

Національний університет “Львівська політехніка”

Кафедра програмного забезпечення

КУРСОВА РОБОТА

з дисципліни «Об’єктно-орієнтоване програмування»

На тему:

«Облік комунальних витрат»

Студента групи ПЗ-25

спеціальності 6.121

«Інженерія програмного забезпечення»

Скаковського А. В.

Керівник: доцент кафедри ПЗ,

к.т.н., доцент Коротєєва Т. О.

Національна шкала _____

Кількість балів ____ Оцінка ECTS ____

Члени комісії _____

Львів – 2023

ЗАВДАННЯ

до курсової роботи з дисципліни

«Об'єктно-орієнтоване програмування»

студента групи ПЗ-25 Скакоського Антона

Тема « *Облік комунальних витрат* »

Варіант:

16. У середовищі Візуального програмування створити програму для обліку витрат по кожній з квартир у багатоквартирному будинку. Кількість квартир визначається статично. Розрахунковий період – 1 місяць. Інформація про квартиру:

Номер квартири | Прізвище власника | Спожито води (кубометрів) | Спожито світла (кВт·год)

Функціональність програми:

1) Можливість внесення інформації по кожній з квартир для визначеного користувачем місяця. Відображення місяців, для яких інформація не заповнена для кожної з квартир.

2) Збереження та зчитування річної інформації з файлу, назва файлу відповідає року.

3) Можливість встановлення ціни в межах року за один кубометр води та за одну кВт·год. Ціна записується у файл року. Розрахунок та відображення вартості спожитих послуг по місячно, агреговану для всіх квартир.

4) Можливість завантаження у програму більше одного року.

5) Вивести власників, які за рік спожили найменше послуг в грошовому еквіваленті та відсортувати їх методом Шелла за прізвищами.

6) Виведення номеру квартири, яка спожила найбільше послуг у грошовому еквіваленті в межах усіх років, які поточно завантаженні у програму.

Для класу створити: 1) Конструктор за замовчуванням; 2) Конструктор з параметрами; 3) конструктор копій; 4) перевизначити операції $>>$, $<<$ для зчитування та запису у файл.

№ з/п	Зміст завдання	Дата
1	Здійснити аналітичний огляд літератури за заданою темою та обґрунтувати вибір інструментальних засобів реалізації.	10.10
2	Побудова UML діаграм	15.10
3	Розробка алгоритмів реалізації	24.10
4	Реалізація завдання (кодування)	04.11
5	Формування інструкції користувача	11.11
6	Оформлення звіту до курсової роботи згідно з вимогами Міжнародних стандартів, дотримуючись такої структури: змiст; алгоритм розв'язку задачі у покроковому представленні; діаграми UML класів, прецедентів, послідовності виконання; код розробленої програми з коментарями; протокол роботи програми для кожного пункту завдання; інструкція користувача та системні вимоги; опис виняткових ситуацій; структура файлу вхідних даних; висновки; список використаних джерел	16.11

Завдання прийнято до виконання: _____ (Скаковський А. В.)

Керівник роботи: _____ (Коротєєва Т. О.)

Дата видачі завдання: 06.09.2023

Зміст

1. Алгоритм розв'язку задачі у покроковому представленні	5
2. Діаграми UML класів, прецедентів, послідовності виконання	7
3. Код розробленої програми	11
4. Протокол роботи програми для кожного пункту завдання	39
5. Інструкція користувача	44
6. Опис виняткових ситуацій	49
7. Структура файлу вхідних даних	61
Висновок	61
Список використаної літератури	63

1. Алгоритм розв'язку задачі у покроковому представленні

1.1 Відображення місяців, для яких інформація не заповнена для кожної з квартир (**NFM**)

fFlats – масив квартир; n – довжина вхідного масиву, i, j – індекси проходу, iResultArray – результуючий масив, iCounter – лічильник пропусків;

NFM – Not Filled Month

NFM1: Ініціалізація: i = 0, j = 0.

NFM2: поки i < 12 виконуємо **NFM3**, інакше **NFM4**.

NFM3: Поки j < n : виконуємо fFlat[j]. isMonthFilled(i):

Якщо false => iCounter++

Інакше перевіряємо чи iCounter != n

Якщо true iResultArray[i] = 0

Якщо false iResultArray[i] = -1

i ++, iCounter = 0, виконуємо **NFM2**.

NFM4: Вихід з алгоритму.

1.2 Розрахунок та відображення вартості спожитих послуг по місячно, агреговану для всіх квартир **CSFEM**

CSFEM – CalculateSum of FlatExpences for Month

fFlats – масив квартир; n – довжина вхідного масиву, i, j – індекси проходу, ResultArray – результуючий масив, TotalMonthExpences – витрати всіх квартир в певний місяць

CSFEM1: Ініціалізація: i = 0, j = 0.

CSFEM2: поки i < 12 виконуємо **CSFEM3**, інакше **CSFEM4**.

CSFEM3: Поки j < n : виконуємо:

TotalMonthExpences += fFlat[j]. calculateMonthExpences(i)

Інакше: ResultArray[i] = TotalMonthExpences

i ++, TotalMonthExpences = 0, виконуємо **CSFEM2**.

CSFEM4: Вихід з алгоритму.

1.3 Вивести власників, які за рік спожили найменше послуг в грошовому еквіваленті та відсортувати їх методом Шелла за прізвищами (**ShSS**)

ShSS – Shell Sort Surname

sSurnameList – вхідний масив прізвищ власників; i, j – індекси проходу; n – розмір масиву; step – крок.

ShSS1. Крок сортування: $\text{step} = n/2$.

Запустити цикл сортування: поки $\text{step} > 0$, виконувати кроки **ShSS2** та **ShSS3**, інакше перейти до виконання **ShSS4**.

ShSS2. Ініціалізація: $i = 1$. Поки $(i \leq n - \text{step})$ $j = i$.

Поки $(j \geq 0)$ та елементи $\text{StringCompare}(\text{sSurnameList}[i], \text{sSurnameList}[j+\text{step}]) == 1$, то $\text{sSurnameList}[i] = \text{sSurnameList}[j+\text{step}]$, $j--$, $i++$.

ShSS3. $\text{step} = \text{step}/2$.

ShSS4. Вихід з алгоритму.

1.4 Виведення номеру квартири, яка спожила найбільше послуг у грошовому еквіваленті в межах усіх років, які поточно завантаженні у програму (**FMEO**)
FMEO - Find Most Expendable Owner

fFlats – масив квартир; n – довжина вхідного масиву; i, j – індекси проходу, TotalFlatExpences – витрати квартир за всі роки; MaxFlatExpences – максимальне значення витрат квартири за всі роки, YearPrices – масив з цінами на кожен рік

FMEO1: Ініціалізація: $i = 0, j = 0, \text{MaxFlatExpences} = 0$.

FMEO2: поки $i < n$ виконуємо **FMEO3**, інакше **FMEO4**.

FMEO3: $\text{TotalFlatExpences} = \text{fFlats}[i].\text{calculateTotalExpences}(\text{YearPrices})$

Якщо $\text{TotalFlatExpences} > \text{MaxFlatExpences}$, то

$\text{MaxFlatExpences} = \text{TotalFlatExpences}$

$i++$ виконуємо **FMEO2**.

FMEO4: Вихід з алгоритму.

2. Діаграми UML класів, прецедентів, послідовності виконання

2.1 UML діаграма класів:

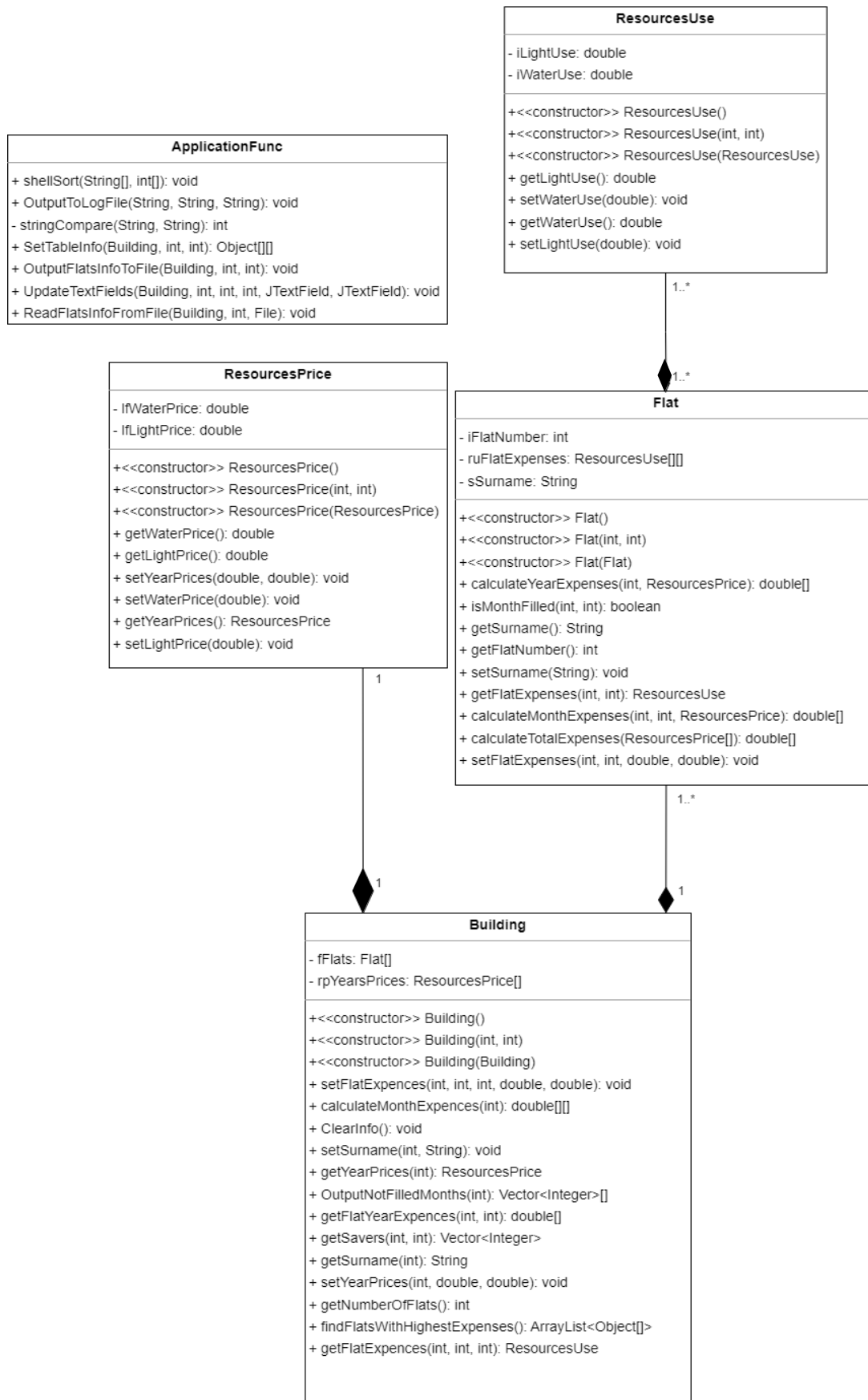


Рис. 2.1. Діаграма класів

2.2. UML діаграма прецедентів:

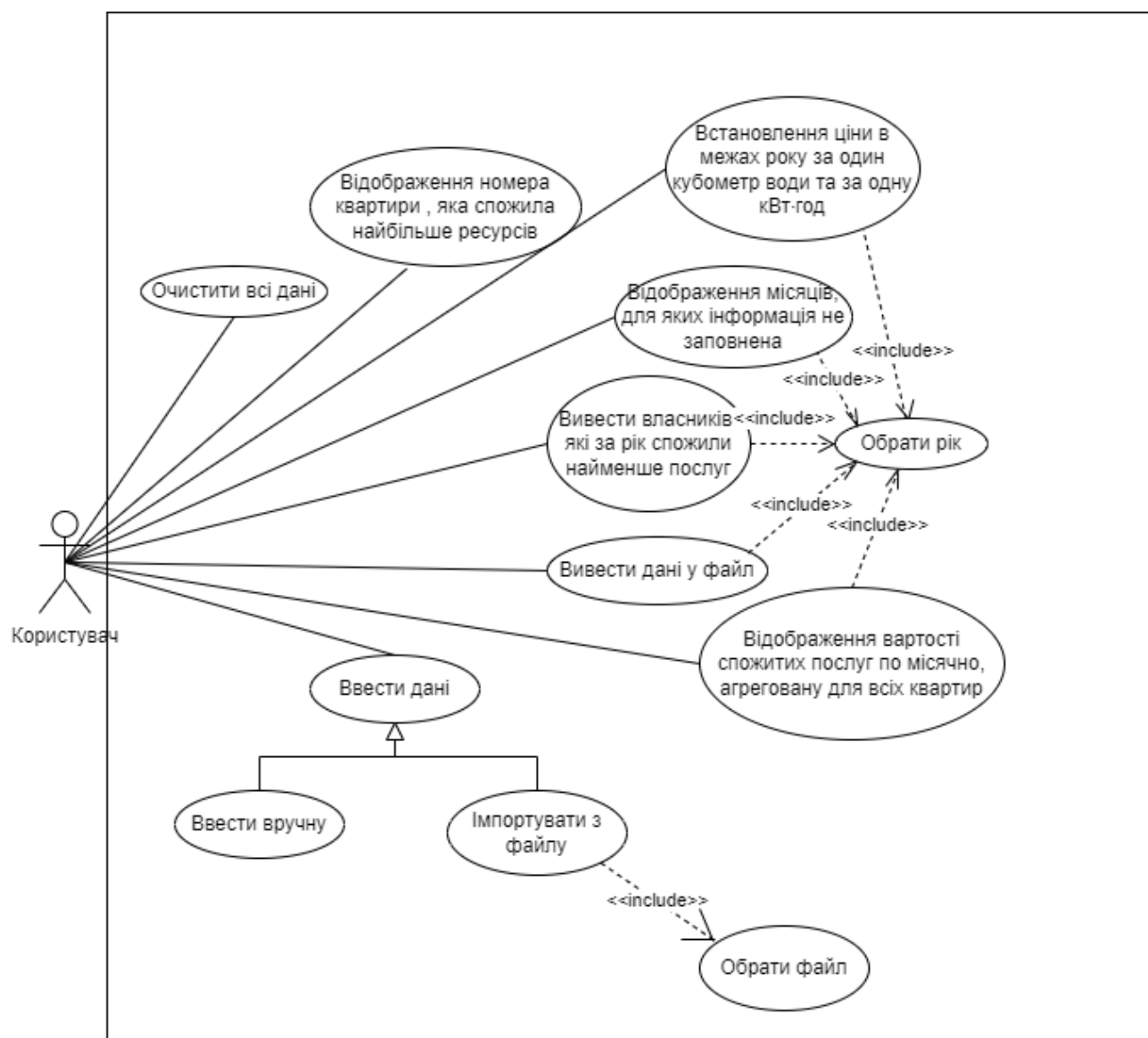
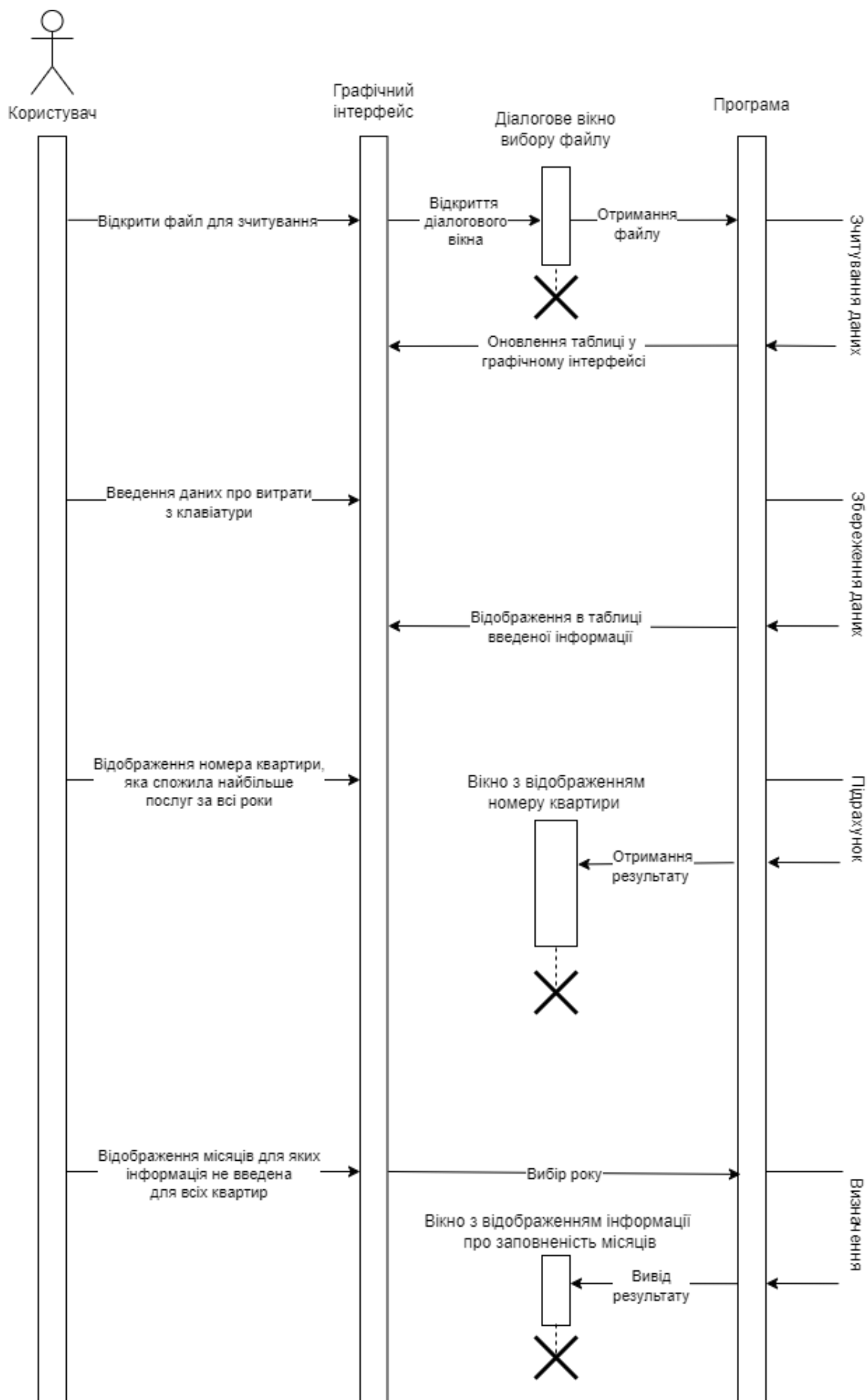


