());
        System.out.print("Payment Schedule: ");
        System.out.println(credit.getPaymentSchedule());
```

```
        double moneyToPay = credit.getPaymentSchedule(dateTake);
        System.out.println(String.format("You should pay %f money in %d month: ",moneyToPay,
paymentMonth));

        credit.pay(dateTake);

        System.out.print(String.format("Total paid unit %d month: ", totalPaymentMonth));
        System.out.println(credit.getTotalPaid(dateGive));
    }

    static private void testCreditPercentage(int sum, String dateTake, String  dateGive){

    CreditPercentage credit = new CreditPercentage(sum, dateTake, dateGive);

    int paymentMonth = 6;
    int totalPaymentMonth = 8;
    int numberPayMonth = 12;

    System.out.print("Sum: ");
    System.out.println(sum);

    System.out.print("Interest rate: ");
    System.out.println(credit.getInterestRate());

    System.out.print("Total sum with interest rate: ");
    System.out.println(credit.getTotal());

    System.out.print("Date when credit taken: ");
    System.out.println(dateTake);
    System.out.print("Date when credit given: ");
    System.out.println(dateGive);

    double payPart = 1000;

    credit.pay("10-2019", payPart);
    credit.pay("11-2019", payPart);
    credit.pay("12-2019", payPart);
    credit.pay("01-2020", payPart);
    credit.pay("02-2020", payPart);
    credit.pay("03-2020", payPart);
    credit.pay("04-2020", payPart);
    credit.pay("05-2020", payPart);
    credit.pay("06-2020", payPart);
    credit.pay("07-2020", payPart);
    credit.pay("08-2020", payPart);
    credit.pay("09-2020", payPart);
    credit.pay("10-2020", payPart);

    System.out.print("Payment history: ");
    System.out.println(credit.getPaymentHistory());
    System.out.print("Payment Schedule: ");
    System.out.println(credit.getPaymentSchedule());

    double moneyToPay = credit.getPaymentSchedule(dateTake);
    System.out.println(String.format("You should pay %f money in %d month: ",moneyToPay,
```

```
paymentMonth));

    credit.pay(dateTake);

    System.out.print(String.format("Total paid unit %d month: ", totalPaymentMonth));
    System.out.println(credit.getTotalPaid(dateGive));
    System.out.print("Total sum with interest rate: ");
    System.out.println(credit.getTotal());
    System.out.print("Overpay: ");
    System.out.println(credit.getOverpay());
}

    static private void testCreditSpecial(int sum, String dateTake, String  dateGive){

        CreditSpecial credit = new CreditSpecial(sum, dateTake, dateGive);

        int paymentMonth = 6;
        int totalPaymentMonth = 8;
        int numberPayMonth = 12;

        System.out.print("Sum: ");
        System.out.println(sum);

        System.out.print("Interest rate: ");
        System.out.println(credit.getInterestRate());

        System.out.print("Interest goverment dotation rate: ");
        System.out.println(credit.getGovermentDotationRate());

        System.out.print("Total sum with interest rate: ");
        System.out.println(credit.getTotal());

        System.out.print("Date when credit taken: ");
        System.out.println(dateTake);
        System.out.print("Date when credit given: ");
        System.out.println(dateGive);

        credit.pay("10-2019");

        System.out.print("Payment history: ");
        System.out.println(credit.getPaymentHistory());
        System.out.print("Payment Schedule: ");
        System.out.println(credit.getPaymentSchedule());

        double moneyToPay = credit.getPaymentSchedule(dateTake);

        System.out.println(String.format("You should pay %f money in %d month: ",moneyToPay,
paymentMonth));

        credit.pay(dateTake);

        System.out.print(String.format("Total paid unit %d month: ", totalPaymentMonth));
        System.out.println(credit.getTotalPaid(dateGive));
        System.out.print("Total sum with interest rate: ");
        System.out.println(credit.getTotal());
    }
```

```
  }
```

Credit.java

```
package Credit;

import java.util.HashMap;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;

public class Credit {

    protected static int minSum = 10000;
    protected static int minDurationMonth = 6;
    protected static double interestRate = 0.2;
    protected static double  interestRateMonth = interestRate/12;
    static double firstPaymentPart = 0.3;

    double body, total;
    int durationMonth;
    Date dateTake, dateGive;
    HashMap <String, Double> paymentHistory, paymentSchedule;

