Основи технологій програмування

Лабораторна робота № 8

Вінницький В’ячеслав Андрійович

ІП-64, 2-ий курс

Кафедра обчислювальної техніки

ІП-6402

Текст програмного коду

```
import java.util.Scanner;
/*
Сформувати набір пропозицій клієнту по цільовим кредитам різних банків.
Враховувати можливість дострокового погашення кредиту й\або збільшення
кредитної лінії. Реалізувати вибір та пошук кредиту за будь-якими
параметрами.
*/
public class Main {
    /**
     * Старт програми відбувається тут
     * Создается список обьектов трёх банков с которыми мы можем выполнять
дальнейшие действия в creditAction
     * @param args
     */
    public static void main(String[] args) {
        //"Вибір кредиту за банком" или "Пошук кредиту за параметрами" или
"Вихід"
        int chooseOrFind;
        //Номер банка
        int check_bank;
        //Тип кредита
        int credit;
        //Сумма кредита
        int suma;
        //Период кредита
        int period;
        //"Достроково погасити" или Збільшити період" или "Нові кредити" или
"Вихід"
        int addAction;
        //Длина нового периода
        int new_period;
        //Возвращаемая сумма
        float newCreditMany;

        CreditAction creditAction = new CreditAction(3);
        MyList myList = new MyList();

        myList.addToList(new Bank1(43.2f, 17.9f, 18.0f, 60, 120));
        myList.addToList(new Bank2(55.6f, 19.9f, 22.0f, 48, 180));
        myList.addToList(new Bank3(49.9f, 19.9f, 19.9f, 72, 240));

        // "i" и "j" управляют выходом из программы
        boolean i = true;
        boolean j;
        while (i){
            j = true;
            System.out.println("1--Вибір кредиту за банком\n2--Пошук кредиту за
параметрами\n3--Вихід");
            chooseOrFind = ChooseAction(3);
            while (j) {
                if (chooseOrFind == 1) {
                    //Ввод от пользователя, выбор банка
                    System.out.println("1--Банк 1\n2--Банк 2\n3--Банк 3");
                    check_bank = ChooseAction(3);
                    //Ввод от пользователя, выбор целевого кредита
                    System.out.println("1--Звичайний кредит\n2--Іпотека\n3--
Кредит на авто");
```

```
                    credit = ChooseAction(3);
                    //Ввод от пользователя, сумма кредита
                    System.out.println("Введіть суму кредиту - ");
                    suma = ChooseAction(500000);
                    //Ввод от пользователя, период кредита
                    System.out.println("На який період - ");
                    float creditMany = 0;
                        try {
                    period =
ChooseAction(creditAction.getMaxMonthsCredit(myList, credit));
                    //Сумма которую нужно отдать банку за кредитный период
                    creditMany = creditAction.getNewCredit(myList, credit, suma,
period);
                    System.out.printf("Взятий новий кредит на суму %d грн на
період %d місяці\nПотрібно повернути %f грн\n",
                        suma, period, creditMany);
                        }catch (MyException e) {
                            System.out.println(e.getNumb());
                            e.purpose();
                        }
                    System.out.println("1--Достроково погасити\n2--Збільшити
період\n3--Нові кредити\n4--Вихід");
                    //Выбор способа взаємодействия
                    addAction = ChooseAction(4);
                if (addAction == 4) {
                        i = false;
                        j = false;
                }else if (addAction == 3) {
                        i = true;
                        j = false;
                }else if (addAction == 2) {

                    //Блок увеличения времени на возвращение кредита
                    System.out.println("Вкажіть новий період - ");

                    try {
                        new_period =
ChooseAction(creditAction.getMaxMonthsCredit(myList, credit));
                        newCreditMany = newPeriod(myList, credit, suma,
new_period, creditAction);
                        System.out.printf("Треба було повернути %f грн\nА тепер