Рис. 2.2. Діаграма прецедентів

2.3. UML діаграма послідовності:



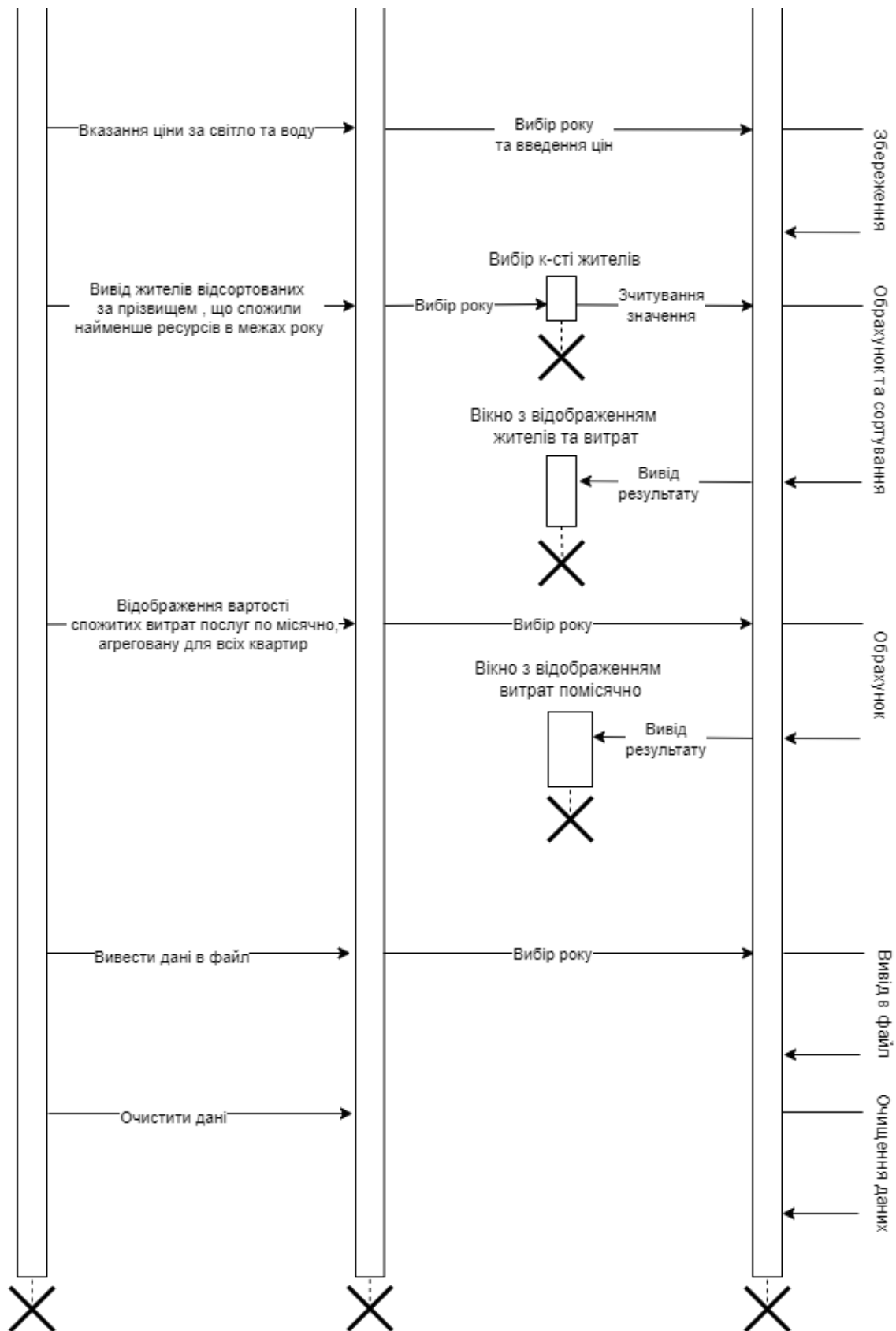


Рис. 2.3. Діаграма послідовності

3. Код розробленої програми

ApplicationFunc.java

```
import javax.swing.*;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;

//абстрактний клас ApplicationFunc, що містить методи-функції необхідні для
роботи програми
public abstract class ApplicationFunc {
    //функція для зчитування даних з файлу
    public static void ReadFlatsInfoFromFile(Building bApartment, int iYearIndex,
File selFile) {

        try {
            BufferedReader br = new BufferedReader(new FileReader(selFile));

            String firstLine = br.readLine();
            String[] prices = firstLine.split(" ");

            double waterPrice = Double.parseDouble(prices[0]);
            double lightPrice = Double.parseDouble(prices[1]);

            bApartment.setYearPrices(iYearIndex, waterPrice, lightPrice);

            for (int i = 0; i < bApartment.getNumberOfFlats() ; i++) {

                if(br.readLine() == null){
                    bApartment.ClearInfo();
                    throw new RuntimeException("Your file don't contain enough
info");
                }

                bApartment.setSurname(i, br.readLine());

                for (int j = 0; j < 12; j++) {

                    String line = br.readLine();
                    String[] values = line.split(" ");

                    double waterUse = Double.parseDouble(values[1]);
                    double lightUse = Double.parseDouble(values[2]);

                    bApartment.setFlatExpences(i, iYearIndex, j, waterUse,
lightUse);
                }
            }

            if(br.readLine() != null){
                throw new RuntimeException("Your file contain more info that
needed");
            }

            br.close();
            JOptionPane.showMessageDialog(null, "Information successfully read");
        }
        catch (RuntimeException exc){
            JOptionPane.showMessageDialog
(null, exc.getMessage(), "Warning:\\", JOptionPane.WARNING_MESSAGE);
        }
        catch (Exception exc){
            JOptionPane.showMessageDialog (null,
```

```

exc.getMessage(), "Error: (", JOptionPane.ERROR_MESSAGE);
    }
}
// функція для виводу в лог файл
public static void OutputToLogFile(String sFilePath, String sFuncName,
String sLogBody) {
    try (FileWriter fw = new FileWriter(sFilePath, true);
        PrintWriter pw = new PrintWriter(fw)) {

        // Отримуємо поточну дату та час
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy
HH:mm");

        String dateTime = dateFormat.format(new Date());

        // Записуємо лог у файл
        pw.println("[ " + dateTime + " ] Назва функції: " + sFuncName + " {");
        pw.println("\n" + sLogBody);
        pw.println("}\n\n");

    } catch (IOException e) {
        e.printStackTrace();
    }
}
//функція для виводу даних у файл
public static void OutputFlatsInfoToFile(Building bApartment, int
iCurOutputYear, int iStartYear){
    try{
        String filePath = "Q:/Anton/Education/Kyrsova/CourseWork/" +
iCurOutputYear + ".txt";
        File file = new File(filePath);

        FileWriter fwWrite = new FileWriter(file, false);

        ResourcesPrice rpYearPrice = bApartment.getYearPrices(iCurOutputYear
- iStartYear);
        fwWrite.write(rpYearPrice.getWaterPrice() + " " +
                        rpYearPrice.getLightPrice() + "\n");

        for (int i = 0; i < bApartment.getNumberOfFlats(); i++) {

            fwWrite.write((i + 1) + "\n");
            fwWrite.write(bApartment.getSurname(i) + "\n");

            for (int j = 0; j < 12; j++) {

                ResourcesUse ruFlatExpences =
bApartment.getFlatExpences(i, iCurOutputYear - iStartYear, j);
                fwWrite.write((j + 1) + " " +
                                ruFlatExpences.getWaterUse() + " " +
                                ruFlatExpences.getLightUse() + "\n");
            }
        }
        fwWrite.close();
        JOptionPane.showMessageDialog(null, "Information successfully
saved");
    }
    catch (RuntimeException exc) {
        JOptionPane.showMessageDialog(null, exc.getMessage(), "Warning:\\",
JOptionPane.WARNING_MESSAGE);
    }
    catch (Exception exc){
        JOptionPane.showMessageDialog(null,
exc.getMessage(), "Error: (", JOptionPane.ERROR_MESSAGE);
    }
}

```

```

    }
    //функція для оновлення текстових полів при додаванні інформації
    public static void UpdateTextFields(Building bApartment, int flat, int year,
    int month, JTextField tfWaterUse, JTextField tfLightUse) {

        ResourcesUse ruFlatUse = bApartment.getFlatExpences(flat, year, month);

        if (!(ruFlatUse.getLightUse() == -1 || ruFlatUse.getWaterUse() == -1)) {
            tfLightUse.setText(String.valueOf(ruFlatUse.getLightUse()));
            tfWaterUse.setText(String.valueOf(ruFlatUse.getWaterUse()));
        }
        else {
            tfLightUse.setText("");
            tfWaterUse.setText("");
        }
    }
    //функція для оновлення головної таблиці
    public static Object[][] SetTableInfo(Building bApartment, int year, int
    month) {

        Object[][] oTableData = new Object[bApartment.getNumberOfFlats()][4];

        for (int i = 0; i < oTableData.length; i++) {

            oTableData[i][0] = i + 1;

            oTableData[i][1] = bApartment.getSurname(i);

            ResourcesUse FlatUse = bApartment.getFlatExpences(i, year, month);

            if (FlatUse.getWaterUse() == -1) {
                oTableData[i][2] = "not filled";
            }
            else {
                oTableData[i][2] = FlatUse.getWaterUse();
            }

            if (FlatUse.getLightUse() == -1) {
                oTableData[i][3] = "not filled";
            }
            else {
                oTableData[i][3] = FlatUse.getLightUse();
            }
        }
        return oTableData;
    }
    //функція для порівняння стрічок
    private static int stringCompare(String sFirstString, String sSecondString)
    {
        int iMaxIteration = Math.min(sFirstString.length(),
        sSecondString.length());

        for (int i = 0; i < iMaxIteration; i++) {
            if (sFirstString.charAt(i) < sSecondString.charAt(i)) {
                return -1;
            } else if (sFirstString.charAt(i) > sSecondString.charAt(i)) {
                return 1;
            }
        }
        return Integer.compare(sFirstString.length(), sSecondString.length());
    }
    //функція для сортування прізвищ
    public static void shellSort(String[] sSurnames, int[] iFlatNumbers) {
        int n = sSurnames.length;

        for (int gap = n / 2; gap > 0; gap /= 2) {

```

```

        for (int i = gap; i < n; i++) {
            String tempSurname = sSurnames[i];
            int tempFlatNumber = iFlatNumbers[i];
            int j;

            for (j = i; j >= gap && stringCompare(sSurnames[j - gap],
tempSurname) > 0; j -= gap) {
                sSurnames[j] = sSurnames[j - gap];
                iFlatNumbers[j] = iFlatNumbers[j - gap];
            }

            sSurnames[j] = tempSurname;
            iFlatNumbers[j] = tempFlatNumber;
        }
    }
}

```

Building.java

```

import java.time.Year;
import java.util.*;

public class Building{
    //змінні класу Building
    private final Flat[] fFlats;
    private final ResourcesPrice[] rpYearsPrices;

    // конструктор з параметрами класу Building
    public Building(int iFlatCount, int iStartYear){

        fFlats = new Flat[iFlatCount];

        for (int i = 0; i < iFlatCount; i++) {
            fFlats[i] = new Flat(i + 1, iStartYear);
        }

        rpYearsPrices = new ResourcesPrice[Year.now().getValue() - iStartYear +
1];

        for (int i = 0; i < rpYearsPrices.length; i++) {
            rpYearsPrices[i] = new ResourcesPrice();
        }
    }
    // конструктор копіювання класу Building
    public Building(Building buildingToCopy) {
        this.fFlats = new Flat[buildingToCopy.fFlats.length];
        for (int i = 0; i < buildingToCopy.fFlats.length; i++) {
            this.fFlats[i] = new Flat(buildingToCopy.fFlats[i]);
        }

        this.rpYearsPrices = new
ResourcesPrice[buildingToCopy.rpYearsPrices.length];
        for (int i = 0; i < buildingToCopy.rpYearsPrices.length; i++) {
            this.rpYearsPrices[i] = new
ResourcesPrice(buildingToCopy.rpYearsPrices[i]);
        }
    }
    //сеттери класу Building
    public void setFlatExpences(int iFlatNumber, int iYear, int iMonth, double
waterUse, double lightUse){
        fFlats[iFlatNumber].setFlatExpenses(iYear, iMonth, waterUse, lightUse);
    }
}

```

```

public void setYearPrices(int iYear, double waterPrice, double lightPrice){
    rpYearsPrices[iYear].setWaterPrice(waterPrice);
    rpYearsPrices[iYear].setLightPrice(lightPrice);
}
void setSurname(int iFlatNumber, String sSurname){
    fFlats[iFlatNumber].setSurname(sSurname);
}

//геттери класу Building
public ResourcesUse getFlatExpences(int iFlatNumber,int iYear, int iMonth){
    return fFlats[iFlatNumber].getFlatExpences(iYear,iMonth);
}
public ResourcesPrice getYearPrices(int iYear){
    return rpYearsPrices[iYear].getYearPrices();
}
public int getNumberOfFlats(){
    return fFlats.length;
}
String getSurname(int iFlatNumber){
    return fFlats[iFlatNumber].getSurname();
}

//функція для визначення витрат квартири за певний рік
public double[] getFlatYearExpences(int iFlatNumber,int iYear){
    double[] FlatYearExpences =
fFlats[iFlatNumber].calculateYearExpences(iYear,rpYearsPrices[iYear]);
    return new double[]{FlatYearExpences[0],FlatYearExpences[1]};
}

//функція для знаходження квартири з найбільшими витратами за всі роки
public ArrayList<Object[]> findFlatsWithHighestExpences() {

    for (int i = 0; i < rpYearsPrices.length; i++) {
        if (rpYearsPrices[i].getWaterPrice() == 0 ||
rpYearsPrices[i].getLightPrice() == 0) {
            throw new RuntimeException("You haven't entered prices for all
years");
        }
    }

    double maxExpenses = -1;
    ArrayList<Object[]> maxFlatInfo = new ArrayList<>();

    for (int i = 0; i < fFlats.length; i++) {
        double[] totalExpenses =
fFlats[i].calculateTotalExpences(rpYearsPrices);
        if(totalExpenses[0] == -1 && totalExpenses[1] == -1){
            throw new RuntimeException("You have not filled information");
        }
        double waterExpenses = totalExpenses[0];
        double lightExpenses = totalExpenses[1];

        if (waterExpenses + lightExpenses > maxExpenses) {
            maxExpenses = waterExpenses + lightExpenses;
            maxFlatInfo.clear();
            maxFlatInfo.add(new Object[]{fFlats[i].getFlatNumber(),
waterExpenses, lightExpenses});
        } else if (waterExpenses + lightExpenses == maxExpenses) {
            maxFlatInfo.add(new Object[]{fFlats[i].getFlatNumber(),
waterExpenses, lightExpenses});
        }
    }
    return maxFlatInfo;
}

//функція для виводу інформація про заповненість інформацією місяців

```

```

public Vector<Integer>[] OutputNotFilledMonths(int year) {
    Vector<Integer>[] vMonthInfo = new Vector[12];

    for (int i = 0; i < 12; i++) {
        Vector<Integer> vUnfilledFlats = new Vector<>();

        for (int j = 0; j < fFlats.length; j++) {
            if (!fFlats[j].isMonthFilled(year, i)) {
                vUnfilledFlats.add(j + 1);
            }
        }

        // Інформація заповнена для всіх квартир
        if (vUnfilledFlats.isEmpty()) {
            vUnfilledFlats.add(0);
        }
        // Інформація не заповнена для всіх квартир
        else if (vUnfilledFlats.size() == fFlats.length) {
            vUnfilledFlats.clear();
            vUnfilledFlats.add(-1);
        }

        vMonthInfo[i] = vUnfilledFlats;
    }
    return vMonthInfo;
}

//функція для знаходження квартир з найменшими витратами
public Vector<Integer> getSavers(int iFlatCount, int iYear) {
    int len = fFlats.length;
    Vector<Double> iFlatsExpenses = new Vector<>();

    Vector<Integer> iSortedFlatNumberList = new Vector<>();
    for (int i = 0; i < len; i++) {

        double[] totalExpenses = fFlats[i].calculateYearExpenses(iYear,
rpYearsPrices[iYear]);
        if (totalExpenses[0] != -1 && totalExpenses[1] != -1) {
            iSortedFlatNumberList.add(i);
            iFlatsExpenses.add(totalExpenses[0] + totalExpenses[1]);
        }
    }
    if (iSortedFlatNumberList.isEmpty()) {
        throw new RuntimeException("You haven't entered usage for all
flats");
    }
    Vector<Integer> result;

    if (iSortedFlatNumberList.size() < iFlatCount) {
        result = new Vector<>(iSortedFlatNumberList.subList(0,
iSortedFlatNumberList.size()));
    }
    else {
        result = new Vector<>(iSortedFlatNumberList.subList(0, iFlatCount));
    }

    int iResultLen = result.size();
    for (int i = 0; i < iResultLen - 1; i++) {
        for (int j = i + 1; j < iResultLen; j++) {
            if (iFlatsExpenses.elementAt(i) > iFlatsExpenses.elementAt(j)) {

                double tempExpenses = iFlatsExpenses.elementAt(i);
                iFlatsExpenses.setElementAt(iFlatsExpenses.elementAt(j), i);
                iFlatsExpenses.setElementAt(tempExpenses, j);

                int tempFlatIndex = iSortedFlatNumberList.elementAt(i);

```



```

iSortedFlatNumberList.setElementAt(iSortedFlatNumberList.elementAt(j), i);
        iSortedFlatNumberList.setElementAt(tempFlatIndex, j);
    }
}

return result;
}

//функція для очищення всіх даних
public void ClearInfo(){
    for (int i = 0; i < rpYearsPrices.length; i++) {
        rpYearsPrices[i].setYearPrices(0,0);
    }
    for (int i = 0; i < fFlats.length ; i++) {
        fFlats[i].setSurname("Unknown");
        for (int j = 0; j < rpYearsPrices.length; j++) {
            for (int k = 0; k < 12; k++) {
                fFlats[i].setFlatExpenses(j, k, -1 ,-1);
            }
        }
    }
}

//функція для підрахунку агрегованих витрат помісячно
public double[][] calculateMonthExpences(int iYear){
    double[][] result = new double[12][2];

    double[] totalMonthExpences = new double[2];
    totalMonthExpences[0]= -1;
    totalMonthExpences[1]= -1;
    int iFlag =0;
    for (int i = 0; i < 12; i++) {
        for (int j = 0; j < fFlats.length; j++) {
            double[] FlatExpences = fFlats[j].calculateMonthExpences(iYear,
i, rpYearsPrices[iYear]);
            if(FlatExpences[0] != -1 && FlatExpences[1] != -1){
                iFlag++;
                if(iFlag == 1){
                    totalMonthExpences[0] = 0;
                    totalMonthExpences[1] = 0;
                }
                totalMonthExpences[0] += FlatExpences[0];
                totalMonthExpences[1] += FlatExpences[1];
            }
        }
        result[i][0] = totalMonthExpences[0];
        result[i][1] = totalMonthExpences[1];
        totalMonthExpences[0]= -1;
        totalMonthExpences[1]= -1;
    }

    return result;
}
}