    Calendar c = Calendar.getInstance();
    String dateFormat = "MM-yyyy";
    SimpleDateFormat sdf = new SimpleDateFormat(dateFormat);

    public Credit(double sum){
        checkSum(sum);
        Date dateTake = new Date();
        Date dateGive = addDate(dateTake, minDurationMonth);

        setBody(sum);
        setDurationMonth(minDurationMonth);
        initTotal();

        setDateTake(dateTake);
        setDateGive(dateGive);
        initPaymentSchedule(dateTake, this.durationMonth);
        initPaymentHistory();
    }

    public Credit(double sum, int duration){
        checkSum(sum);
        checkDuration(duration);

        Date dateTake = new Date();
        Date dateGive = addDate(dateTake, duration);

        setBody(sum);
        setDurationMonth(durationMonth);
        initTotal();

        setDateTake(dateTake);
        setDateGive(dateGive);
```

```
        initPaymentSchedule(dateTake, this.durationMonth);
        initPaymentHistory();
    }

    public Credit(double sum, String dateTakeS, String dateGiveS){
        checkSum(sum);
        checkDateS(dateTakeS);
        checkDateS(dateGiveS);

        dateTake = keyToDate(dateTakeS);
        dateGive = keyToDate(dateGiveS);
        durationMonth = getDurationMonth(dateTake, dateGive);

        setBody(sum);
        setDurationMonth(durationMonth);
        initTotal();

        setDateTake(dateTake);
        setDateGive(dateGive);
        initPaymentSchedule(dateTake, durationMonth);
        initPaymentHistory();
    }

    public Credit(Credit anotherCredit){

        this.body = anotherCredit.getBody();
        this.total = anotherCredit.getTotal();
        this.dateTake = keyToDate(anotherCredit.getDateTake());
        this.dateGive = keyToDate(anotherCredit.getDateGive());
        this.durationMonth = anotherCredit.getDuration();
        this.paymentSchedule = new HashMap<String, Double> (anotherCredit.getPaymentSchedule());
        this.paymentHistory = new HashMap<String, Double> (anotherCredit.getPaymentHistory());
    }

    // ---------- Interface

    public double getBody(){
        return body;
    }

    public double getTotal(){
        return total;
    }

    public int getDuration() {
        return durationMonth;
    }

    public String getDateTake() {
        return dateToKey(dateTake);
    }

    public String getDateGive(){
        return dateToKey(dateGive);
    }
```

```
public HashMap getPaymentHistory() {
   return paymentHistory;
}

public HashMap getPaymentSchedule(){
   return paymentSchedule;
}

public double getPaymentSchedule(String dateS){
   checkDateS(dateS);

   return paymentSchedule.get(dateS);
}

public double getTotalPaid(String dateS){
   checkDateS(dateS);

   Date startDate = dateTake;
   Date endDate = keyToDate(dateS);
   c.setTime(startDate);
   double totalSum = 0;

   while(getDurationMonth(c.getTime(), endDate) != 0){
      totalSum += getPaymentHistory(dateToKey(c.getTime()));
   }

   return totalSum;
}

public double getOverpay(){
   return total - body;
}

public double getPaymentHistory(String dateS){
   checkDateS(dateS);
   double record = paymentHistory.containsKey(dateS)? paymentHistory.get(dateS): 0;
   return record;
}

public void pay(String dateS){
   checkDateS(dateS);

   paymentHistory.put(dateS, paymentSchedule.get(dateS));
   paymentSchedule.put(dateS, 0.);
   body -= paymentSchedule.get(dateS);
}

public double getInterestRate() { return interestRate; }

public double getInterestRateMonth() { return interestRateMonth; }

// ---------- Setters

void setBody(double sum){ body = sum; }

void initTotal(){
```

```
        total = (1+interestRate/12*durationMonth)*body;
    }

    void setDurationMonth(int durationMonth) { this.durationMonth = durationMonth;}

    void setDateTake(Date d){ dateTake = d; }

    void setDateGive(Date d){ dateGive = d; }

    void initPaymentSchedule(Date dateTake, int duration){
        paymentSchedule = new HashMap();