треба повернути %f грн\n", creditMany, newCreditMany);
                        creditMany = newCreditMany;
                        period = new_period;
                    }catch (MyException e){
                        System.out.println(e.getNumb());
                        e.purpose();}

                } else {
                    //Досрочное погашение кредита
                    System.out.printf("Достроково погасити %f грн?\n1--Так\n2--
Ні\n", creditMany);
                    if (ChooseAction(2) == 1)
                        System.out.printf("Кредит %f грн достроково погашено,
заплачено було %d\n", creditMany, suma);
                    else
                        j = false;
                }
```

```
                }
                else if (chooseOrFind == 2) {
                    //Поиск по пареметрам(сортировка)
                    System.out.println("1--Найменший та найбільший відсоток\n2--
Найдовший та найменший період\n3--До головного меню");
                    credit = ChooseAction(3);
                    if (credit == 3)
                        j = false;
                    else if(credit == 2){
                        System.out.printf("Найдовший період - %d\nНайкоротший -
%d\n",
                                creditAction.MinMaxMonths(myList,0),
creditAction.MinMaxMonths(myList,5));
                    }else
                        System.out.printf("Найдовший період - %f\nНайкоротший -
%f\n",
                                creditAction.MinMaxCredit(myList,0),
creditAction.MinMaxCredit(myList,5));

                }
                else {
                    j = false;
                    i = false;
                }
            }
        }
    }

    /**
     * Метод для выбора действия в интерфейсе
     * @param max максимальное значение(месяцев, процентов)
     * @return Действие в интерфейсе
     */

    private static int ChooseAction(int max) {
        int num;
        Scanner scan = new Scanner(System.in);
        do {
            System.out.printf("Ведіть число від %d до %d: ", 1, max);
            while (!scan.hasNextInt()) {
                System.out.printf("Ведіть число!!! від %d до %d: ", 1, max);
                scan.next();
            }
            num = scan.nextInt();
        }while ((num > max) || (num < 1));
        return num;

    }

    /**
     * @param myList Список банков
     * @param credit Тип кредита
     * @param suma Сумма кредита
     * @param new_period Длина нового периода
     * @param creditAction Обьект нужен для создания нового метода
     * @return Новый период, используется метод getNewCredit, как и для обычного
кредита
     */
    private static float newPeriod(MyList myList, int credit, int suma, int
```

```
new_period, CreditAction creditAction) {

        return creditAction.getNewCredit(myList, credit, suma, new_period);
    }

}
    class Bank1 extends MainBank{
    /** Первый Банк который наслудует главный банк
     * @param percentUsualCredit Процент по обычному кредиту
     * @param percentHomeCredit Процент по  ипотеке
     * @param percentCarCredit Процент по кредиту на авто
     * @param maxMonthsUsual Макс. месяцев по обычному кредиту
     * @param maxMonthsCarAndHome Макс. месяцев по кредиту на авто или дом
     */

    Bank1(float percentUsualCredit, float percentHomeCredit, float
percentCarCredit,
          int maxMonthsUsual, int maxMonthsCarAndHome) {
        super(percentUsualCredit, percentHomeCredit, percentCarCredit,
                maxMonthsUsual, maxMonthsCarAndHome);
    }
}
class Bank2 extends MainBank{
    /**
     *  Второй Банк который наслудует главный банк
     * @param percentUsualCredit Процент по обычному кредиту
     * @param percentHomeCredit Процент по  ипотеке
     * @param percentCarCredit Процент по кредиту на авто
     * @param maxMonthsUsual Макс. месяцев по обычному кредиту
     * @param maxMonthsCarAndHome Макс. месяцев по кредиту на авто или дом
     */
    Bank2(float percentUsualCredit, float percentHomeCredit, float
percentCarCredit,
          int maxMonthsUsual, int maxMonthsCarAndHome) {