```

Flat.java

```

import java.time.Year;

public class Flat {
    //змінні класу Flat
    private String sSurname;
    private int iFlatNumber;
    private ResourcesUse[][] ruFlatExpenses;
}

```

```

//конструктор за замовчуванням класу Flat
Flat(){
    sSurname = "Unknown";
    this.iFlatNumber = 0;
}
//конструктор з параметрами класу Flat
Flat(int FlatNumber,int StartYear){
    sSurname = "Unknown";
    this.iFlatNumber = FlatNumber;
    ruFlatExpenses = new ResourcesUse[Year.now().getValue() - StartYear +
1][12];
    for (int i = 0; i < Year.now().getValue() - StartYear + 1; i++) {
        for (int j = 0; j < 12; j++) {
            ruFlatExpenses[i][j] = new ResourcesUse();
        }
    }
}
//конструктор копій класу Flat
public Flat(Flat fFlatInfo) {
    this.iFlatNumber = fFlatInfo.getFlatNumber();
    this.sSurname = fFlatInfo.sSurname;

    this.ruFlatExpenses = new
ResourcesUse[fFlatInfo.ruFlatExpenses.length][];
    for (int i = 0; i < fFlatInfo.ruFlatExpenses.length; i++) {
        this.ruFlatExpenses[i] = new
ResourcesUse[fFlatInfo.ruFlatExpenses[i].length];
        for (int j = 0; j < fFlatInfo.ruFlatExpenses[i].length; j++) {
            this.ruFlatExpenses[i][j] = new
ResourcesUse(fFlatInfo.ruFlatExpenses[i][j]);
        }
    }
}

//геттери класу Flat
public String getSurname(){
    return this.sSurname;
}
public int getFlatNumber() {
    return this.iFlatNumber;
}
public ResourcesUse getFlatExpenses(int iYear, int iMonth){
    return this.ruFlatExpenses[iYear][iMonth];
}

//сеттери класу Flat
public void setSurname(String Surname) {
    this.sSurname = Surname;
}
public void setFlatExpenses(int iYear, int iMonth, double WaterUse, double
LightUse) {
    this.ruFlatExpenses[iYear][iMonth].setWaterUse(WaterUse);
    this.ruFlatExpenses[iYear][iMonth].setLightUse(LightUse);
}

//функція для підрахунку витрат за місяць
public double[] calculateMonthExpenses(int iYear,int iMonth, ResourcesPrice
rpPriceForYear) {
    if(rpPriceForYear.getLightPrice() == 0 || rpPriceForYear.getWaterPrice()
== 0){

```

```

        throw new RuntimeException("You haven't entered prices for year");
    }
    double waterExpenses = -1;
    double lightExpenses = -1;

    if (ruFlatExpenses[iYear][iMonth].getWaterUse() != -1 &&
ruFlatExpenses[iYear][iMonth].getLightUse() != -1) {
        waterExpenses = ruFlatExpenses[iYear][iMonth].getWaterUse() *
rpPriceForYear.getWaterPrice();
        lightExpenses = ruFlatExpenses[iYear][iMonth].getLightUse() *
rpPriceForYear.getLightPrice();
    }

    return new double[]{waterExpenses, lightExpenses};
}
//функція для підрахунку витрат за рік
public double[] calculateYearExpenses(int iYear, ResourcesPrice
rpPriceForYear) {
    double waterYearExpenses = 0;
    double lightYearExpenses = 0;
    int iSkipCount = 0;

    for (int i = 0; i < 12; i++) {
        double[] monthExpenses = calculateMonthExpenses(iYear, i,
rpPriceForYear);
        if(monthExpenses[0] == -1 && monthExpenses[1] == -1){
            iSkipCount++;
            continue;
        }
        waterYearExpenses += monthExpenses[0];
        lightYearExpenses += monthExpenses[1];
    }
    if (iSkipCount == 12){
        return new double[]{-1, -1};
    }

    return new double[]{waterYearExpenses, lightYearExpenses};
}
//функція для підрахунку витрат за всі роки
public double[] calculateTotalExpenses(ResourcesPrice[] rpYearsPrices) {
    double totalWaterExpenses = 0;
    double totalLightExpenses = 0;
    int iSkipCount = 0;

    for (int i = 0; i < ruFlatExpenses.length; i++) {
        double[] yearExpenses = calculateYearExpenses(i, rpYearsPrices[i]);

        if(yearExpenses[0]==-1 && yearExpenses[1] == -1){
            iSkipCount++;
            continue;
        }
        totalWaterExpenses += yearExpenses[0];
        totalLightExpenses += yearExpenses[1];
    }
    if (iSkipCount == ruFlatExpenses.length){
        return new double[]{-1, -1};
    }
    return new double[]{totalWaterExpenses, totalLightExpenses};
}

@Override
public String toString() {
    return "Flat {" +
        "sSurname='" + sSurname + '\'' +
        ", iFlatNumber=" + iFlatNumber +
        ", ruFlatExpenses=" + Arrays.toString(ruFlatExpenses) +

```

```

        '}}';
    }
    //функція для перевірки чи заповнена інформація для даного місяця
    public boolean isMonthFilled(int iYear, int iMonth) {

        ResourcesUse flatExpenses = this.getFlatExpenses(iYear, iMonth);
        return flatExpenses.getWaterUse() != -1 && flatExpenses.getLightUse() !=
-1;
    }

}

```

ResourcesPrice.java

```

public class ResourcesPrice{
    //змінні класу ResourcesPrice
    private double lfWaterPrice;
    private double lfLightPrice;

    // конструктор за замовчуванням класу ResourcesPrice
    public ResourcesPrice() {
        this.lfLightPrice = 0;
        this.lfWaterPrice = 0;
    }
    // конструктор з параметрами класу ResourcesPrice
    public ResourcesPrice(double lfWaterPrice, double lfLightPrice) {
        this.lfWaterPrice = lfWaterPrice;
        this.lfLightPrice = lfLightPrice;
    }
    // конструктор копіювання класу ResourcesPrice
    public ResourcesPrice(ResourcesPrice resourcesPriceToCopy) {
        this.lfWaterPrice = resourcesPriceToCopy.lfWaterPrice;
        this.lfLightPrice = resourcesPriceToCopy.lfLightPrice;
    }

    //геттери класу ResourcesPrice
    public double getLightPrice() {
        return lfLightPrice;
    }
    public double getWaterPrice() {
        return lfWaterPrice;
    }
    public ResourcesPrice getYearPrices(){
        return this;
    }

    //сеттери класу ResourcesPrice
    public void setLightPrice(double lfLightPrice) {
        this.lfLightPrice = lfLightPrice;
    }
    public void setWaterPrice(double lfWaterPrice) {
        this.lfWaterPrice = lfWaterPrice;
    }
    public void setYearPrices(double lfWaterPrice, double lfLightPrice){
        this.lfWaterPrice = lfWaterPrice;
        this.lfLightPrice = lfLightPrice;}
}

```

ResourcesUse.java

```

public class ResourcesUse {
    //змінні класу ResourcesUse6

```

```

private double iWaterUse;
private double iLightUse;

//коструктор за замовчуванням класу ResourcesUse
public ResourcesUse(){
    this.iWaterUse = -1;
    this.iLightUse = -1;
}

//коструктор з параметрами класу ResourcesUse
public ResourcesUse(double WaterUse, double LightUse){
    this.iWaterUse = WaterUse;
    this.iLightUse = LightUse;
}

//коструктор копій класу ResourcesUse
public ResourcesUse(ResourcesUse ruFlatExpen) {
    this.iWaterUse = ruFlatExpen.getWaterUse();
    this.iLightUse = ruFlatExpen.getLightUse();
}

//геттери класу ResourcesUse
public double getWaterUse() {
    return iWaterUse;
}
public double getLightUse() {
    return iLightUse;
}

//сеттери класу ResourcesUse
public void setWaterUse(double iWaterUse) {
    this.iWaterUse = iWaterUse;
}
public void setLightUse(double iLightUse) {
    this.iLightUse = iLightUse;
}
}

```

AddInfo.java

```

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.FileReader;

public class AddFlatInfo extends JPanel{
    //елементи інтерфейсу (змінні класу AddFlatInfo)
    private JComboBox cmbYear;
    private JComboBox cmbMonth;
    private JTextField tfSurname;
    private JComboBox cmbFlatNumber;
    private JTextField tfWaterUse;
    private JTextField tfLightUse;
    private JButton jbAddInfo;
    private JButton jbReadFromFile;
    private JPanel AddInfo;

    //геттери класу AddFlatInfo
    public JButton getJbReadFromFile() {
        return jbReadFromFile;
    }
}

```

```

public JButton getJbAddInfo() {
    return jbAddInfo;
}
public JComboBox getCmbYear() {
    return cmbYear;
}
public JComboBox getCmbMonth() {
    return cmbMonth;
}
public JTextField getTfSurname() {
    return tfSurname;
}
public JComboBox getCmbFlatNumber() {
    return cmbFlatNumber;
}
public JTextField getTfWaterUse() {
    return tfWaterUse;
}
public JTextField getTfLightUse() {
    return tfLightUse;
}

//сеттери класу AddFlatInfo
public void setCmbYear(int[] iYears) {
    for (int i = 0; i < iYears.length; i++) {
        this.cmbYear.addItem(iYears[i]);
    }
}
public void setCmbFlatNumber(int iFlatsNumber) {
    for (int i = 0; i < iFlatsNumber; i++) {
        cmbFlatNumber.addItem("Flat №"+(i+1));
    }
}
}

```

ChooseFunction.java

```

import javax.swing.*.*;

public class ChooseFunction extends JPanel{
    //елементи інтерфейсу (змінні класу ChooseFunction)
    private JButton jbOutputFile;
    private JButton jbDisplaySavers;
    private JButton jbDisplayRichPerson;
    private JTextField tfWaterPrice;
    private JTextField tfLightPrice;
    private JButton jbSetPrice;
    private JButton jbShowNotFilledInfo;
    private JButton jbMonthsExpences;
    private JComboBox cmbYear;
    private JButton jbClearInfo;
    private JPanel Function;

    //геттери класу ChooseFunction

    public JButton getJbMonthsExpences() {
        return jbMonthsExpences;
    }
    public JButton getJbClearInfo() {
        return jbClearInfo;
    }
    public JTextField getTfWaterPrice() {

```

```

        return tfWaterPrice;
    }
    public JTextField getTfLightPrice() {
        return tfLightPrice;
    }
    public JButton getJbDisplaySavers() {
        return jbDisplaySavers;
    }
    public JButton getJbDisplayRichPerson() {
        return jbDisplayRichPerson;
    }
    public JButton getJbSetPrice() {
        return jbSetPrice;
    }
    public JButton getJbShowNotFilledInfo() {
        return jbShowNotFilledInfo;
    }
    public JButton getJbOutputFile() {return jbOutputFile;}
    public JComboBox getCmbYear() {
        return cmbYear;
    }
}

//сеттери класу ChooseFunction
public void setCmbYear(int[] iYears) {
    for (int i = 0; i < iYears.length; i++) {
        this.cmbYear.addItem(iYears[i]);
    }
}
}

```

MainApplicationWindow.java

```

import javax.swing.*.*;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.io.File;
import java.time.Year;
import java.util.ArrayList;
import java.util.Vector;

public class MainApplicationWindow extends JFrame {

    //функціональні елементи
    final private int iStartYear;
    final private int iFlatsNumber;
    private final Building bApartment;

    //елементи інтерфейсу (змінні класу MainApplicationWindow)
    private JTable jtFlatShowInfo;
    private final DefaultTableModel dtbTableInfo;
    private JPanel MainApplicationWindow;
    private JPanel jpControlItem;
    private AddFlatInfo AddInfoItem;
    private ChooseFunction FunctionsManager;
    private JComboBox cmbTableYear;
    private JComboBox cmbTableMonth;
    private JComboBox cmbFunctionMenu;
    private final String sFuncOutput =

```

```

"Q:\\Anton\\Education\\Kyrsova\\CourseWork\\FuncOutput.txt";
private JPanel ControlBlock;

//конструктор класу MainApplicationWindow
MainApplicationWindow(int FlatsNumber, int StartYear){

    this.iFlatsNumber= FlatsNumber;
    this.iStartYear= StartYear;
    bApartment = new Building(iFlatsNumber,iStartYear);

    //відцентрування вікна
    Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
    this.setSize(900, 625);
    int x = (int) ((dimension.getWidth() - this.getWidth()) / 2);
    int y = (int) ((dimension.getHeight() - this.getHeight()) / 2);
    this.setLocation(x, y);

    //дефолтні налаштування вікна
    this.setContentPane(this.MainApplicationWindow);
    this.setTitle("Flat consumption manager");
    this.setVisible(true);
    this.setDefaultCloseOperation(EXIT_ON_CLOSE);

    //Заповнення комбобоксів
    int[] iYears = new int[Year.now().getValue() - iStartYear + 1];
    for (int i = 0; i < iYears.length; i++) {
        iYears[i] = iStartYear + i;
    }

    for (int iYear : iYears) {
        cmbTableYear.addItem(iYear);
    }

    //Вказання змісту comboBox для блоку AddInfoItem

    AddInfoItem.setCmbYear(iYears);

    AddInfoItem.setCmbFlatNumber(iFlatsNumber);

    //Вказання змісту comboBox для блоку FunctionManager

    FunctionsManager.setCmbYear(iYears);

    //створення основної таблиці
    String[] sColumnsNames = {"№","Surname","Water use","Light use"};
    dtbTableInfo = new
DefaultTableModel(ApplicationFunc.SetTableInfo(bApartment,cmbTableYear.getSelectedIndex(), cmbTableMonth.getSelectedIndex()),sColumnsNames){
        @Override
        public boolean isCellEditable(int row, int column) {
            //all cells false
            return false;
        }
    };
    jtFlatShowInfo.setModel(dtbTableInfo);

    //listener для обрання типу меню
    cmbFunctionMenu.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            CardLayout cl = (CardLayout) jpControlItem.getLayout();
            if(cmbFunctionMenu.getSelectedIndex() == 0) {
                cl.show(jpControlItem, "Card1");
            }
        }
    });
}

```



```

        } else if (cmbFunctionMenu.getSelectedIndex() == 1) {
            cl.show(jpControlItem, "Card2");
        }
    }
});

//listener для сканування з файлу
AddInfoItem.getJbReadFromFile().addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        try{
            final JFileChooser fc = new
JFileChooser("Q:\\Anton\\Education\\Kyrsova\\CourseWork\\files\\");
            fc.showOpenDialog(null);

            File selFile = fc.getSelectedFile();
            if (selFile == null) {
                throw new RuntimeException("Choose file");
            }

            String sFileName = selFile.getName();
            String[] fileNameParts = sFileName.split("\\.");

            int iCurInputYear = Integer.parseInt(fileNameParts[0]);

            if (iCurInputYear < iStartYear || iCurInputYear
>Year.now().getValue()){
                throw new RuntimeException("Selected file isn't included
to accounting period");
            }

ApplicationFunc.ReadFlatsInfoFromFile(bApartment, iCurInputYear - iStartYear,
selFile);

AddInfoItem.getTfSurname().setText(bApartment.getSurname(AddInfoItem.getCmbFlatN
umber().getSelectedIndex()));
            if(AddInfoItem.getCmbMonth().getSelectedIndex() ==
cmbTableMonth.getSelectedIndex() &&
                AddInfoItem.getCmbYear().getSelectedIndex() ==
cmbTableYear.getSelectedIndex()){
                ApplicationFunc.UpdateTextFields(bApartment,

AddInfoItem.getCmbFlatNumber().getSelectedIndex(),

cmbTableYear.getSelectedIndex(),

cmbTableMonth.getSelectedIndex(),

AddInfoItem.getTfWaterUse(),

AddInfoItem.getTfLightUse()

);
            }

//заповнення полів з ціною та перемикання комбобоксу на рік
зчитування
            ResourcesPrice rpYearPrice =
bApartment.getYearPrices(iCurInputYear - iStartYear);

FunctionsManager.getCmbYear().setSelectedItem(iCurInputYear);

FunctionsManager.getTfWaterPrice().setText(String.valueOf(rpYearPrice.getWaterPr
ice()));

```

```

FunctionsManager.getTfLightPrice().setText(String.valueOf(rpYearPrice.getLightPr
ice()));

        //оновлення таблиці
        cmbTableYear.setSelectedItem(iCurInputYear);

dtbTableInfo.setDataVector(ApplicationFunc.SetTableInfo(bApartment, cmbTableYear.
getSelectedIndex(), cmbTableMonth.getSelectedIndex(), sColumnsNames);

        //вивід у файл

        String sLogBody = "File Selected: " + selFile.getName() +
"\n" +
                "Input Year: " + iCurInputYear + "\n" +
                "Data Successfully Read and Updated\n";

ApplicationFunc.OutputToLogFile(sFuncOutput, "ReadFromFile", sLogBody);

        }
        catch (NumberFormatException exc) {
            JOptionPane.showMessageDialog(null, "Selected file is
incorrect or contains incorrect data", "Warning:\\", JOptionPane.WARNING_MESSAGE);
        }
        catch (RuntimeException exc){

JOptionPane.showMessageDialog(null, exc.getMessage(), "Warning:\\", JOptionPane.WAR
NING_MESSAGE);
        }
        catch (Exception exc){

JOptionPane.showMessageDialog(null, exc.getMessage(), "Error: (", JOptionPane.ERROR_
MESSAGE);
        }
    }
});
//listener для додавання витрат
AddInfoItem.getJbAddInfo().addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try{
            // перевірки на коректність вводу
            if (AddInfoItem.getTfSurname().getText().equals("")) {
                throw new RuntimeException("Enter surname");
            }
            if (AddInfoItem.getTfWaterUse().getText().equals("")) {
                throw new RuntimeException("Enter water use value");
            }
            if (AddInfoItem.getTfLightUse().getText().equals("")) {
                throw new RuntimeException("Enter light use value");
            }
            String surname = AddInfoItem.getTfSurname().getText();

            surname = surname.replaceAll("\\s", "");
            String regex = "[a-zA-Z]+$";

            if (!surname.matches(regex)) {
                throw new RuntimeException("Enter a valid
surname\nEnglish letters only");
            }

            double waterUse =
Double.parseDouble(AddInfoItem.getTfWaterUse().getText());