        double firstPayment = total*firstPaymentPart;
        double otherPayment = total*(1-firstPaymentPart)/durationMonth;
        c.setTime(dateTake);
        paymentSchedule.put(dateToKey(c.getTime()), firstPayment);
        for(int i=1; i<durationMonth; ++i){
            c.setTime(dateTake);
            c.add(Calendar.MONTH, i);
            paymentSchedule.put(dateToKey(c.getTime()), otherPayment);
        }
    }

    void initPaymentHistory(){
        paymentHistory = new HashMap();
    }

    // ---------- Helpers

    void checkSum(double sum){

    }

    void checkDateS(String dateS){

    }

    void checkDuration(int duration){

    }

    String dateToKey(Date d){
        String key = sdf.format(d);
        return key;
    }

    Date keyToDate(String key){
        try {
            return sdf.parse(key);
        } catch (java.text.ParseException pe) {
            throw new IllegalArgumentException(String.format("Incorrect date format, expect %s",
dateFormat));
        }
    }

    int getDurationMonth(Date a, Date b){
```

```
        c.setTime(a);
        int y1 = c.get(Calendar.YEAR), m1 = c.get(Calendar.MONTH);
        c.setTime(b);
        int y2 = c.get(Calendar.YEAR), m2 = c.get(Calendar.MONTH);
        int duration  = (y2 - y1)*12 + m2 - m1;

        return duration;
    }

    Date addDate(Date d, int months){
        c.setTime(d);
        c.add(Calendar.MONTH, months);

        return new Date();
    }

    Date addDate(Date d1, Date d2){

        return new Date();
    }
}
```

CreditPercentage.java

```
package Credit;

import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;

import Credit.Credit;

public class CreditPercentage extends Credit {
    private double startBody;

    public CreditPercentage(double sum, String dateTakeS, String dateGiveS) {
        super(sum, dateTakeS, dateGiveS);
        setStartBody(sum);
        initPaymentSchedule(keyToDate(getDateTake()), getDuration());
    }

    void initPaymentSchedule(Date dateTake, int duration) {
        paymentSchedule = new HashMap<String, Double>();

        for (int i = 0; i < durationMonth; ++i) {
            c.setTime(dateTake);
            c.add(Calendar.MONTH, i);
            paymentSchedule.put(dateToKey(c.getTime()), 0.);
        }

        paymentSchedule.put("percents", 0.);
    }

    public void pay(String dateS, double sum){
        checkDateS(dateS);
        checkSum(sum);

        String dateGiveS = dateToKey(dateGive);

        paymentHistory.put(dateS, sum);
```

```
        body -= sum;

        double percents = paymentSchedule.get("percents");
        percents += body*interestRateMonth;
        paymentSchedule.put("percents", percents);

        if (dateGiveS.equals(dateS)){
            System.out.print("Overpay: ");
            System.out.println(paymentSchedule.get("percentage"));
        }

    }

    public double getTotal(){
        return getPaymentSchedule("percents") + startBody;
    }

    private void setStartBody(double sum){ startBody = sum; }

    public double getPaymentSchedule(String dateS){
        checkDateS(dateS);

        System.out.print(" ");

        return paymentSchedule.get(dateS);
    }

    public double getOverpay(){
        return paymentSchedule.get("percents");
    }

}
```

CreditSpecial.java

```
package Credit;

import Credit.Credit;

public class CreditSpecial extends Credit{
    private static double govermentDotationRate = 0.1;

    public CreditSpecial(double sum, String dateTake, String dateGive){
        super(sum, dateTake, dateGive);

        this.total = sum*(1 + Credit.interestRate - govermentDotationRate);
        super.initPaymentSchedule(keyToDate(dateTake), durationMonth);
    }

    public static double getGovermentDotationRate() {
        return govermentDotationRate;
    }
}
```

CreditUniform.java