        super(percentUsualCredit, percentHomeCredit, percentCarCredit,
                maxMonthsUsual, maxMonthsCarAndHome);
    }
}
class Bank3 extends MainBank {
    /**
     *  Третий Банк который наслудует главный банк
     * @param percentUsualCredit Процент по обычному кредиту
     * @param percentHomeCredit Процент по  ипотеке
     * @param percentCarCredit Процент по кредиту на авто
     * @param maxMonthsUsual Макс. месяцев по обычному кредиту
     * @param maxMonthsCarAndHome Макс. месяцев по кредиту на авто или дом
     */
    Bank3(float percentUsualCredit, float percentHomeCredit, float
percentCarCredit,
          int maxMonthsUsual, int maxMonthsCarAndHome) {
        super(percentUsualCredit, percentHomeCredit, percentCarCredit,
                maxMonthsUsual, maxMonthsCarAndHome);
    }
}
import java.util.Arrays;

public class CreditAction {
    private MainBank[] mainBanks;
    /**
```

```
     * Конструктор класса
     * @param numberOfBank Розмерность массива обьектов
     */
    CreditAction(int numberOfBank) {
        mainBanks = new MainBank[numberOfBank];
    }
    /**
     * В этом методе берётся новый кредит
     * @param list Список банков
     * @param credit Тип кредита
     * @param suma Сумма кредита
     * @param period Период
     * @return Сумма к погашению
     */
    public float getNewCredit(MyList list, int credit, int suma, int period) {
        float percentCredit = 0.0f;
        switch (credit) {
            case 1:
                percentCredit = list.get(0).getPercentUsualCredit();
                break;
            case 2:
                percentCredit = list.get(1).getPercentHomeCredit();
                break;
            case 3:
                percentCredit = list.get(2).getPercentCarCredit();
                break;
        }
        return suma * (1 + (percentCredit / 100) * (period / 12));
    }

    /**
     * @param list Список банков
     * @param credit Тип кредита
     * @return Максимальний период на который можно взять кредит
     */

    public int getMaxMonthsCredit(MyList list, int credit) throws MyException{
        int a = 0;
        switch (credit) {

            case 1:
                a = list.get(0).getMaxMonthsUsual();
                break;
            case 2:
                a = list.get(1).getMaxMonthsCarAndHome();
                break;
            case 3:
                a = list.get(2).getMaxMonthsCarAndHome();
                break;
        }
        return a;
    }

    /**
     * @param list Список банков
     * @param who Индекс
     * @return Значения минимальных та максимальных месяцев

     */
```

```java
    public int MinMaxMonths(MyList list,int who) {
        int[] creditMonths = new int[6];
        int i = 0;
        for(int j = 0 ,n = list.size(); j < n ; j++) {
            creditMonths[i] = list.get(j).getMaxMonthsUsual();
            creditMonths[i + 1] = list.get(j).getMaxMonthsCarAndHome();
            i += 2;
        }
        System.out.println(Arrays.toString(creditMonths));
        int flag;
        for (int j = 1; j < creditMonths.length; j++) {
            for (int k = creditMonths.length - 1; k >= j; k--) {
                if (creditMonths[k] > creditMonths[k - 1]) {
                    flag = creditMonths[k - 1];
                    creditMonths[k - 1] = creditMonths[k];
                    creditMonths[k] = flag;
                }
            }
        }

        System.out.println(Arrays.toString(creditMonths));
        if (who == 0)
            return creditMonths[who];
        else
            return creditMonths[who];
    }

    /** Функция дублирует предидущую, но для float значений
     * @param list Список банков
     * @param who Индекс
     * @return Значения минимальных та максимальных процентов
     */
    public float MinMaxCredit(MyList list,int who) {
        float[] creditCredit = new float[9];
        int i = 0;
        for(int j = 0 ; j< list.size();j++) {
            creditCredit[i] =  list.get(j).getPercentCarCredit();
            creditCredit[i + 1] = list.get(j).getPercentHomeCredit();