```

```

        double lightUse =
Double.parseDouble(AddInfoItem.getTfLightUse().getText());

        // збереження даних

bApartment.setFlatExpences(AddInfoItem.getCmbFlatNumber().getSelectedIndex(),
AddInfoItem.getCmbYear().getSelectedIndex(),
AddInfoItem.getCmbMonth().getSelectedIndex(),
                                waterUse,
                                lightUse);

bApartment.setSurname(AddInfoItem.getCmbFlatNumber().getSelectedIndex(),
                                surname);

        //перемикання комбобоксів року та місяця на зміну
інформації

cmbTableYear.setSelectedItem(AddInfoItem.getCmbYear().getSelectedItem());
cmbTableMonth.setSelectedItem(AddInfoItem.getCmbMonth().getSelectedItem());

        //оновлення таблиці

dtbTableInfo.setDataVector(ApplicationFunc.SetTableInfo(bApartment, cmbTableYear.
getSelectedIndex(), cmbTableMonth.getSelectedIndex(), sColumnsNames);

        //вивід у файл

        String sLogBody = "Surname: " +
AddInfoItem.getTfSurname().getText() + "\n" +
                                "Water Use: " +
AddInfoItem.getTfWaterUse().getText() + "\n" +
                                "Light Use: " +
AddInfoItem.getTfLightUse().getText() + "\n" +
                                "Flat Number: " +
(AddInfoItem.getCmbFlatNumber().getSelectedIndex()+1) + "\n" +
                                "Year: " +
AddInfoItem.getCmbYear().getSelectedItem() + "\n" +
                                "Month: " +
AddInfoItem.getCmbMonth().getSelectedItem() + "\n" +
                                "Processed: Data successfully added\n";

ApplicationFunc.OutputToLogFile(sFuncOutput, "AddInformation", sLogBody);

        }
        catch (NumberFormatException exc) {
            JOptionPane.showMessageDialog(null, "You entered wrong value
of water or light use", "Warning:\\", JOptionPane.WARNING_MESSAGE);
        }
        catch (RuntimeException exc){

JOptionPane.showMessageDialog(null, exc.getMessage(), "Warning:\\", JOptionPane.WAR
NING_MESSAGE);
        }
        catch (Exception exc){

JOptionPane.showMessageDialog(null, exc.getMessage(), "Error: (", JOptionPane.ERROR_
MESSAGE);
        }
    }
});
//listener для оновлення полів вводу витрат та прізвища при перемиканні

```

комбобоксу номера квартири

```
        AddInfoItem.getCmbFlatNumber().addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                int iSelectedFlat =
AddInfoItem.getCmbFlatNumber().getSelectedIndex();
                String SurnameOfSelectedFlat =
bApartment.getSurname(iSelectedFlat);
                if(!SurnameOfSelectedFlat.equals("Unknown")) {
                    AddInfoItem.getTfSurname().setText(SurnameOfSelectedFlat);
                }
                else{
                    AddInfoItem.getTfSurname().setText("");
                }
                int iSelectedMonth =
AddInfoItem.getCmbMonth().getSelectedIndex();
                int iSelectedYear = AddInfoItem.getCmbYear().getSelectedIndex();

ApplicationFunc.UpdateTextFields(bApartment,iSelectedFlat,iSelectedYear,
iSelectedMonth, AddInfoItem.getTfWaterUse(),AddInfoItem.getTfLightUse());
            }
        });
```

//listener для оновлення полів вводу витрат та прізвища при перемиканні комбобоксу року

```
        AddInfoItem.getCmbYear().addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                int iSelectedFlat =
AddInfoItem.getCmbFlatNumber().getSelectedIndex();
                int iSelectedMonth =
AddInfoItem.getCmbMonth().getSelectedIndex();
                int iSelectedYear = AddInfoItem.getCmbYear().getSelectedIndex();

ApplicationFunc.UpdateTextFields(bApartment,iSelectedFlat,iSelectedYear,
iSelectedMonth, AddInfoItem.getTfWaterUse(),AddInfoItem.getTfLightUse());
            }
        });
```

//listener для оновлення полів вводу витрат та прізвища при перемиканні комбобоксу місяця

```
        AddInfoItem.getCmbMonth().addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                int iSelectedFlat =
AddInfoItem.getCmbFlatNumber().getSelectedIndex();
                int iSelectedMonth =
AddInfoItem.getCmbMonth().getSelectedIndex();
                int iSelectedYear = AddInfoItem.getCmbYear().getSelectedIndex();

ApplicationFunc.UpdateTextFields(bApartment,iSelectedFlat,iSelectedYear,
iSelectedMonth, AddInfoItem.getTfWaterUse(),AddInfoItem.getTfLightUse());
            }
        });
```

//listener для виводу номеру квартири, яка спожила найбільше послуг в грошовому еквіваленті

```
        FunctionsManager.getJbDisplayRichPerson().addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    ArrayList<Object[]> maxFlatInfoList =
bApartment.findFlatsWithHighestExpenses();
```

```

        if (!maxFlatInfoList.isEmpty()) {
            StringBuilder message = new StringBuilder("Flats with
the highest expenses:\n");

            double waterExpenses = 0;
            double lightExpenses = 0;

            for (Object[] flatInfo : maxFlatInfoList) {
                int flatNumber = (int) flatInfo[0];
                waterExpenses = (double) flatInfo[1];
                lightExpenses = (double) flatInfo[2];

                message.append("Flat ").append(flatNumber)
                    .append(String.format(", Water Expenses:
%.2f", waterExpenses))
                    .append(String.format(", Light Expenses:
%.2f", lightExpenses)).append("\n");
            }
            message.append(String.format("Total expenses:
%.2f\n", waterExpenses + lightExpenses));

            JOptionPane.showMessageDialog(null, message.toString(),
"Highest Expenses", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "No flats found with
expenses.", "Highest Expenses", JOptionPane.INFORMATION_MESSAGE);
        }
        //вивід у файл

        String sLogBody = "Flats with the highest expenses:\n";

        double totalWaterExpenses = 0;
        double totalLightExpenses = 0;

        for (Object[] flatInfo : maxFlatInfoList) {
            int flatNumber = (int) flatInfo[0];
            double waterExpenses = (double) flatInfo[1];
            double lightExpenses = (double) flatInfo[2];

            sLogBody += String.format("Flat %d, Water Expenses:
%.2f, Light Expenses: %.2f\n", flatNumber, waterExpenses, lightExpenses);

            totalWaterExpenses += waterExpenses;
            totalLightExpenses += lightExpenses;
        }

        sLogBody += String.format("Total expenses: %.2f\n",
totalWaterExpenses + totalLightExpenses);

        ApplicationFunc.OutputToLogFile(sFuncOutput, "DisplayRichPerson", sLogBody);
    }
    catch (RuntimeException exc) {

        JOptionPane.showMessageDialog(null, exc.getMessage(), "Warning:\\", JOptionPane.WAR
NING_MESSAGE);
    }
    catch (Exception exc) {

        JOptionPane.showMessageDialog(null, exc.getMessage(), "Error: (", JOptionPane.ERROR_
MESSAGE);
    }
}

});
//listener для виводу інформації за обраний рік у файл
FunctionsManager.getJbOutputFile().addActionListener(new

```

```

ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int iSelectedYear =
(int)FunctionsManager.getCmbYear().getSelectedItem();

ApplicationFunc.OutputFlatsInfoToFile(bApartment,iSelectedYear,iStartYear);

        //вивід у файл

        String sLogBody = "Flat information successfully output to
file";

ApplicationFunc.OutputToLogFile(sFuncOutput,"OutputToFile",sLogBody);

    }
});
//listener для встановлення ціни в межах року
FunctionsManager.getJbSetPrice().addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {

        try{
            if(FunctionsManager.getTfWaterPrice().getText().equals("")){
                throw new RuntimeException("Enter water price value");
            }
            if(FunctionsManager.getTfLightPrice().getText().equals("")){
                throw new RuntimeException("Enter light price value");
            }
            double waterPrice =
Double.parseDouble(FunctionsManager.getTfWaterPrice().getText());
            double lightPrice =
Double.parseDouble(FunctionsManager.getTfLightPrice().getText());

bApartment.setYearPrices(FunctionsManager.getCmbYear().getSelectedIndex(),
waterPrice, lightPrice);
            JOptionPane.showMessageDialog(null,"Price successfully
set");

            //вивід у файл
            String sLogBody = String.format("Water price set to %.2f,
Light price set to %.2f", waterPrice, lightPrice);

            ApplicationFunc.OutputToLogFile(sFuncOutput, "SetPrice",
sLogBody);

        }
        catch (NumberFormatException exc) {
            JOptionPane.showMessageDialog(null,"You entered wrong value
of water or light price","Warning:\\",JOptionPane.WARNING_MESSAGE);
        }
        catch (RuntimeException exc){

JOptionPane.showMessageDialog(null,exc.getMessage(),"Warning:\\",JOptionPane.WAR
NING_MESSAGE);
        }
        catch(Exception exc){

JOptionPane.showMessageDialog(null,exc.getMessage(),"Error:(",JOptionPane.ERROR_
MESSAGE);
        }

    }
}
}

```

```

});
//listener для оновлення полів для вказання ціни
FunctionsManager.getCmbYear().addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        int iSelectedYear =
FunctionsManager.getCmbYear().getSelectedIndex();
        ResourcesPrice rpWLPrise =
bApartment.getYearPrices(iSelectedYear);

        if(!(rpWLPrise.getLightPrice() == 0 || rpWLPrise.getWaterPrice()
== 0)){

FunctionsManager.getTfLightPrice().setText(String.valueOf(rpWLPrise.getLightPrice()));

FunctionsManager.getTfWaterPrice().setText(String.valueOf(rpWLPrise.getWaterPrice()));

        }
        else {
            FunctionsManager.getTfLightPrice().setText("");
            FunctionsManager.getTfWaterPrice().setText("");
        }
    }
});
//listener для відображення місяців для яких інформація не заповнена
FunctionsManager.getJbShowNotFilledInfo().addActionListener(new
ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            int iSelectedYear =
FunctionsManager.getCmbYear().getSelectedIndex();

            Vector result[] =
bApartment.OutputNotFilledMonths(iSelectedYear);

            String sLogBody = "Not filled info for selected year:\n";
            String[] ShowInfoColumns= {"Month","Not filled info"};
            Object[][] oTableData = new Object[12][2];

            //заповнення інформації для таблиці
            for (int i = 0; i < oTableData.length; i++) {
                sLogBody += String.format("Month: %s, Not filled info:",
cmbTableMonth.getItemAt(i));

                oTableData[i][0] = cmbTableMonth.getItemAt(i);
                oTableData[i][1] = "";
                if (-1 == (int)result[i].get(0)) {
                    sLogBody += "for each flats";
                    oTableData[i][1] = "for each flats";
                }
                else if(0 == (int)result[i].get(0)) {
                    sLogBody += "for any flats";
                    oTableData[i][1] = "for any flats";
                }
                else{
                    for (int j = 0; j < result[i].size(); j++) {
                        oTableData[i][1] +=
String.valueOf(result[i].get(j))+" ";
                        sLogBody += String.valueOf(result[i].get(j))+" "
;

                    }
                }
                sLogBody += "\n";
            }
        }
    }
});

```

```

        DefaultTableModel dtbShowInfo = new
DefaultTableModel(oTableData, ShowInfoColumns) {
    @Override
    public boolean isCellEditable(int row, int column) {
        //all cells false
        return false;
    }
};
ShowInfo showInfo = new ShowInfo(dtbShowInfo, 1);

//зміна кольорів елементів таблиці відповідно до їх вмісту
DefaultTableCellRenderer renderer = new
DefaultTableCellRenderer() {
    @Override
    public Component getTableCellRendererComponent(JTable
table, Object value, boolean isSelected, boolean hasFocus, int row, int column)
{
        Component c =
super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row,
column);

        Object column1Value = table.getValueAt(row, 1);

        // Змінити фон та текст в залежності від значень
        if (column1Value instanceof String) {
            String notFilledInfo = (String) column1Value;

            // Змінити колір фону в залежності від значень
            if ("for each flats".equals(notFilledInfo)) {
                c.setBackground(new Color(205, 92, 92));
                c.setForeground(Color.BLACK);
            } else if ("for any
flats".equals(notFilledInfo)) {
                c.setBackground(new Color(60, 179, 113));
                c.setForeground(Color.BLACK);
            } else {
                c.setBackground(new Color(240, 230, 140));
                c.setForeground(Color.BLACK);
            }
        }
        return c;
    }
};
// Задаємо рендерер для стовпця

showInfo.getJtInfo().getColumnModel().getColumn(1).setCellRenderer(renderer);

//вивід у файл

ApplicationFunc.OutputToLogFile(sFuncOutput,
"ShowNotFilledInfo", sLogBody);

}
catch(Exception exc){

JOptionPane.showMessageDialog(null, exc.getMessage(), "Error: (", JOptionPane.ERROR_
MESSAGE);

}

});
//listener для відображення жителів , які спожили найменше послуг за рік
відсортованих за прізвищем

```



```

        FunctionsManager.getJbDisplaySavers().addActionListener(new
ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try{
            int iSelectedYear =
FunctionsManager.getCmbYear().getSelectedIndex();

            //вивід вікна для вибору к-сті людей для відображення
            String[] sChoosePeople = new String[iFlatsNumber];
            for (int i = 0; i < iFlatsNumber; i++) {
                sChoosePeople[i] = "" + (i + 1) + " people";
            }
            JComboBox cmbChoosePeople = new JComboBox(sChoosePeople);

            int iAnswer = JOptionPane.showConfirmDialog(null,
cmbChoosePeople, "How many people you want to display: ",
JOptionPane.DEFAULT_OPTION);

            int iSeversCount = cmbChoosePeople.getSelectedIndex() + 1;

            if (iAnswer == JOptionPane.OK_OPTION) {

                String sLogBody = "Displayed savers information:\n";
                Vector <Integer> vflatNumbers =
bApartment.getSavers(iSeversCount, iSelectedYear);
                int[] iFlatNumbers = new int[vflatNumbers.size()];
                String[] sSurname = new String[iSeversCount];
                Object[][] oTableData = new
Object[vflatNumbers.size()][4];
                for (int i = 0; i < iSeversCount; i++) {
                    iFlatNumbers[i] = vflatNumbers.elementAt(i);
                    sSurname[i] =
bApartment.getSurname(vflatNumbers.elementAt(i));
                }

                ApplicationFunc.shellSort(sSurname, iFlatNumbers);

                //заповнення даних таблиці
                for (int i = 0; i < vflatNumbers.size(); i++) {

                    oTableData[i][0] = sSurname[i];
                    double[] FlatExpences =
bApartment.getFlatYearExpences(iFlatNumbers[i], iSelectedYear);
                    sLogBody += String.format("Surname: %s, Water price:
%.2f, Light price: %.2f, Total price: %.2f\n", sSurname[i], FlatExpences[0],
FlatExpences[1], FlatExpences[0] + FlatExpences[1]);
                    oTableData[i][1] = String.format("%.2f",
FlatExpences[0]);
                    oTableData[i][2] = String.format("%.2f",
FlatExpences[1]);
                    oTableData[i][3] = String.format("%.2f",
FlatExpences[0] + FlatExpences[1]);
                }
                String[] ShowInfoColumns = {"Surname", "Water price",
"Light price", "Total price"};

                DefaultTableModel dtbShowInfo = new
DefaultTableModel(oTableData, ShowInfoColumns){

```

```

        @Override
        public boolean isCellEditable(int row, int column) {
            //all cells false
            return false;
        }
    };
    new ShowInfo(dtbShowInfo, 2);
    if(vflatNumbers.size() < iSeversCount){
        throw new RuntimeException(String.format("You have
only %d entered flat usage, not %d",vflatNumbers.size(), iSeversCount));
    }
    //вивід у файл

        ApplicationFunc.OutputToLogFile(sFuncOutput,
"DisplaySavers", sLogBody);
    }

    }
    catch (RuntimeException exc){

JOptionPane.showMessageDialog(null,exc.getMessage(),"Warning:\\",JOptionPane.WAR
NING_MESSAGE);
    }
    catch(Exception exc){

JOptionPane.showMessageDialog(null,exc.getMessage(),"Error:(",JOptionPane.ERROR_
MESSAGE);
    }

    }
    });
    //listener для відображення агрегованих витрат помісячно
    FunctionsManager.getJbMonthsExpences().addActionListener(new
ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            try{
                int iSelectedYear =
FunctionsManager.getCmbYear().getSelectedIndex();

                String sLogBody = "Calculated monthly expenses for the
selected year:\n";

                double[][] MonthsExpences =
bApartment.calculateMonthExpences(iSelectedYear);

                String[] ShowInfoColumns = {"Month", "Water Expences",
"Light Expences", "Total Expences"};
                Object[][] oTableData = new Object[12][4];

                //заповнення інформації для таблиці
                for (int i = 0; i < oTableData.length; i++) {

                    oTableData[i][0] = cmbTableMonth.getItemAt(i);
                    if(MonthsExpences[i][0] != -1 || MonthsExpences[i][1] !=
-1) {
                        oTableData[i][1] = String.format("%.2f",
MonthsExpences[i][0]);
                        oTableData[i][2] = String.format("%.2f",
MonthsExpences[i][1]);
                        oTableData[i][3] = String.format("%.2f",
MonthsExpences[i][0] + MonthsExpences[i][1]);
                        sLogBody += String.format("Month: %s, Water
Expences: %.2f, Light Expences: %.2f, Total Expences: %.2f\n",

```

```

                                cmbTableMonth.getItemAt(i),
MonthsExpences[i][0], MonthsExpences[i][1], MonthsExpences[i][0] +
MonthsExpences[i][1]);
                                }
                                else{
                                    oTableData[i][1] = "-";
                                    oTableData[i][2] = "-";
                                    oTableData[i][3] = "not filled usage";
                                    sLogBody += String.format("Month: %s, Water
Expenses: -, Light Expenses: -, Total Expenses: Not filled
usage\n", cmbTableMonth.getItemAt(i));
                                }
                                }

                                DefaultTableModel dtbShowInfo = new
DefaultTableModel(oTableData, ShowInfoColumns){
                                @Override
                                public boolean isCellEditable(int row, int column) {
                                    //all cells false
                                    return false;
                                }
                                };
                                ShowInfo showInfo = new ShowInfo(dtbShowInfo, 3);
                                DefaultTableCellRenderer centerRenderer = new
DefaultTableCellRenderer();
                                centerRenderer.setHorizontalAlignment( JLabel.CENTER );

                                showInfo.getJtInfo().getColumnModel().getColumn(1).setCellRenderer(
                                centerRenderer );

                                showInfo.getJtInfo().getColumnModel().getColumn(2).setCellRenderer(
                                centerRenderer );

                                //вивід у файл

                                ApplicationFunc.OutputToLogFile(sFuncOutput,
"MonthsExpences", sLogBody);
                                }
                                catch (RuntimeException exc){

JOptionPane.showMessageDialog(null, exc.getMessage(), "Warning:\\", JOptionPane.WAR
NING_MESSAGE);
                                }
                                catch (Exception exc){

JOptionPane.showMessageDialog(null, exc.getMessage(), "Error:(", JOptionPane.ERROR_
MESSAGE);
                                }
                                }
                                });
                                //listener для очищення всіх даних
                                FunctionsManager.getJbClearInfo().addActionListener(new ActionListener()
{
                                @Override
                                public void actionPerformed(ActionEvent e) {
                                    bApartment.ClearInfo();

                                //оновлення таблиці

                                dtbTableInfo.setDataVector(ApplicationFunc.SetTableInfo(bApartment, cmbTableYear.
getSelectedIndex(), cmbTableMonth.getSelectedIndex(), sColumnsNames);

                                //оновлення полів вводу
                                FunctionsManager.getTfWaterPrice().setText("");
                                FunctionsManager.getTfLightPrice().setText("");
                                AddInfoItem.getTfSurname().setText("");