```
package Credit;

import java.util.HashMap;
import java.util.Calendar;
import java.text.ParseException;

public class CreditUniform extends Credit{
```

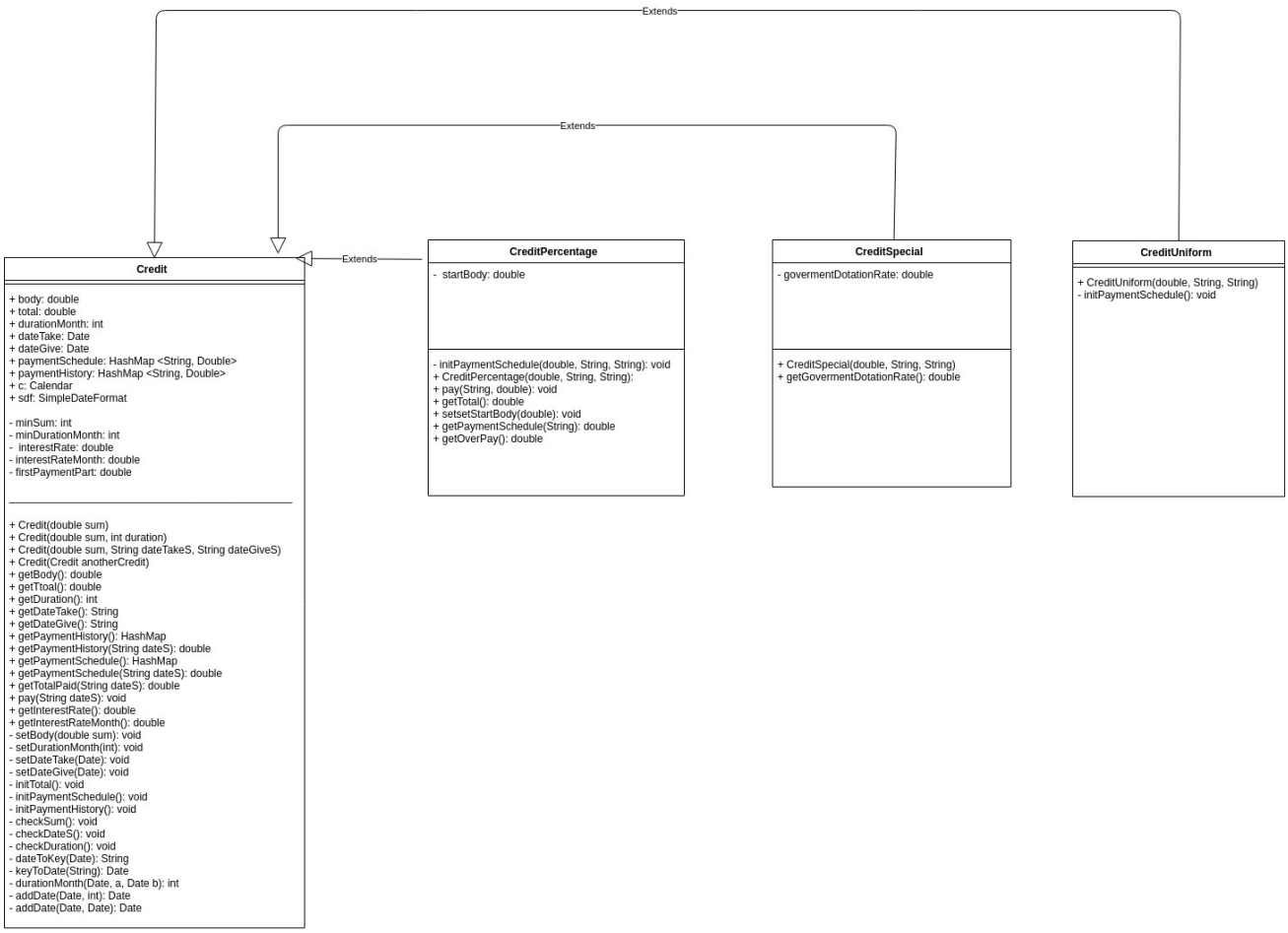
```
    public CreditUniform(double sum, String dateTakeS, String dateGiveS){
        super(sum, dateTakeS, dateGiveS);
        initPaymentSchedule();
    }

    void initPaymentSchedule(){
        paymentSchedule = new HashMap<String, Double>();

        double payment = total/durationMonth;
        for(int i=0; i<durationMonth; ++i){
            c.setTime(dateTake);
            c.add(Calendar.MONTH, i);
            paymentSchedule.put(dateToKey(c.getTime()), payment);
        }
    }