            creditCredit[i + 2] = list.get(j).getMaxMonthsUsual();
            i += 3;
        }
        float flag;
        for (int j = 1, n = creditCredit.length; j < n; j++) {
            for (int k = creditCredit.length - 1; k >= j; k--) {
                if (creditCredit[k] > creditCredit[k - 1]) {
                    flag = creditCredit[k - 1];
                    creditCredit[k - 1] = creditCredit[k];
                    creditCredit[k] = flag;
                }
            }
        }
        if (who == 0)
            return creditCredit[who];
        else
            return creditCredit[creditCredit.length - 1];
    }
}
/**
 * Интерфейс создан из-за ошибки в IDE
```

```java
 * При присутствии метода AddList в интерфейсе IMyList
 * происходила ошибка типв "method does not override from its superclass"
 */
public interface HelpInterface {
    /**
     * Метод добавляет в список новый банк
     * @param item Банк который нужно добавить
     * @return Список с добавленным елементом Банк
     */
    boolean addToList(MainBank item);
}
public interface IMyList<MainBank> extends Iterable<MainBank>, HelpInterface{
    /**
     *
     * @return Возвращает размер списка(заполненные)
     */
    int size();

    /**
     *
     * @return Возвращает полный размер списка (ячейки с выделенной памятью)
     */
    int allSize();

    /**
     *
     * @param index индекс банка в списке
     * @return Возвращает Банк
     */
    MainBank get(int index);

    /**
     * @param index индекс банка в списке
     * @return Возвращает процент по обычному кредиту
     */
    float getpercentUsualCredit(int index);

    /**
     * @param index индекс банка в списке
     * @return Возвращает процент по ипотеке
     */
    float getpercentHomeCredit(int index);

    /**
     * @param index индекс банка в списке
     * @return Возвращает процент по кредиту на авто
     */
    float getpercentCarCredit(int index);

    /**
     * @param index индекс банка в списке
     * @return Возвращает максимальное кол-во месяцев по ипотеке
     */
    int getmaxMonthsUsual(int index);

    /**
     * @param index индекс банка в списке
     * @return Возвращает максимальное кол-во месяцев по кредиту
     */
```

```
    int maxMonthsCarAndHome(int index);


}public abstract class MainBank {
    //Процент по обычному кредиту и ипотеке
    private int maxMonthsUsual, maxMonthsCarAndHome;
    //Процент по кредиту на авто,  Макс. месяцев по обычному кредиту, по кредиту
на авто или ипотеке
    private float percentUsualCredit, percentHomeCredit, percentCarCredit;

    /**
     * Конструктор
     * @param percentUsualCredit Процент по обычному кредиту
     * @param percentHomeCredit Процент по  ипотеке
     * @param percentCarCredit Процент по кредиту на авто
     * @param maxMonthsUsual Макс. месяцев по обычному кредиту
     * @param maxMonthsCarAndHome Макс. месяцев по кредиту на авто или ипотеке
     */
    MainBank(float percentUsualCredit, float percentHomeCredit, float
percentCarCredit,
             int maxMonthsUsual, int maxMonthsCarAndHome) {
        this.percentCarCredit = percentCarCredit;
        this.percentHomeCredit = percentHomeCredit;
        this.percentUsualCredit = percentUsualCredit;
        this.maxMonthsUsual = maxMonthsUsual;
        this.maxMonthsCarAndHome = maxMonthsCarAndHome;
    }

    /**
     * @return Возвращает максимальное кол-во месяцев по ипотеке
     */
    public int getMaxMonthsUsual() {
        return maxMonthsUsual;
    }

    /**
     * @return @return Возвращает максимальное кол-во месяцев по кредиту
     */
    public int getMaxMonthsCarAndHome() {

        return maxMonthsCarAndHome;
    }

    /**
     * @return Возвращает процент по обычному кредиту
     */
    public float getPercentUsualCredit() {

        return percentUsualCredit;
    }

    /**
     * @return Возвращает процент по ипотеке
     */
    public float getPercentHomeCredit() {

        return percentHomeCredit;
    }
```

```
    /**
     * @return Возвращает процент по кредиту на авто
     */
    public float getPercentCarCredit() {

        return percentCarCredit;
    }
}