```

```

        AddInfoItem.getTfWaterUse().setText("");
        AddInfoItem.getTfLightUse().setText("");

        JOptionPane.showMessageDialog(null, "Data successfully cleared");

        //вивід у файл
        String sLogBody = "Data successfully cleared\n";

        ApplicationFunc.OutputToLogFile(sFuncOutput, "MonthsExpences",
sLogBody);
    }
});
//listener для зміни таблиці відповідно до вибору комбобоксу року
cmbTableYear.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        int iSelectedYear= cmbTableYear.getSelectedIndex();
        int iSelectedMonth = cmbTableMonth.getSelectedIndex();

        dtbTableInfo.setDataVector(ApplicationFunc.SetTableInfo(bApartment,iSelectedYear
,iSelectedMonth), sColumnsNames);
    }
});
//listener для зміни таблиці відповідно до вибору комбобоксу місяця
cmbTableMonth.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        int iSelectedYear= cmbTableYear.getSelectedIndex();
        int iSelectedMonth = cmbTableMonth.getSelectedIndex();

        dtbTableInfo.setDataVector(ApplicationFunc.SetTableInfo(bApartment,iSelectedYear
,iSelectedMonth), sColumnsNames);
    }
});
}
}

```

ShowInfo.java

```

import javax.swing.*;

public class ChooseFunction extends JPanel{
    //елементи інтерфейсу (змінні класу ChooseFunction)
    private JButton jbOutputFile;
    private JButton jbDisplaySavers;
    private JButton jbDisplayRichPerson;
    private JTextField tfWaterPrice;
    private JTextField tfLightPrice;
    private JButton jbSetPrice;
    private JButton jbShowNotFilledInfo;
    private JButton jbMonthsExpences;
    private JComboBox cmbYear;
    private JButton jbClearInfo;
    private JPanel Function;

    //геттери класу ChooseFunction

    public JButton getJbMonthsExpences() {
        return jbMonthsExpences;
    }
    public JButton getJbClearInfo() {
        return jbClearInfo;
    }
}

```

```

    }
    public JTextField getTfWaterPrice() {
        return tfWaterPrice;
    }
    public JTextField getTfLightPrice() {
        return tfLightPrice;
    }
    public JButton getJbDisplaySavers() {
        return jbDisplaySavers;
    }
    public JButton getJbDisplayRichPerson() {
        return jbDisplayRichPerson;
    }
    public JButton getJbSetPrice() {
        return jbSetPrice;
    }
    public JButton getJbShowNotFilledInfo() {
        return jbShowNotFilledInfo;
    }
    public JButton getJbOutputFile() {return jbOutputFile;}
    public JComboBox getCmbYear() {
        return cmbYear;
    }
}

//сеттери класу ChooseFunction
public void setCmbYear(int[] iYears) {
    for (int i = 0; i < iYears.length; i++) {
        this.cmbYear.addItem(iYears[i]);
    }
}
}

```

WelcomeWindow.java

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.time.Year;

public class WelcomeWindow extends JFrame {
    //елементи інтерфейсу (змінні класу WelcomeWindow)
    private JTextField tfFlatsNumber;
    private JTextField tfStartYear;
    private JButton jbStart;
    private JPanel WelcomePanel;

    //конструктор класу WelcomeWindow
    WelcomeWindow(){
        //відцентрування вікна при виводі на екран
        Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
        this.setSize(330, 300);
        int x = (int) ((dimension.getWidth() - this.getWidth()) / 2);
        int y = (int) ((dimension.getHeight() - this.getHeight()) / 2);
        this.setLocation(x, y);

        // дефолтні налаштування вікна
        this.setContentPane(this.WelcomePanel);
        this.setTitle("Welcome");
        this.setVisible(true);
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

```

```

// listener для початку роботи
jbStart.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try{
            int iFlatsNumber =
Integer.parseInt(tfFlatsNumber.getText());

            if(!(iFlatsNumber > 1)){
                throw new RuntimeException("Number of flats must be >
1");
            }

            int iStartYear = Integer.parseInt(tfStartYear.getText());

            if(!(iStartYear > 0 && iStartYear <=
Year.now().getValue())){
                throw new RuntimeException("Start year must be between
the first and current year");
            }

            new MainApplicationWindow(iFlatsNumber, iStartYear);
            dispose();
        }
        catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(null,"Please enter correct
values","Warning:\\",JOptionPane.WARNING_MESSAGE);
        }
        catch (RuntimeException exc){