}
```

UML діаграм:



Результат виконання роботи:

Test Credit
Sum: 10000
Interest rate: 0.2
Total sum: 11833.33

Date when credit was taken: 11-2019
Date when credit was given: 10-2020

Pay 3550.00 on 11-2019
Pay 753.03 on 12-2019
Pay 753.03 on 01-2020

Payment Schedule
09-2020: 753.03
06-2020: 753.03
01-2020: 0.00
11-2019: 0.00
03-2020: 753.03
07-2020: 753.03
12-2019: 0.00
02-2020: 753.03
05-2020: 753.03
04-2020: 753.03
08-2020: 753.03

Payment history
01-2020: 753.03
11-2019: 3550.00
12-2019: 753.03

Total paid is 0.00 until 06-2020
Overpay: 1833.33

Test CreditPercentage
Sum: 10000
Interest rate:0.2
Total sum with interest rate: 10000.0
Date when credit taken: 12-2019
Date when credit given: 12-2020
Sum: 10000
Interest rate: 0.2
 Total sum: 10000.00

Date when credit was taken: 12-2019
Date when credit was given: 12-2020

Payment Schedule
09-2020: 0.00
06-2020: 0.00
10-2020: 0.00
03-2020: 0.00
12-2019: 0.00
02-2020: 0.00
05-2020: 0.00

11-2020: 0.00
percents: 650.00
08-2020: 0.00
01-2020: 0.00
07-2020: 0.00
04-2020: 0.00

Payment history
09-2020: 1000.00
06-2020: 1000.00
10-2020: 1000.00
03-2020: 1000.00
12-2019: 1000.00
02-2020: 1000.00
05-2020: 1000.00
08-2020: 1000.00
01-2020: 1000.00
11-2019: 1000.00
07-2020: 1000.00
04-2020: 1000.00
10-2019: 1000.00

Total paid is 1000.00 until 06-2020
Overpay: 650.00

Test CreditUniform
Sum: 10000
Interest rate: 0.2
Total sum: 12000.00

Date when credit was taken: 11-2019
Date when credit was given: 11-2020

Pay 1000.00 on 11-2019
Pay 1000.00 on 12-2019
Pay 1000.00 on 01-2020

Payment Schedule
09-2020: 1000.00
06-2020: 1000.00
10-2020: 1000.00
01-2020: 0.00
11-2019: 0.00
03-2020: 1000.00
07-2020: 1000.00
12-2019: 0.00
02-2020: 1000.00
05-2020: 1000.00
04-2020: 1000.00
08-2020: 1000.00

Payment history
01-2020: 1000.00
11-2019: 1000.00
12-2019: 1000.00

Total paid is 0.00 until 06-2020
Overpay: 2000.00

Test CreditSpecial
Sum: 10000
Interest rate: 0.2
Interest goverment dotation rate: 0.1
Total sum: 11000.00

Date when credit was taken: 11-2019
Date when credit was given: 11-2020

Pay 3300.00 on 11-2019
Pay 641.67 on 12-2019
Pay 641.67 on 01-2020

Payment Schedule
09-2020: 641.67
06-2020: 641.67
10-2020: 641.67
01-2020: 0.00
11-2019: 0.00
03-2020: 641.67
07-2020: 641.67
12-2019: 0.00
02-2020: 641.67
05-2020: 641.67
04-2020: 641.67
08-2020: 641.67

Payment history
01-2020: 641.67
11-2019: 3300.00
12-2019: 641.67

Total paid is 0.00 until 06-2020
Overpay: 1000.00

Висновок: в ході виконання роботи було освоєно не тільки механізм наслідування в мові програмування JAVA, а також типи данних: HashMap, Date, Calendar. Також було освоєно правильний підхід до побудови архітектури класів. Зроблено UML-діаграму.