public class MyException extends Exception{
    private int wrongNumber;
    public  int getNumb(){return wrongNumber;}
    public void purpose(){
        System.out.println("Please enter correct data ");
    }
    public MyException(String message, int num){

        super(message);
        wrongNumber = num;
    }

}

import java.util.Iterator;

public class MyIterable<T> implements Iterator<T> {

    /**
     * индекс в списке
     */
    private int index = 0;
    /**
     * Значение
     */
    private T[] values;

    MyIterable(T[] values){
        this.values = values;
    }

    /**
     * @return Если hasNext() вызывается впервые - вернет true
     */
    @Override
    public boolean hasNext() {
        return false;
    }

    /**
     * @return  Возвращает текущий ел-т и ссылку на следующий
     */
    @Override
    public T next() {
        return null;
    }
}
import java.lang.*;
import java.util.ArrayList;
import java.util.Iterator;
```

```java
public class MyList implements IMyList {

    /**
     * Стандартный размер
     */
    final static int DEFAULT_CAPACITY = 10;
    /**
     * Массив обьектов "Банк"
     */
    private MainBank[] banks;

    /**
     * Чтобы знать актуальный размер списка
     */
    private int pointer = 0;
    /**
     * Размер
     */
    private int size = 10;

    /**
     * Создание списка с дефолтной длины
     */
    public  MyList(){
        banks = new MainBank[DEFAULT_CAPACITY];
    }

    /**
     * Конструктор с добавлением первого банка
     * @param value первый банк
     */
    public  MyList(MainBank value){

        banks = new MainBank[DEFAULT_CAPACITY];
        addToList(value);
    }

    /**
     * Конструктор с добавлением банков
     * @param list список
     */
    public  MyList(ArrayList<MainBank> list){
        try {
            banks = new MainBank[list.size()];
        }
        catch (NegativeArraySizeException e) {
            e.getMessage();
        }

        for (int i = 0, n = list.size(); i < n; i++){
            banks[i] = list.get(i);
        }
    }

    /**
     * @return Возвращает размер списка(заполненные)
     */
    @Override
    public int size() {
```

```
        int count = 0;
        while(banks[count]!= null) {

            count++;
        }
        return count;
    }


    /**
     * @return Возвращает полный размер списка (ячейки с выделенной памятью)
     */
    @Override
    public int allSize() {
        return banks.length;
    }

    /**
     * @param index индекс банка в списке
     * @return Возвращает Банк
     */
    @Override
    public MainBank get(int index) {
        return banks[index];
    }

    /**
     * @param index индекс банка в списке
     * @return Возвращает процент по обычному кредиту
     */
    @Override
    public float getpercentUsualCredit(int index) {
        return banks[index].getPercentUsualCredit();
    }

    /**
     * @param index индекс банка в списке
     * @return Возвращает процент по ипотеке
     */
    @Override
    public float getpercentHomeCredit(int index) {
        return banks[index].getPercentHomeCredit();
    }

    /**
     * @param index индекс банка в списке
     * @return Возвращает процент по кредиту на авто
     */
    @Override
    public float getpercentCarCredit(int index) {
        return banks[index].getPercentCarCredit();
    }

    /**
     * @param index индекс банка в списке
     * @return Возвращает максимальное кол-во месяцев по ипотеке
     */
    @Override
    public int getmaxMonthsUsual(int index) {
```

```
        return banks[index].getMaxMonthsUsual();
    }

    /**
     * @param index индекс банка в списке
     * @return Возвращает максимальное кол-во месяцев по кредиту
     */
    @Override
    public int maxMonthsCarAndHome(int index) {
        return banks[index].getMaxMonthsUsual();
    }

    /**
     * @return Итератор
     */
    @Override
    public Iterator<MainBank> iterator() {
        return new MyIterable<>(banks);
    }

    /**
     *  Добавляет новый элемент в список. При достижении размера внутреннего
     массива происходит его увеличение в два раза.