JOptionPane.showMessageDialog(null,exc.getMessage(),"Warning:\\",JOptionPane.WAR
NING_MESSAGE);
        }
    }
});
}
public static void main(String[] args){
    new WelcomeWindow();
}
}

```

4. Протокол роботи програми для кожного пункту завдання

№	Surname	Water use	Light use
1	Unknown	not filled	not filled
2	Unknown	not filled	not filled
3	Unknown	not filled	not filled
4	Unknown	not filled	not filled
5	Unknown	not filled	not filled
6	Unknown	not filled	not filled
7	Unknown	not filled	not filled
8	Unknown	not filled	not filled
9	Unknown	not filled	not filled
10	Unknown	not filled	not filled
11	Unknown	not filled	not filled
12	Unknown	not filled	not filled
13	Unknown	not filled	not filled
14	Unknown	not filled	not filled
15	Unknown	not filled	not filled

Рис. 4.1. Головне вікно програми при запуску

№	Surname	Water use	Light use
1	Unknown	not filled	not filled
2	Unknown	not filled	not filled
3	Legeza	11.4	165.8
4	Unknown	not filled	not filled
5	Unknown	not filled	not filled
6	Unknown	not filled	not filled
7	Unknown	not filled	not filled
8	Unknown	not filled	not filled
9	Unknown	not filled	not filled
10	Unknown	not filled	not filled
11	Unknown	not filled	not filled
12	Unknown	not filled	not filled
13	Unknown	not filled	not filled
14	Unknown	not filled	not filled
15	Unknown	not filled	not filled

Рис. 4.2. Введення даних у таблицю «вручну»

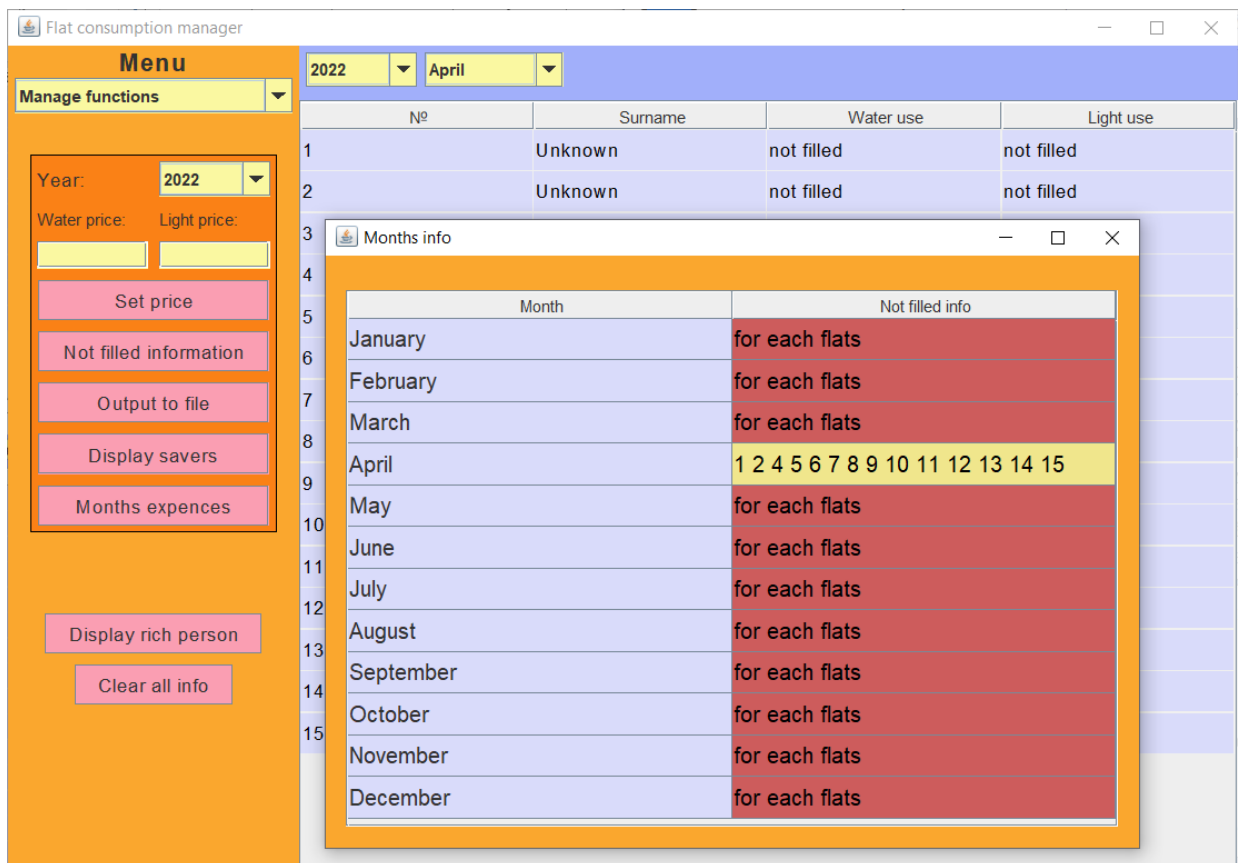


Рис. 4.3. Відображення місяців, для яких інформація не заповнена для кожної з квартир

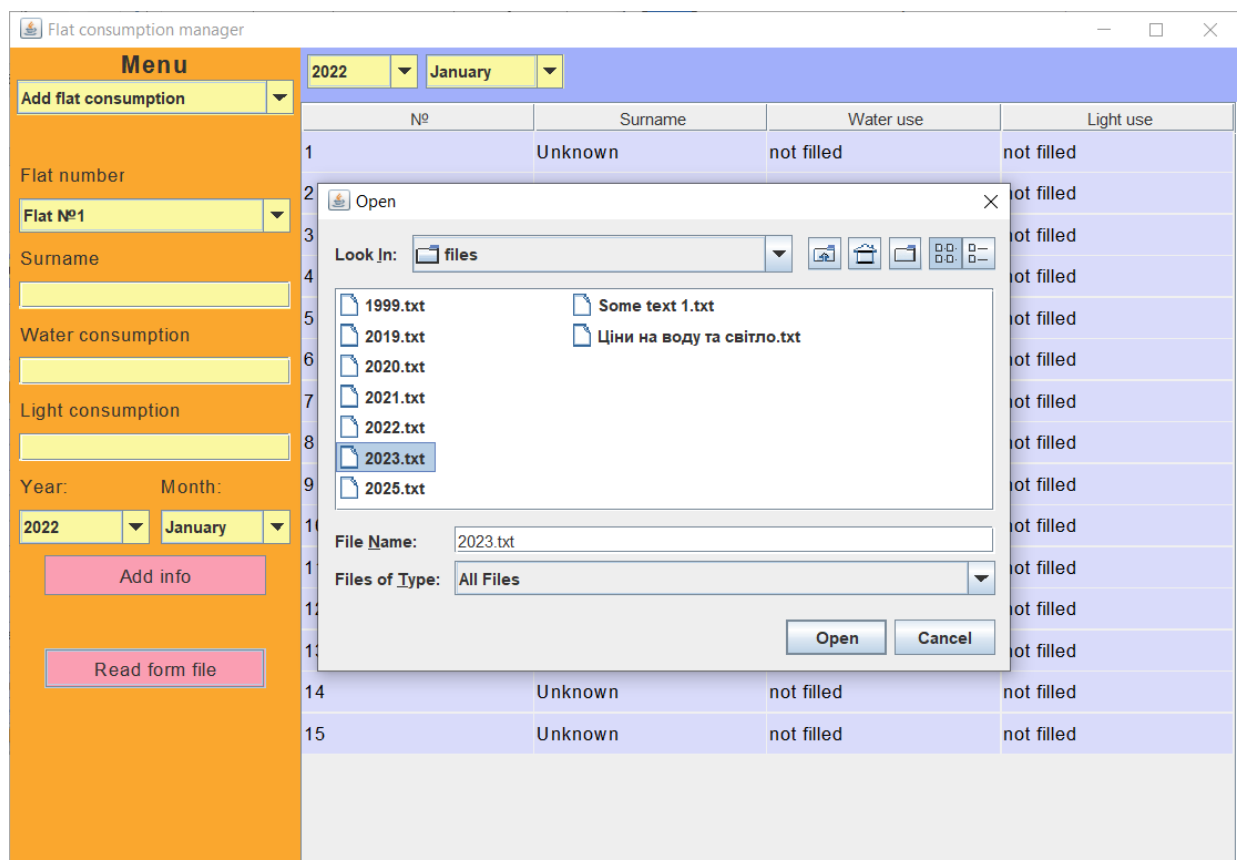


Рис. 4.4. Діалог вибору файлу для імпорту даних

Flat consumption manager

Menu

Add flat consumption

Flat number
Flat №1

Surname
Legeza

Water consumption

Light consumption

Year: 2022 Month: January

Add info

Read form file

№	Surname	Water use	Light use
1	Legeza	11.5	150.0
2	Skakovskiy	10.5	160.0
3	Bodnar	11.3	140.0
4	Lukianov	10.9	150.0
5	Feduniak	11.5	155.0
6	Revus	10.7	165.0
7	Razik	11.2	145.0
8	Bachynskiy	11.7	140.0
9	Vashchuk	10.8	150.0
10	Ovcharenko	11.5	155.0
11	Sakh	10.7	165.0
12	Yaroshovych	11.2	145.0
13	Franchuk	11.7	140.0
14	Debeliak	10.8	150.0
15	Holub	11.0	148.0

Рис. 4.5. Головне вікно програми після імпорту даних

Flat consumption manager

Menu

Manage functions

Year: 2022

Water price: 24.6 Light price: 1.3

Set price

Not filled information

Output to file

Display savers

Months expenses

Display rich person

Clear all info

№	Surname	Water use	Light use
1	Legeza	11.5	150.0
2	Skakovskiy	10.5	160.0
3	Bodnar	11.3	140.0
4	Lukianov	10.9	150.0
5	Feduniak	11.5	155.0
6			165.0
7			145.0
8			140.0
9			150.0
10	Ovcharenko	11.5	155.0
11	Sakh	10.7	165.0
12	Yaroshovych	11.2	145.0
13	Franchuk	11.7	140.0
14	Debeliak	10.8	150.0
15	Holub	11.0	148.0

Message

Price successfully set

OK

Рис. 4.6. Встановлення ціни в межах року за один кубометр води та за одну кВт·год

Flat consumption manager

Menu

Manage functions

Year: 2023

Water price: 29.1 Light price: 1.6

Set price

Not filled information

Output to file

Display savers

Months expenses

Display rich person

Clear all info

2023 January

Nº	Surname	Water use	Light use
1	Legeza	11.5	150.0
2	Skakovskiy	10.5	160.0
3	Bodnar	11.3	140.0

Month expenses

Month	Water Expenses	Light Expenses	Total Expenses
January	4859,70	3612,80	8472,50
February	4337,81	3379,80	7717,61
March	5399,96	3181,40	8581,36
April	4980,92	4119,00	9099,92
May	4090,46	3424,60	7515,06
June	3747,08	3834,20	7581,28
July	5085,68	4074,20	9159,88
August	5330,12	3906,68	9236,80
September	4163,21	3603,80	7767,01
October	4628,81	3531,80	8160,61
November	2929,37	3794,20	6723,57
December	3974,06	3989,40	7963,46

Рис. 4.7. Результат розрахунку та відображення вартості спожитих послуг по місячно, агреговану для всіх квартир

Flat consumption manager

Menu

Add flat consumption

Flat number

Flat №1

Surname

2019 January

2019 2020 2021 2022 2023

Nº	Surname
1	Unknown
2	Unknown
3	Unknown
4	Unknown

Рис. 4.8. Можливість завантаження у програму більше одного року

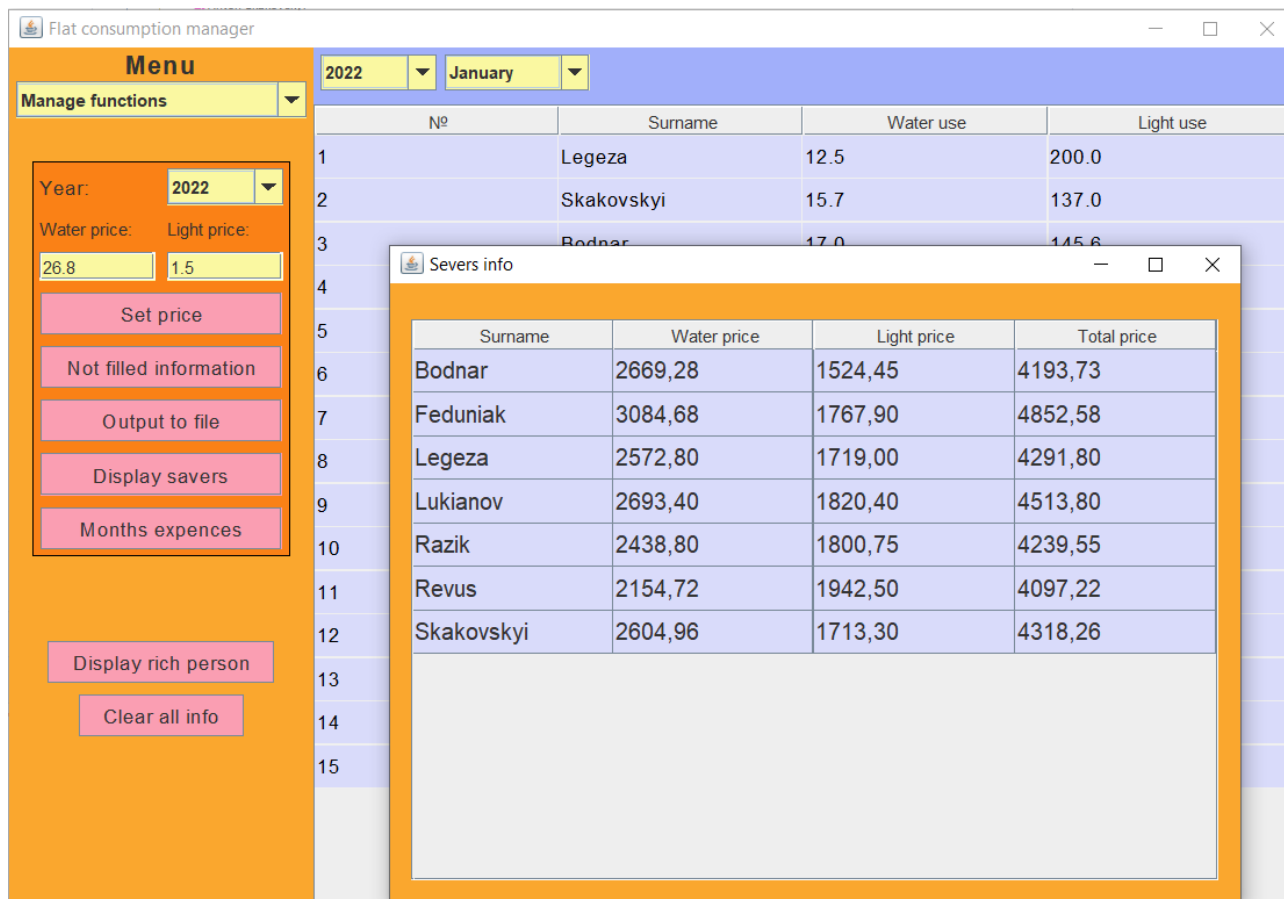


Рис. 4.9. Результат виведення власників, які за рік спожили найменше послуг в грошовому еквіваленті

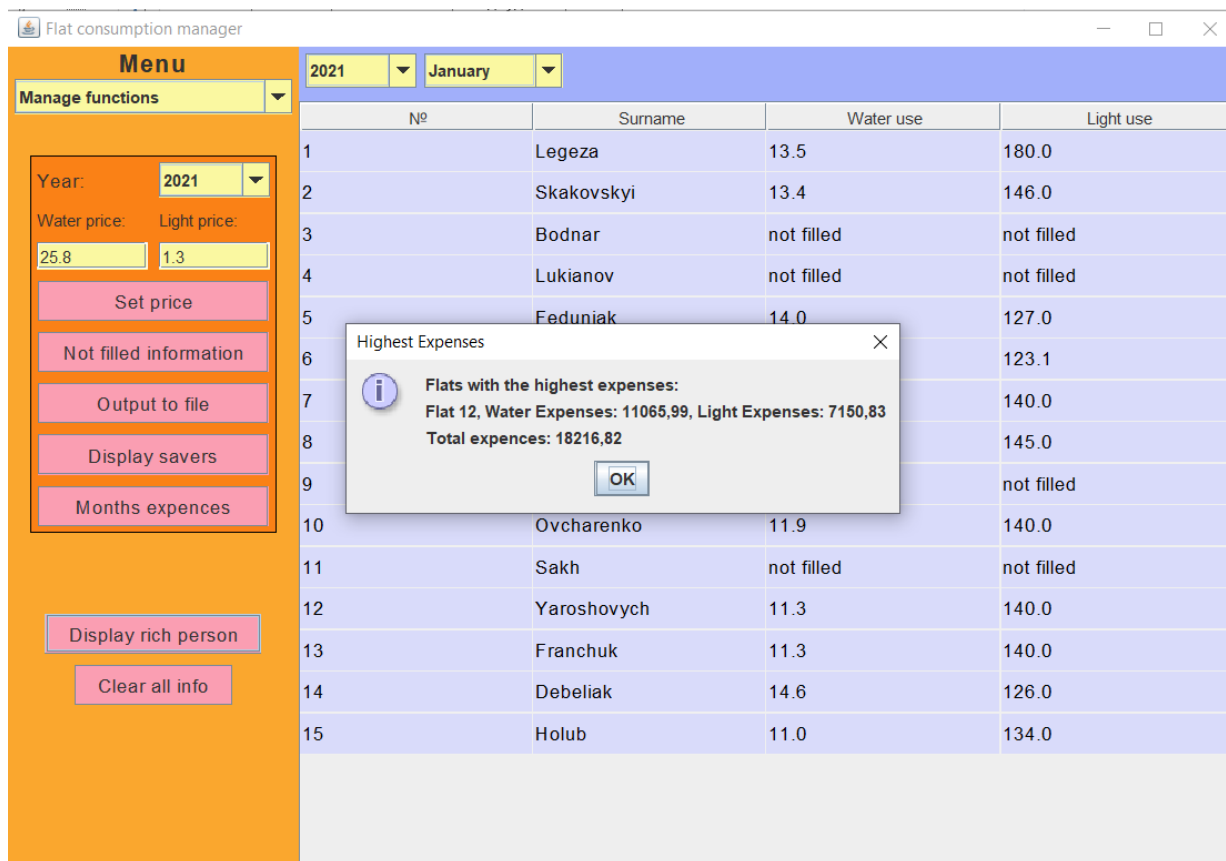


Рис. 4.10. Результат виведення номеру квартири, яка спожила найбільше послуг у грошовому еквіваленті в межах усіх років

5. Інструкція користувача

5.1. Компоненти ПЗ

Програму розроблено на мові програмування Java у середовищі IntelliJ IDEA і може експлуатуватися на базі операційних систем з наявним пакетом Java. Для коректної роботи пакету необхідна користувацька машина з частотою ЦП не менше 1.2 GHz, оперативною пам'яттю не менше 1024 мб.

5.2 Встановлення та налаштування ПЗ

- 1) Завантажити Java Oracle 8 за посиланням <https://www.java.com/ru/download/>
- 2) Запустити попередньо завантажену версію SmartHouse.jar

5.3. Базові функції ПЗ

Початок роботи програми:

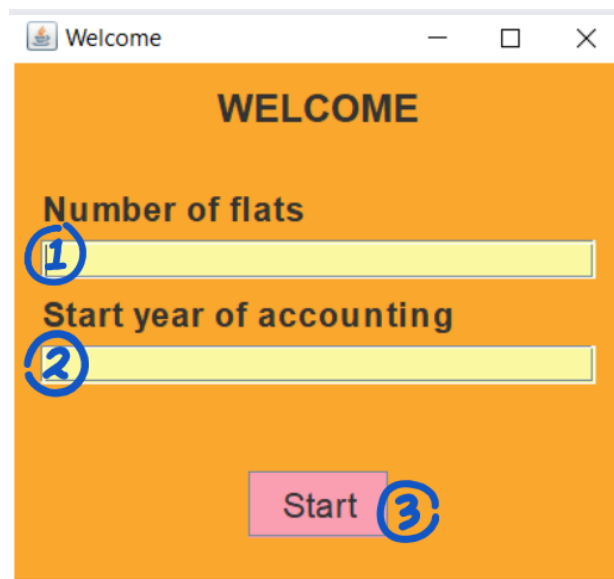
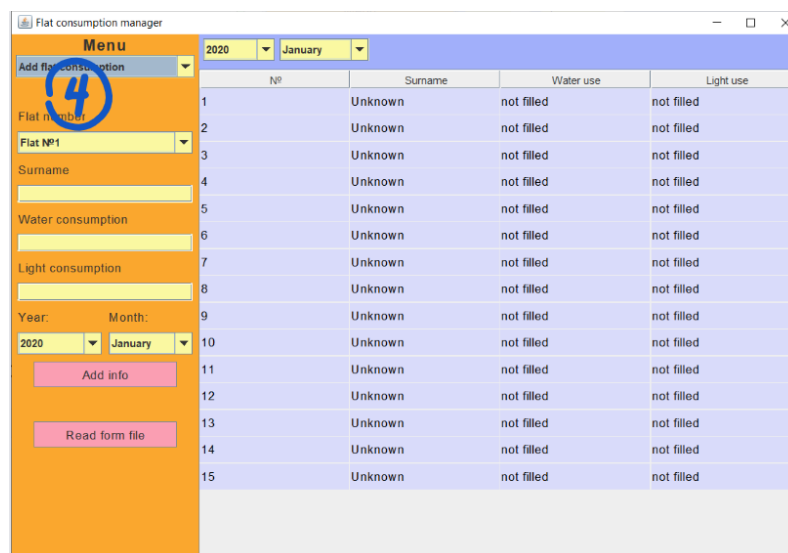


Рис. 5.1. Початкове вікно



№	Surname	Water use	Light use
1	Unknown	not filled	not filled
2	Unknown	not filled	not filled
3	Unknown	not filled	not filled
4	Unknown	not filled	not filled
5	Unknown	not filled	not filled
6	Unknown	not filled	not filled
7	Unknown	not filled	not filled
8	Unknown	not filled	not filled
9	Unknown	not filled	not filled
10	Unknown	not filled	not filled
11	Unknown	not filled	not filled
12	Unknown	not filled	not filled
13	Unknown	not filled	not filled
14	Unknown	not filled	not filled
15	Unknown	not filled	not filled

Рис. 5.2 .Основне вікно програми

Для початку роботи прогарми потрібно заповнити два інформаційних поля (рис. 5.1; опція 1 та опція 2), після чого натиснути Start (рис. 5.1; опція 3). У разі успіху вказана інформація відобразиться вікно (рис. 5.2; опція 4). На цьому етапі можна починати роботу з інформацією.

F1. Ввід інформації з файлу.

Для зчитування інформації необхідно перейти у режим меню Add flat consumption (рис. 5.3; опція 1), після чого натиснути на кнопку Read from file (рис. 5.3; опція 2). У відкритому діалоговому вікні (рис. 5.3; опція 3) обрати необхідний файл, після чого натиснути Open (рис. 5.3; опція 4). У разі успіху з'явиться повідомлення “Information successfully read” (рис 5.4). На цьому етапі можна починати роботу з інформацією.

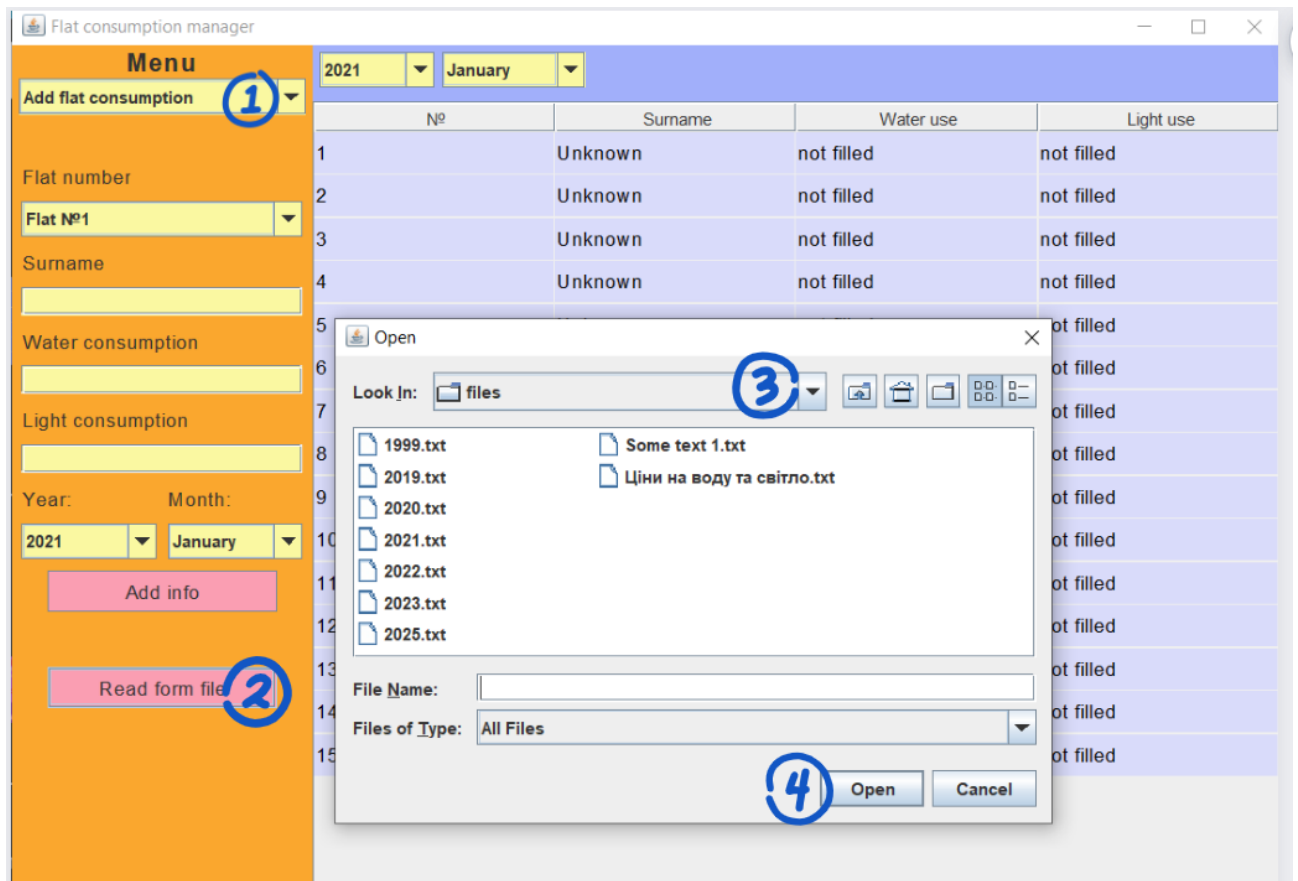


Рис.5.3. Файловий ввід інформації

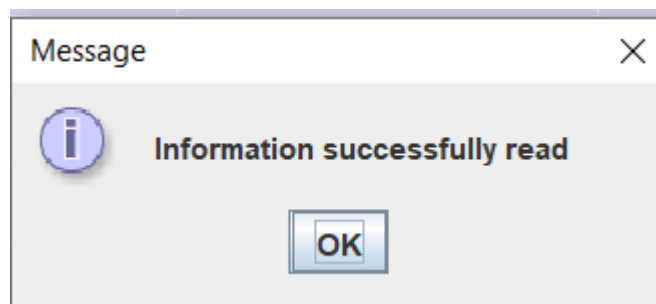


Рис. 5.4. Результат файлового вводу інформації

F2. Ручний ввід.

Для вводу інформації необхідно перейти у режим меню Add flat consumption (рис. 5.5; опція 1), після чого заповнити інформаційні поля (рис.5.5; опція 2), після чого натиснути Add info (рис. 5.5; опція 3). У разі успіху вказана інформація відобразиться у таблиці (рис. 5.5; опція 4). На цьому етапі можна починати роботу з інформацією.

Flat consumption manager

Menu

Add flat consumption 1

Flat number

Flat №1 2

Surname

Legeza

Water consumption

15

Light consumption

14

Year: 2020 Month: March

Add info 3

Read form file

№	Surname	Water use	Light use
1	Legeza	15.0	14.0 4
2	Unknown	not filled	not filled
3	Unknown	not filled	not filled
4	Unknown	not filled	not filled
5	Unknown	not filled	not filled
6	Unknown	not filled	not filled
7	Unknown	not filled	not filled
8	Unknown	not filled	not filled
9	Unknown	not filled	not filled
10	Unknown	not filled	not filled
11	Unknown	not filled	not filled
12	Unknown	not filled	not filled
13	Unknown	not filled	not filled
14	Unknown	not filled	not filled
15	Unknown	not filled	not filled

Рис.5.5. Ручний ввід інформації

Е3. Вивід інформації у файл.

Для запису інформації у файл необхідно перейти у режим меню Manage functions (рис. 5.6; опція 1), вказати рік (рис. 5.6; опція 2). Потім натиснути кнопку Output to file (рис. 5.6; опція 3). У разі успіху з'явиться напис "Information successfully" (рис. 5.7)

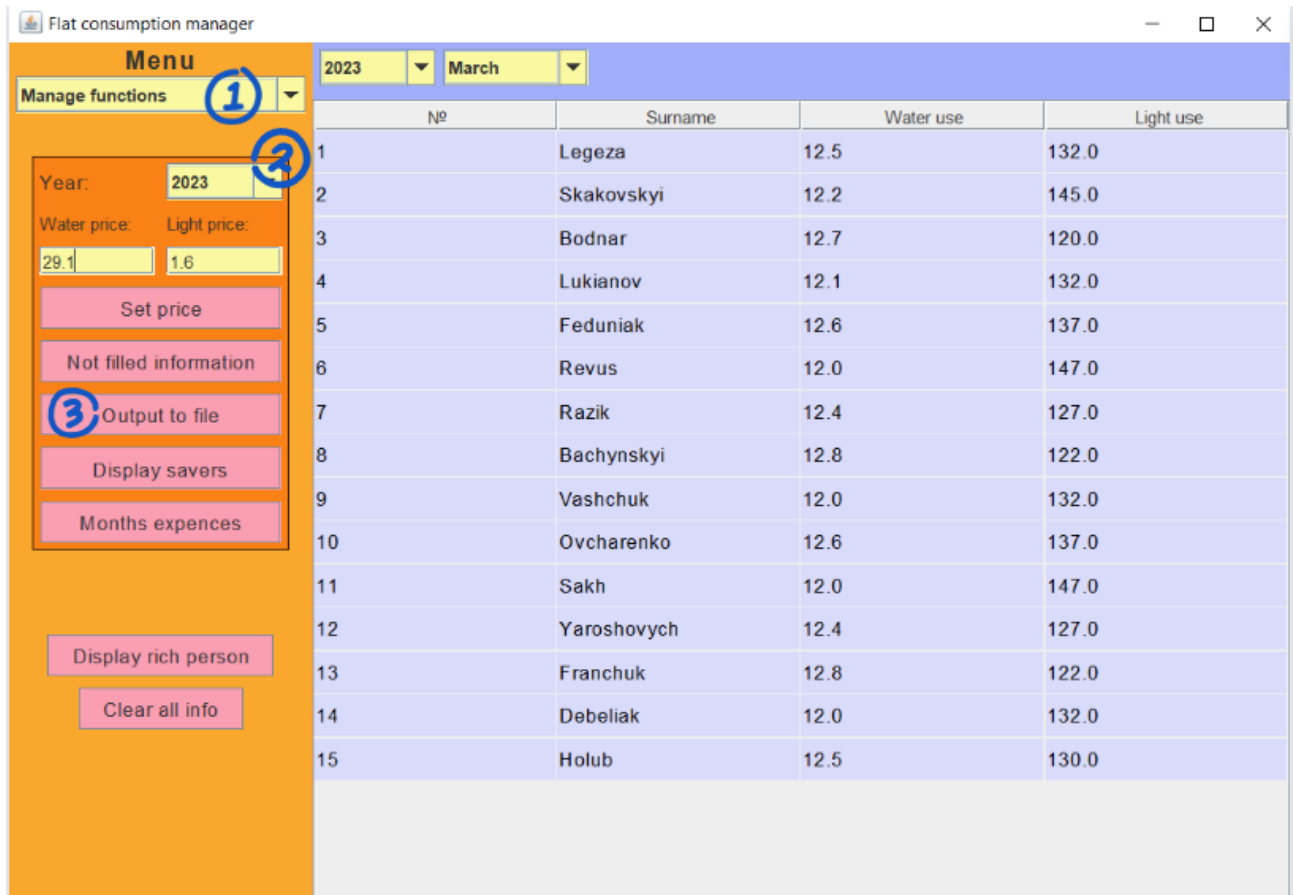


Рис.5.6. Вивід інформації у файл

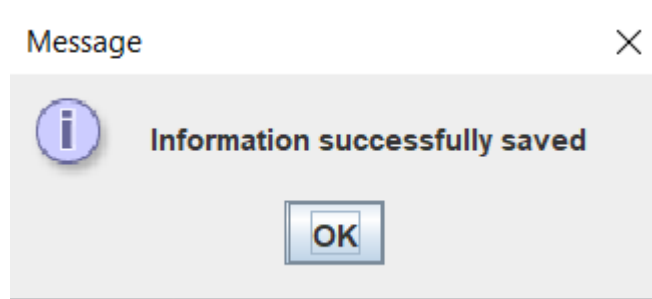


Рис. 5.7. Результат виводу в файл

Ф4. Встановлення ціни в межах року.

Для встановлення ціни в межах року за один кубометр води та за одну кВт·год необхідно перейти у режим меню Manage functions (рис. 5.8; опція 1), вказати рік (рис. 5.8; опція 2), заповнити поля дійсними числовими значеннями (рис. 5.8; опція 3). Потім натиснути кнопку Set price (рис. 5.8; опція 4). У разі успіху з'явиться напис "Price successfully set" (рис. 5.9)

Nº	Surname	Water use	Light use
1	Legeza	12.5	200.0
2	Skakovskiy	15.7	137.0
3	Bodnar	17.0	145.6
4	Lukianov	15.2	143.0
5	Feduniak	16.2	136.0
6	Revus	not filled	not filled
7	Razik	10.2	130.0
8	Bachynskiy	11.0	135.0
9	Vashchuk	not filled	not filled
10	Ovcharenko	11.3	140.0
11	Sakh	11.3	140.0
12	Yaroshovych	11.3	140.0
13	Franchuk	not filled	not filled
14	Debeliak	11.3	140.0

Рис. 5.8. Встановлення ціни

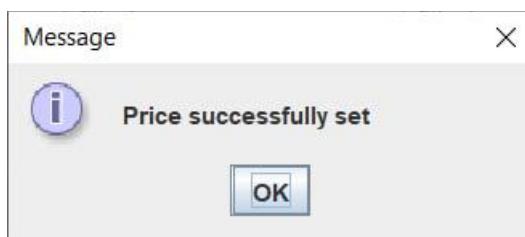


Рис. 5.9. Результат вставлення ціни

F5. Відображення місяців, для яких інформація не заповнена для кожної з квартир.

Для відображення місяців, для яких інформація не заповнена для кожної з квартир необхідно перейти у режим меню Manage functions (рис. 5.10; опція 1), вказати рік (рис. 5.10; опція 2). Потім натиснути кнопку Not filled information (рис.5.10; опція 3). У разі успіху з'явиться вікно «Months Info» (рис.5.10; опція 4)

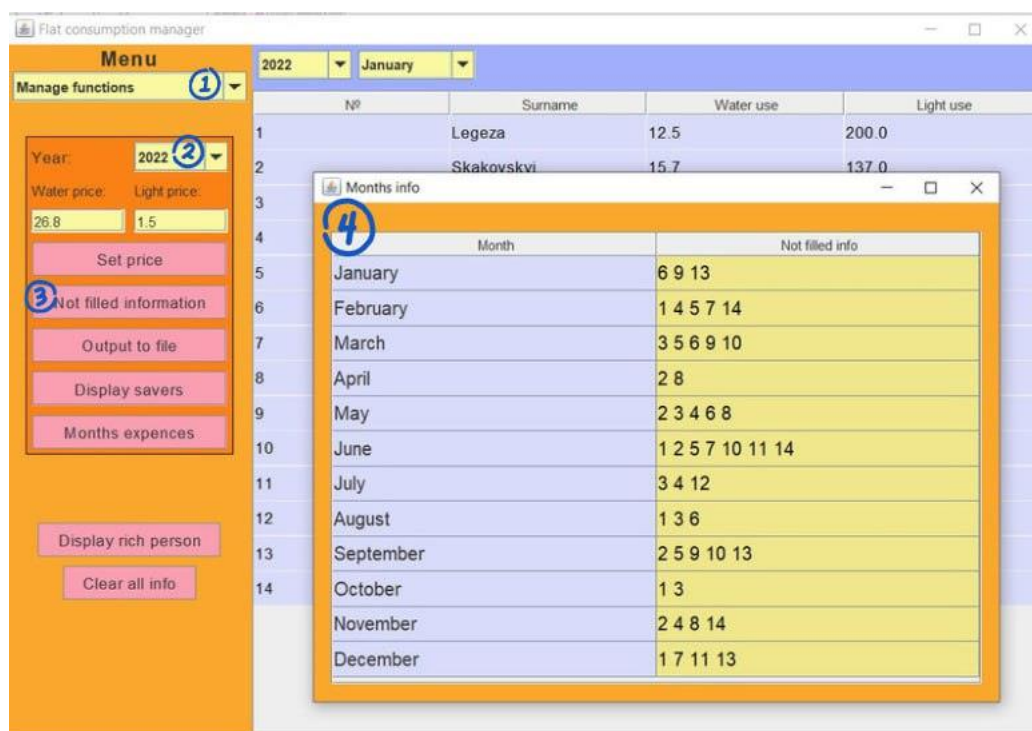


Рис. 5.10. Відображення місяців, для яких інформація не заповнена для кожної з квартир

Ф6. Розрахунок та відображення вартості спожитих послуг по місячно, агреговану для всіх квартир.

Для розрахунок та відображення вартості спожитих послуг по місячно, агреговану для всіх квартир необхідно перейти у режим меню Manage functions (рис. 5.11; опція 1), вказати рік (рис. 5.11; опція 2), встановити ціну (рис. 5.11; опція 3). Потім натиснути кнопку Months Experiences (рис. 5.11; опція 4). У разі успіху з'явиться вікно «Month expences» (рис. 5.11; опція 5)

The screenshot displays the 'Flat consumption manager' application. The main window has a 'Menu' section on the left with the following elements:

- Manage functions** (labeled 1)
- Year:** 2022 (labeled 2)
- Water price:** 26.8 (labeled 3)
- Light price:** 1.5 (labeled 3)
- Buttons:** Set price, Not filled information, Output to file, Display savers, Months expences (labeled 4), Display rich person, Clear all info

The 'Month expences' window (labeled 5) shows a table of monthly expenses for January 2022:

Month	Water Expences	Light Expences	Total Expences
January	3832,40	2379,90	6212,30
February	3340,96	1902,20	5243,16
March	3381,16	1961,00	5342,16
April	4445,12	2734,40	7179,52
May	3306,12	2231,00	5537,12
June	2242,16	1646,00	3888,16
July	4158,36	2330,15	6488,51
August	3826,04	2539,55	6365,59
September	3059,56	1854,80	4914,36
October	3847,48	2765,45	6612,93
November	2882,68	2315,60	5198,28
December	3185,52	2341,10	5526,62

Рис. 5.11. Відображення вартості спожитих послуг по місячно, агреговану для всіх квартир

Ф7. Вивести власників, які за рік спожили найменше послуг в грошовому еквіваленті.

Для виведення власників, які за рік спожили найменше послуг в грошовому еквіваленті необхідно перейти у режим меню Manage functions (рис. 5.12; опція 1), вказати рік (рис. 5.12; опція 2), встановити ціну (рис. 5.12; опція 3). Потім натиснути кнопку Display servers (рис 5.12; опція 4). Потім ввести к-сть власників для відображення (рис 5.12; опція 5). У разі успіху з'явиться вікно «Servers info» (рис 5.13; опція 6).

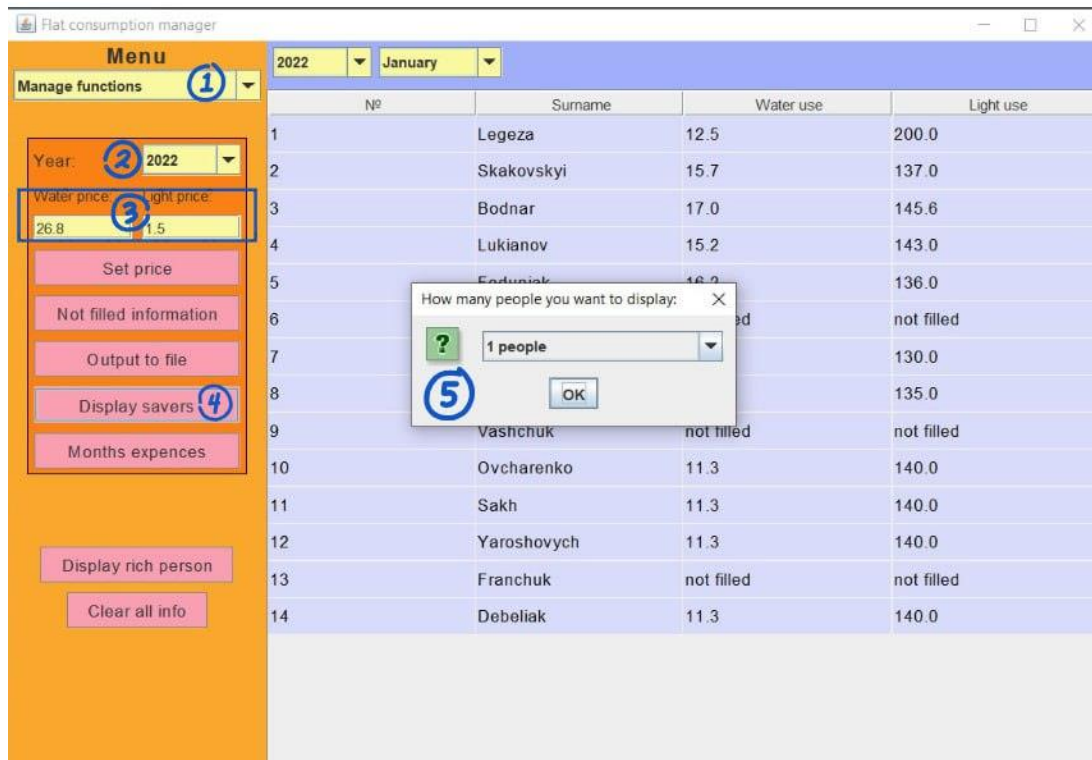


Рис. 1.12. Відображення власників, що спожили найменше послуг в грошовому еквіваленті

Surname	Water price	Light price	Total price
Bodnar	3579,30	2765,60	6344,90
Feduniak	3631,68	3050,08	6681,76
Legeza	3573,48	2977,12	6550,60
Lukianov	3550,20	2954,88	6505,08
Revus	3518,19	3240,80	6758,99
Skakovskiy	3567,66	3102,72	6670,38

Рис. 5.13. Вивід власників

F8. Виведення номеру квартири, яка спожила найбільше послуг у грошовому еквіваленті в межах усіх років.

Для виведення номеру квартири, яка спожила найбільше послуг у грошовому еквіваленті в межах усіх років необхідно перейти у режим меню Manage functions (рис. 5.14; опція 1), для всіх років встановити ціну (рис. 5.14; опція 2). Потім натиснути кнопку Display servers rich person (рис. 5.14; опція 3). З'явиться діалогове вікно (рис. 5.14; опція 4).

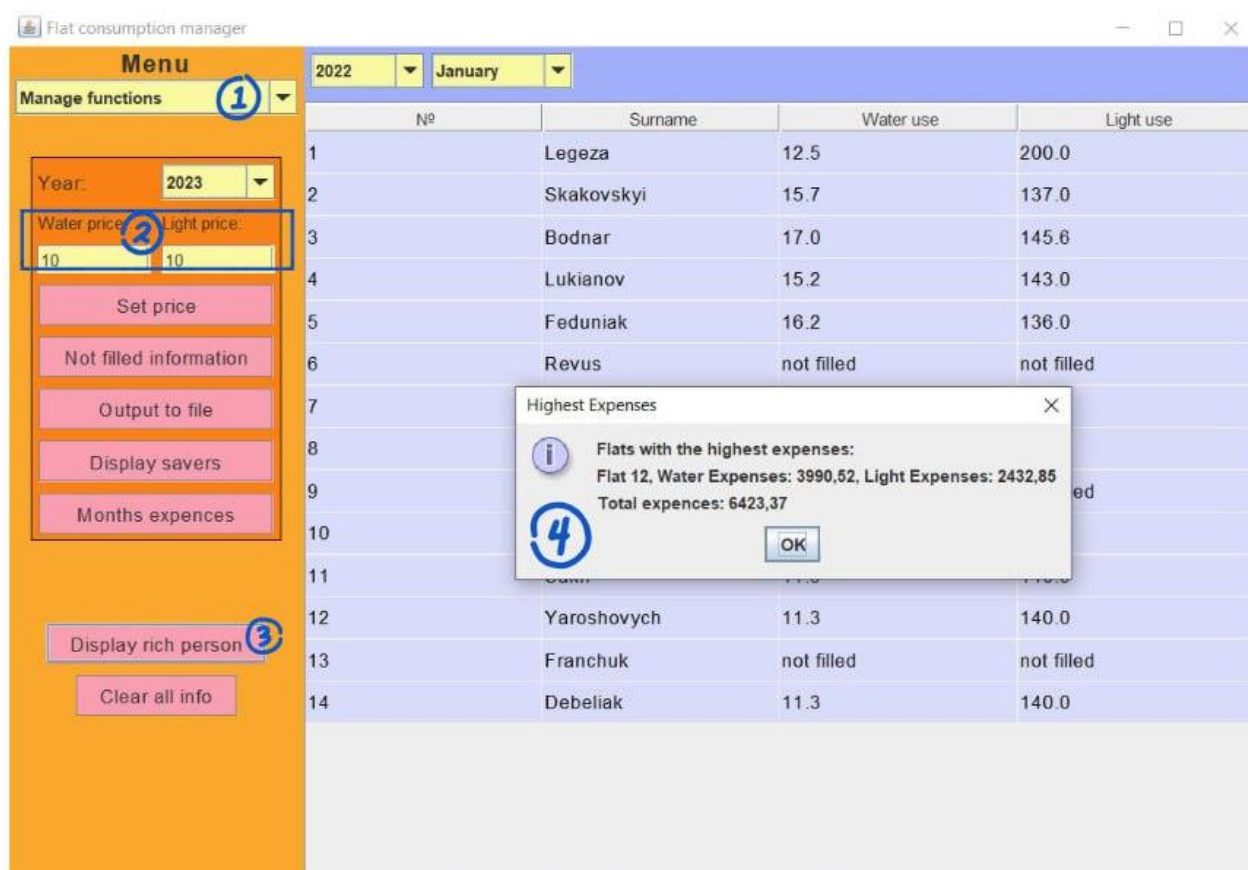


Рис. 5.14. Вивід номеру квартири, що спожила найбільше послуг в грошовому еквіваленті за всі роки

Ф9. Очищення всіх даних.

Для очищення всіх даних необхідно перейти у режим меню Manage functions (рис. 5.15; опція 1). Потім натиснути кнопку Clear all info (рис 5.15; опція 2). З'явиться діалогове вікно (рис 5.15; опція 3).



Рис. 5.15. Очищення інформації

6. Опис виняткових ситуацій

6.1. Якщо при спробі розпочати програму поля не заповнені або містять не числове значення:

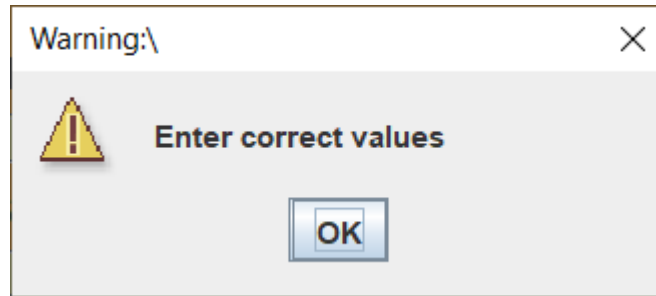


Рис. 6.1. Повідомлення «Введіть правильні значення»

6.2. Якщо при спробі розпочати програму поле кількості квартир містить число не більше за один:

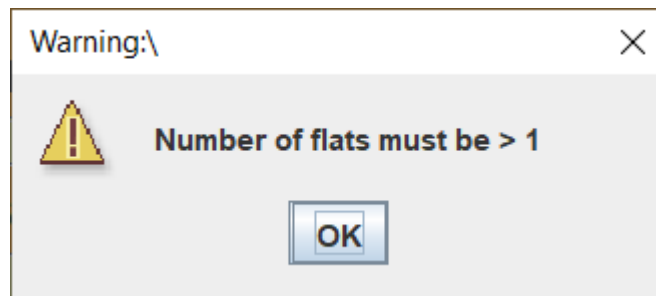


Рис. 6.2. Повідомлення «Кількість квартир повина бути > 1 »

6.3. Якщо при спробі розпочати програму поле початку року обліку містить числове значення менше за 1 або більше за значення поточного року:

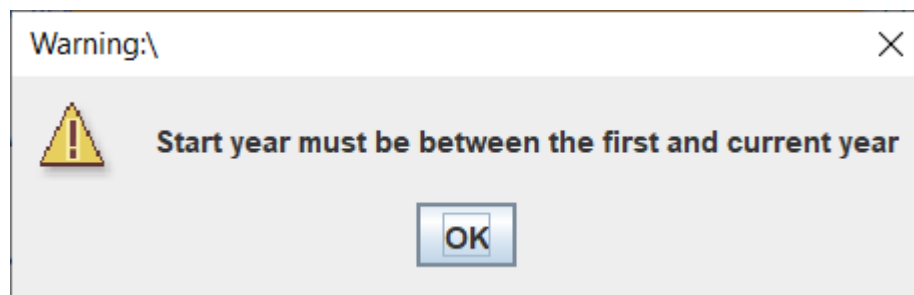


Рис. 6.3. Повідомлення «Початковий рік повинен бути від 1 до поточного року»

6.4. Якщо при спробі додавання інформації поле для вводу прізвища не заповнене:

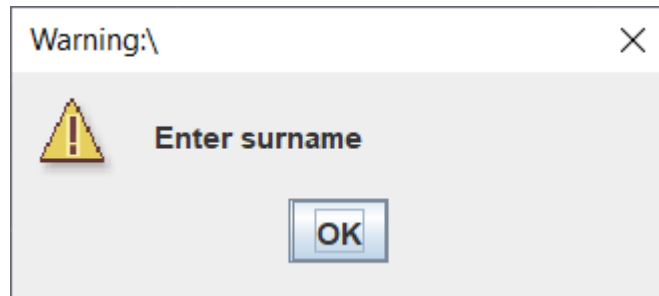


Рис. 6.4. Повідомлення «Введіть прізвище»

6.5. Якщо при спробі додавання інформації поле для вводу к-сті спожитої води не заповнене:

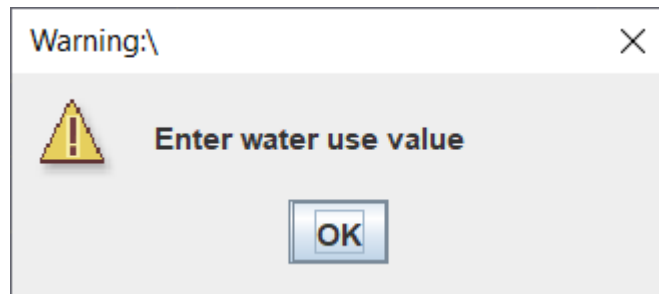


Рис. 6.5. Повідомлення «Введіть значення спожитої води»

6.6. Якщо при спробі додавання інформації поле для вводу к-сті витраченого світла не заповнене:

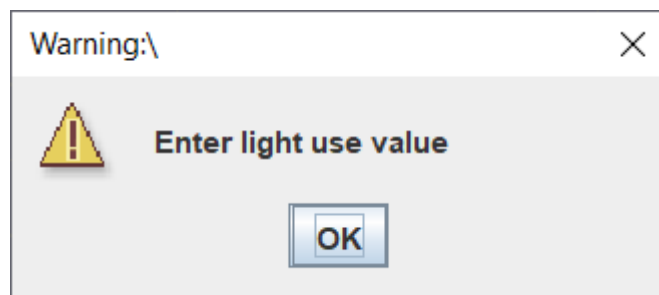


Рис. 6.6. Повідомлення «Введіть значення витраченого світла»

6.7. Якщо при спробі додавання інформації поле для вводу спожитої води або витраченого світла містять не дійсні числові значення:

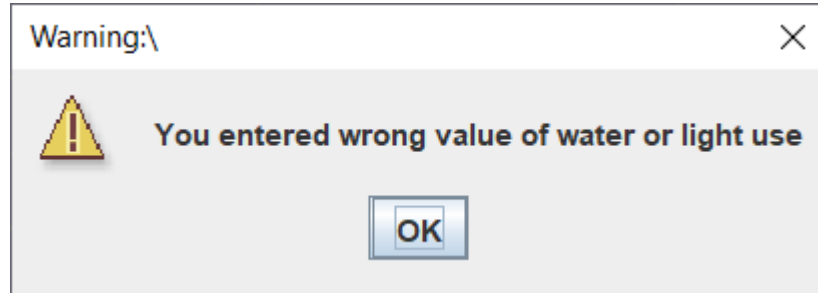


Рис. 6.7. Повідомлення «Ви ввели хибні значення використаної води або світла»

6.8. Якщо при спробі зчитування даних користувач відмінив вибір файлу:

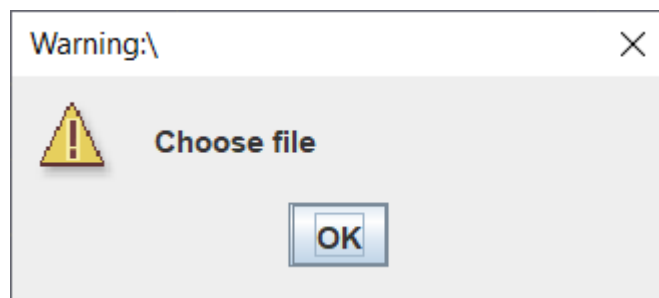


Рис. 6.8. Повідомлення «Оберіть файл »

6.9. Якщо при спробі зчитування даних користувач обрав файл, рік якого не входить в межі обліку витрат:

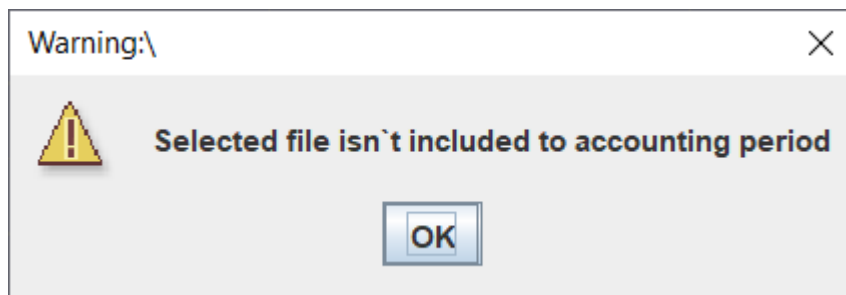


Рис. 6.9. Повідомлення «Вибраний файл не входить до розрахункового періоду»

6.10. Якщо при спробі зчитування даних користувач обрав файл, ім'я якого не відповідає року або не містить коректні дані для зчитування:

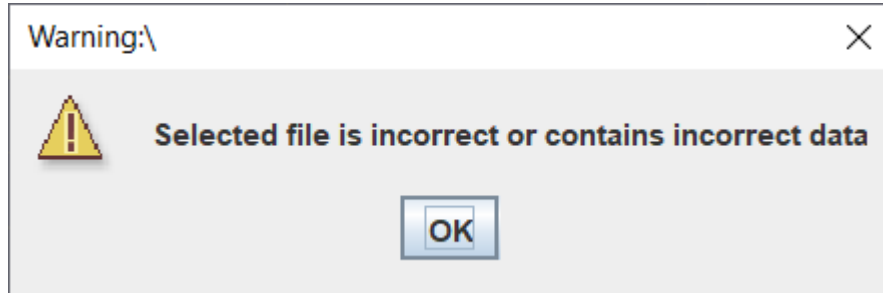


Рис. 6.10. Повідомлення «Вибраний файл є некоректним або містить некоректні дані»

6.11. Якщо при спробі вказання ціни за 1 м³ поле для вводу не заповнене:

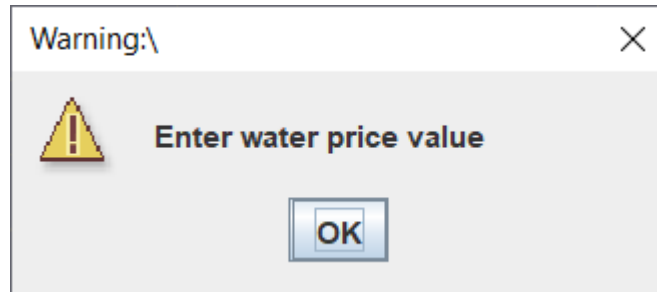


Рис. 6.11. Повідомлення «Введіть значення вартості води»

6.12. Якщо при спробі вказання ціни за 1 кВт•год поле для вводу не заповнене:

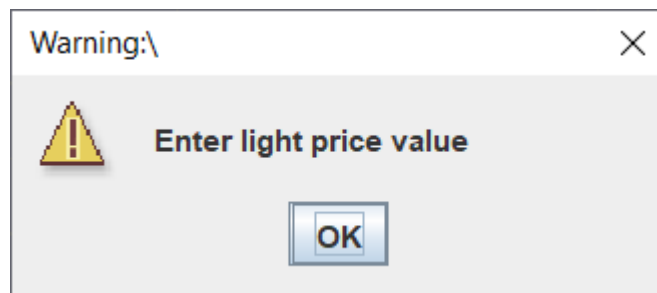


Рис. 6.12. Повідомлення «Введіть значення вартості світла»

6.13. Якщо при спробі вказання ціни за 1 м³ або 1 кВт•год поле для вводу містить не дійсні числові значення:

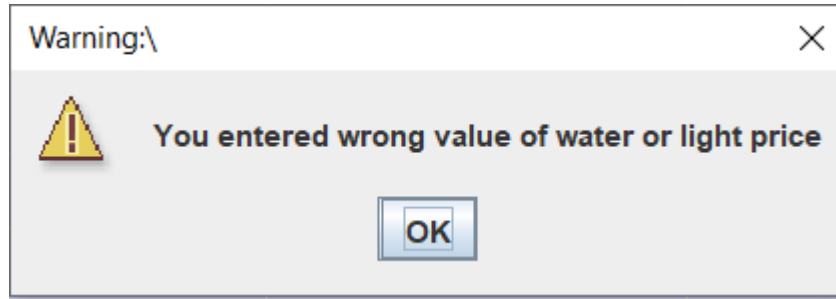


Рис. 6.13. Повідомлення «Ви ввели невірне значення ціни за воду або світло»

6.14. Якщо при спробі визначення номеру квартири, яка спожила найбільше послуг в грошовому еквіваленті не встановлено ціна за ресурси для всіх років:

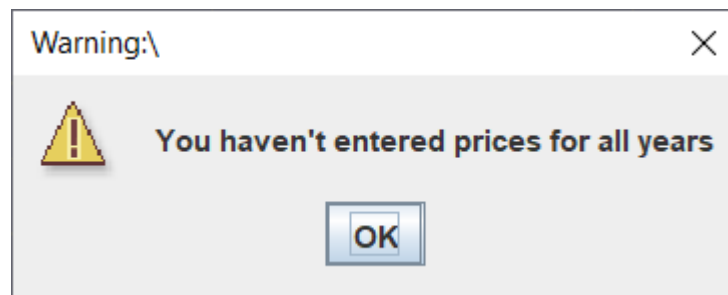


Рис. 6.14. Повідомлення «Ви не ввели ціни для всіх років»

6.15. Якщо при спробі визначення номеру квартири, яка спожила найбільше послуг в грошовому еквіваленті не зазначено жодної інформації щодо витрат квартир:

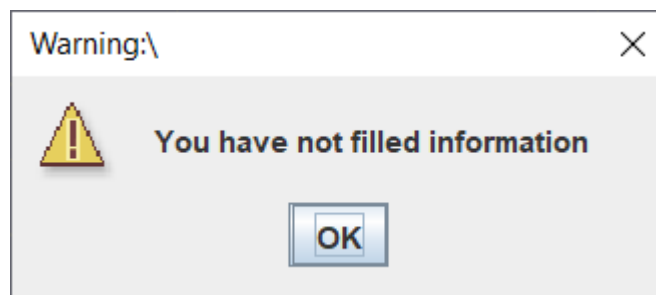


Рис. 6.15. Повідомлення «Ви не заповнили інформацію»

6.16. Якщо при спробі обрахунку витрат всіх квартир агреговано для кожного місяця не вказано цін послуг за рік:

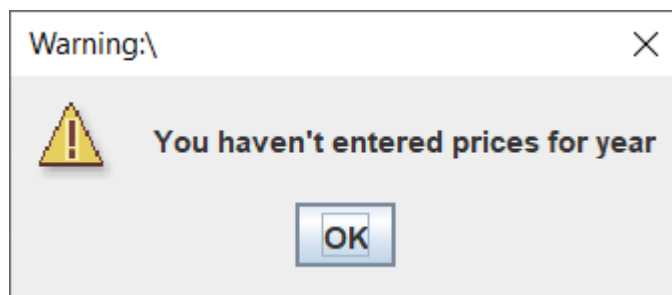


Рис. 6.16. Повідомлення «Ви не вказали ціни на рік»

7. Структура файлу вхідних даних

Помісячна інформація про облік комунальних витрат багатоквартирного будинку збурігаються в текстовому файлі. Розроблена програма дозволяє зберігати дані у файл та імпортувати з файлу, за умови що дані записані у певному форматі.

Ім'я файлу відповідає року з даними про комунальні витрати: rrrr.txt

Приклад назви файлу: 2021.txt

Формат запису даних:

Перший рядок:

- Ціна за воду Ціна за світло

Подальші рядки повторюються в такому форматі:

- Номер квартири
- Прізвище
- Номер місяця К-сть спожитих м³ води К-сть витрачених кВт•год електроенергії

Приклад запису даних:

29.1 1.6

1

Legeza

1 11.5 150

2 10 140

3 12.5 132

4 11.3 180

5 9 142

6 8 161

7 11.9 171

8 12.1 163.7

9 9.7 150

10 10.9 147

11 6.9 158

12 9 166

Номер квартири. Записується цілим числовим значенням починаючи з 1

Прізвище. Записується стрічкою символів з латинських літер (Стрічка “Unknown” – зарезервована для позначення незаповненої інформації)

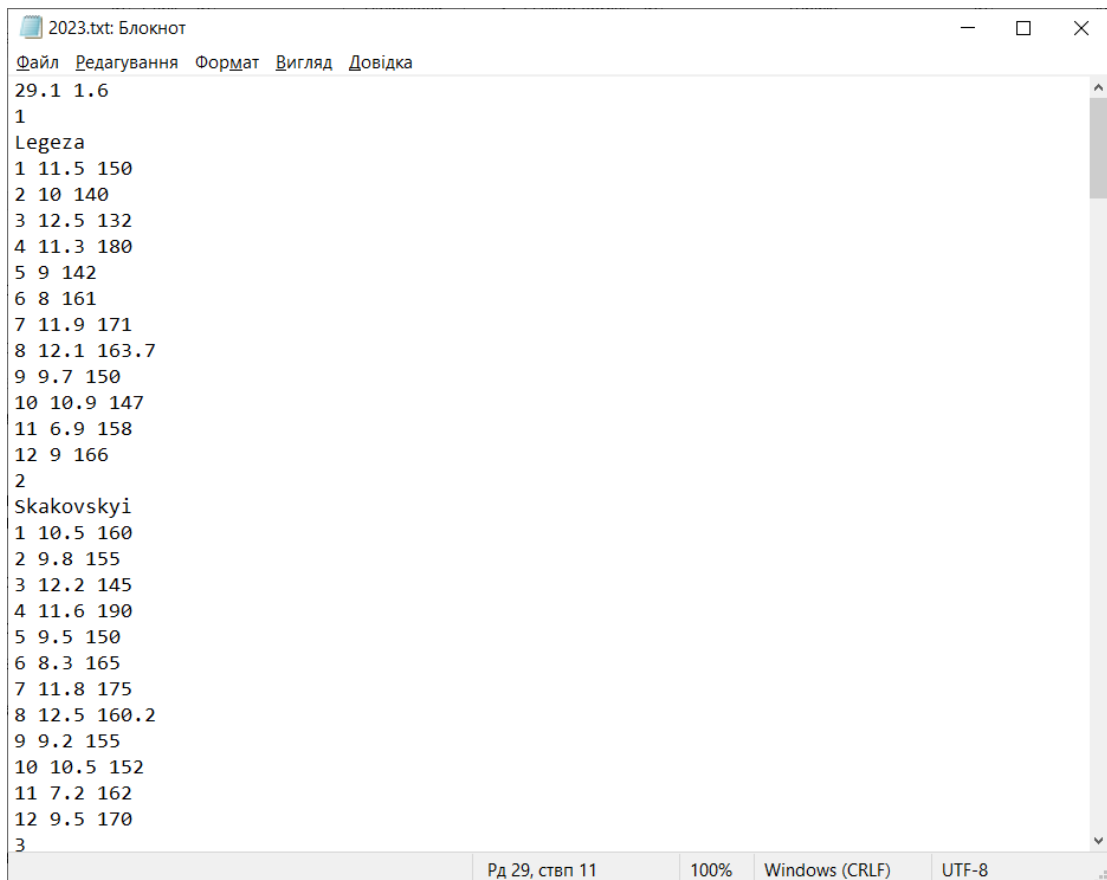
Номер місяця. Записується цілим числовим значенням від 1 до 12

Ціна за воду. Записується дійсним числовим значенням; відображає вартість за 1 м³. Валюта UAH, гривня. (0 - зарезервованим для позначення незаповненої інформації)

Ціна за світло. Записується дійсним числовим значенням; відображає вартість за 1 кВт•год. Валюта UAH, гривня. (Значення 0 - зарезервованим для позначення незаповненої інформації)

К-сть витраченої води. Записується дійсним числовим значенням; відображає кількість витрачених м³ води за 1 місяць; має бути невід’ємним значенням. (Значення -1 - зарезервованим для позначення незаповненої інформації)

К-сть витраченого світла. Записується дійсним числовим значенням; відображає кількість витрачених кВт•год за 1 місяць; має бути невід’ємним значенням. (Значення -1 - зарезервованим для позначення незаповненої інформації)