     * @param item Банк который нужно добавить
     * @return  Список с добавленным елементом Банк
     */
    @Override
    public boolean addToList(MainBank item) {
        try {
            if(pointer >= size/2){// если размер списка уже больше чем половина
заданного размера, то размер = размер*1.5
                size = size + size >> 1;
                MainBank[] temp = banks;
                banks = new MainBank[size];
                System.arraycopy(temp, 0, banks, 0, temp.length);//перемещаем
массив
            }
            banks[pointer] = item;
            pointer++;
            return true;

        }catch (ClassCastException ex) {
            ex.printStackTrace();
        }
        return false;
    }
}
import org.junit.Test;

import static org.junit.Assert.*;

public class Bank1Test {
    @Test
    public void test1() {
        MainBank bank = new Bank1(43.2f, 17.9f, 18.0f, 60, 120);
        assertEquals(60,bank.getMaxMonthsUsual());
        assertEquals(120,bank.getMaxMonthsCarAndHome());
        assertEquals(43.2f,bank.getPercentUsualCredit(),0.000000000001);
        assertEquals(17.9f,bank.getPercentHomeCredit(),0.000000000001);
```

```
        assertEquals(18.0f,bank.getPercentCarCredit(),0.000000000001);
    }

}
import org.junit.Test;

import static org.junit.Assert.*;

public class Bank2Test {
    @Test
    public void test2() {
        MainBank bank = new Bank2(55.6f, 19.9f, 22.0f, 48, 180);
        assertEquals(48,bank.getMaxMonthsUsual());
        assertEquals(180,bank.getMaxMonthsCarAndHome());
        assertEquals(55.6f,bank.getPercentUsualCredit(),0.000000000001);
        assertEquals(19.9f,bank.getPercentHomeCredit(),0.000000000001);
        assertEquals(22.0f,bank.getPercentCarCredit(),0.000000000001);
    }

}
import org.junit.Test;

import static org.junit.Assert.*;

public class Bank3Test {
    @Test
    public void test3() {
        MainBank bank = new Bank3(43.2f, 17.9f, 18.0f, 60, 120);
        assertEquals(60,bank.getMaxMonthsUsual());
        assertEquals(120,bank.getMaxMonthsCarAndHome());
        assertEquals(43.2f,bank.getPercentUsualCredit(),0.000000000001);
        assertEquals(17.9f,bank.getPercentHomeCredit(),0.000000000001);
        assertEquals(18.0f,bank.getPercentCarCredit(),0.000000000001);
    }
}
import org.junit.Test;

import static org.junit.Assert.*;

public class CreditActionTest {
    @Test
    public void CreditActionTest() {
        CreditAction creditAction = new CreditAction(3);
        MyList myList = new MyList();
        myList.addToList(new Bank1(43.2f, 17.9f, 18.0f, 60, 120));
        myList.addToList(new Bank2(55.6f, 19.9f, 22.0f, 48, 180));
        myList.addToList(new Bank3(49.9f, 19.9f, 19.9f, 72, 240));

assertEquals(creditAction.getNewCredit(myList,1,100,10),100.0f,0.0000001);

        try {
            assertEquals(creditAction.getMaxMonthsCredit(myList,1),60);
        }catch (MyException e){
            System.out.println(e.getNumb());
            e.purpose();
        }
        assertEquals(creditAction.MinMaxCredit(myList,1),17.9f,0.00000001);
        assertEquals(creditAction.MinMaxMonths(myList,1),180);
```

```
    }

}
import org.junit.Test;
import static org.junit.Assert.*;

public class MyListTest {

    @Test
    public void MyListTest(){
            MyList myList = new MyList();
            assertEquals(myList.size(),0);
            assertEquals(myList.allSize(),10);
            CreditAction creditAction = new CreditAction(7);
            assertEquals(myList.allSize(),10);
            myList.addToList(new Bank1(43.2f, 17.9f, 18.0f, 60, 120));
            assertEquals(myList.allSize(),10);
            assertEquals(myList.getpercentUsualCredit(0), 43.2f,0.0000001 );
            assertEquals(myList.getpercentHomeCredit(0), 17.9f,0.0000001 );
            assertEquals(myList.getpercentCarCredit(0), 18.0f,0.0000001 );
            assertEquals(myList.getmaxMonthsUsual(0), 60);
            assertEquals(myList.maxMonthsCarAndHome(0), 60);
            assertEquals(myList.size(),1);


    }
}
import java.io.*;
/**
 * Класс реалізует ввод/вывод данных в файл
 */
public class Serialization {
    /**
     * Метод сериализаации/записи файла колекции как одного обьекта.