```
2023.txt: Блокнот
Файл  Редагування  Формат  Вигляд  Довідка
29.1 1.6
1
Legeza
1 11.5 150
2 10 140
3 12.5 132
4 11.3 180
5 9 142
6 8 161
7 11.9 171
8 12.1 163.7
9 9.7 150
10 10.9 147
11 6.9 158
12 9 166
2
Skakovskyi
1 10.5 160
2 9.8 155
3 12.2 145
4 11.6 190
5 9.5 150
6 8.3 165
7 11.8 175
8 12.5 160.2
9 9.2 155
10 10.5 152
11 7.2 162
12 9.5 170
3
Рд 29, ствп 11 100% Windows (CRLF) UTF-8
```

Рис. 7.1. Приклад подання інформації у файлі

Висновок

Під час виконання цієї курсової роботи я закріпив теоретичні та практичні навички, набуті при вивченні дисципліни «Об'єктно орієнтоване програмування». Також поглибив знання у багатьох областях, що стосуються теорії інженерії програмного забезпечення, програмованих алгоритмів, розробки програмного забезпечення мовою Java та створення інтерфейсних застосунків з застосуванням фреймворку Java Swing. Отримав практичні навички створення документації до програмного забезпечення, а саме: інструкції користувача, покрокового представлення алгоритмів та UML діаграм. Як результат, розробив свій застосунок відповідно до заданого функціоналу варіанту №ПЗ25_16.

Список використаної літератури

1. Об'єктно-орієнтоване програмування: методичні вказівки до виконання курсової роботи для студентів спеціальності 6.121 «Інженерія програмного забезпечення» / Укл. Коротєєва Т.О., Дяконюк Л.М.— Львів: Національний університет “Львівська політехніка” кафедра програмного забезпечення, 2023. – 27с.
2. Левус Є., Мельник Н. Вступ до інженерії програмного забезпечення : навч. посіб. Львів : Вид-во Львів. політехніки, 2018. 246 с.
3. Коротєєва Т.О. Алгоритми та структури даних: навчальний посібник. Львів : Вид-во Львів. політехніки, 2014. 279 с.
4. The Java™ Tutorials [Електронний ресурс] – Режим доступу: <https://docs.oracle.com/javase/tutorial/index.html>
5. W3 schools Java Tutorial [Електронний ресурс] – Режим доступу: <https://www.w3schools.com/java/default.asp>
6. Java2s Java Swing Tutorials [Електронний ресурс] – Режим доступу: http://www.java2s.com/Tutorials/Java/Java_Swing/index.htm