     * @param file Название файла
     * @param list Название коллекции
     */
    public static void serializeList(String file, MyList list) throws
IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(file))) {
            oos.writeObject(list);
            oos.flush();
        }
    }
    /**
     * Метод сериализаации/записи файла колекции как последовательности
обьектов.
     * @param file Название файла
     * @param list Название коллекции
     */
    public static void serializeSeq(String file, MyList list) throws IOException
{
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(file))) {
            oos.writeInt(list.size());
            for (int i = 0; i < list.size(); i++) {
```

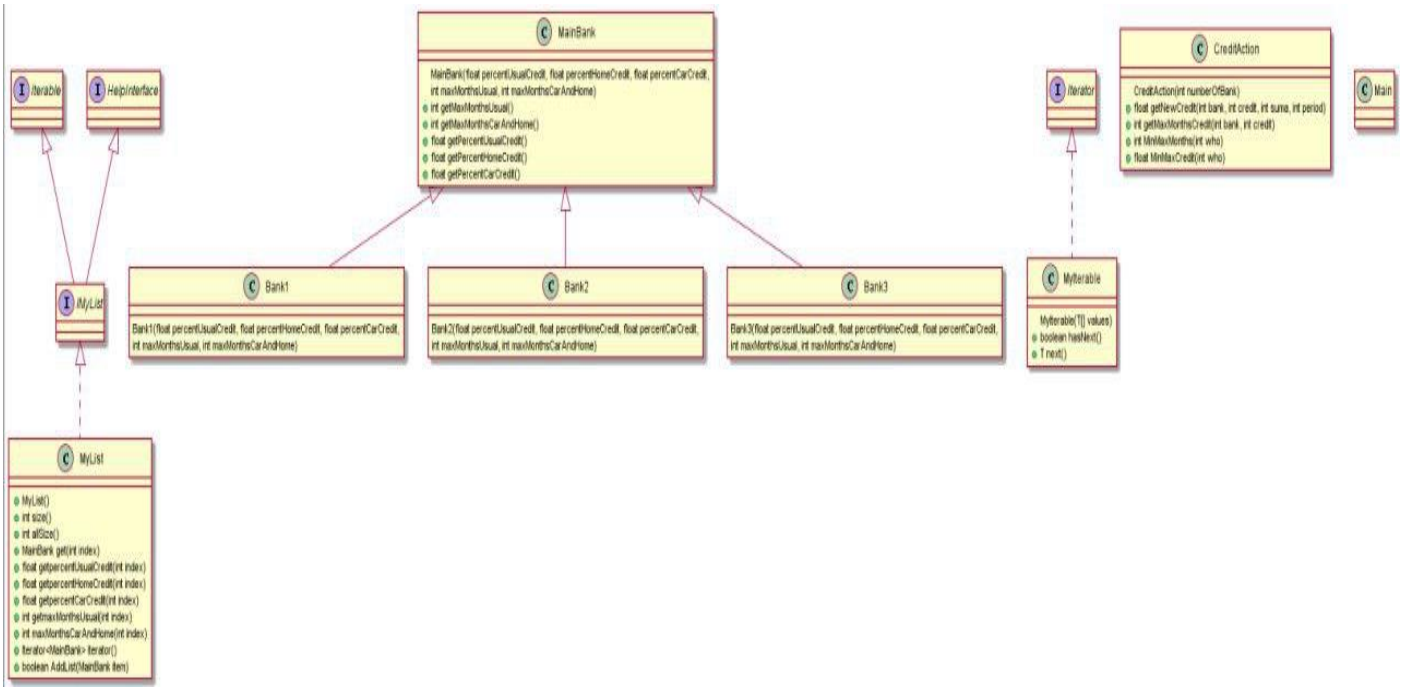
```
                if (list.get(i) != null)
                    oos.writeObject(list.get(i));
            }
            oos.flush();
        }
    }
    /**
     * Метод десериализаации/считывания файла колекции как одного обьекта.
     * @param file Название файла
     */
    public static MyList deserializeList(String file) throws IOException,
ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(file))) {
            return (MyList) ois.readObject();
        }
    }
    /**
     * Метод десериализаации/считывания файла колекции как последовательности
обьектов.
     * @param file Название файла
     */
    public static MyList deserializeSeq(String file) throws IOException,
ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(file))) {
            int size = ois.readInt();
            MyList set = new MyList();
            for (int i = 0; i < size; i++) {
                set.addToList((MainBank) ois.readObject());
            }
            return set;
        }
    }
}
import org.junit.Test;
import java.io.IOException;
import static org.junit.Assert.*;

public class SerializationTest {
    /**
     * Тест проверяет serializeList на ввод/чтение
     */
    @Test
    public void serializeList() throws IOException, ClassNotFoundException {
        MyList myList = new MyList();
        myList.addToList(new Bank1(43.2f, 17.9f, 18.0f, 60, 120));
        Serialization.serializeList("test1.bin", myList);
        MyList list = Serialization.deserializeList("test1.bin");
        assertEquals(list.allSize(), myList.allSize());
    }
    /**
     * Тест проверяет serializeSeq  на ввод/чтение
     */
    @Test
    public void serializeSeq() throws IOException, ClassNotFoundException {
        MyList myList = new MyList();
        myList.addToList(new Bank1(43.2f, 17.9f, 18.0f, 60, 120));
        Serialization.serializeSeq("test2.bin", myList);
```

```
        MyList hashSet = Serialization.deserializeSeq("test2.bin");
        assertEquals(myList.allSize(), hashSet.allSize());
    }


}
```

Діаграма класів


Iterable
Help/interface
MainBank
MainBank(float percentUsualCredit, float percentHomeCredit, float percentCarCredit, int maxMonthsUsual, int maxMonthsCarAndHome)
getMaxMonthsUsual()
getMaxMonthsCarAndHome()
float getPercentUsualCredit()
float getPercentHomeCredit()
float getPercentCarCredit()
Iterator
CreditAction
CreditAction(int numberOfBank)
float getNewCredit(int bank, int credit, int suma, int period)
int getMaxMonthsCredit(int bank, int credit)
int MinMaxMonths(int who)
float MinMaxCredit(int who)
Main
IMyList
Bank1
Bank1(float percentUsualCredit, float percentHomeCredit, float percentCarCredit, int maxMonthsUsual, int maxMonthsCarAndHome)
Bank2
Bank2(float percentUsualCredit, float percentHomeCredit, float percentCarCredit, int maxMonthsUsual, int maxMonthsCarAndHome)
Bank3
Bank3(float percentUsualCredit, float percentHomeCredit, float percentCarCredit, int maxMonthsUsual, int maxMonthsCarAndHome)
Myiterable
Myiterable(T[] values)
boolean hasNext()
T next()
MyList
MyList()
int size()
int allSize()
MainBank get(int index)
float getpercentUsualCredit(int index)
float getpercentHomeCredit(int index)
float getpercentCarCredit(int index)
int getmaxMonthsUsual(int index)
int maxMonthsCarAndHome(int index)
Iterator<MainBank> iterator()
boolean AddList(MainBank item)