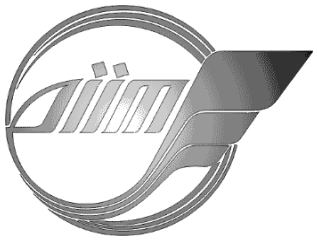
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



Дніпровський національний університет
залізничного транспорту імені академіка В. Лазаряна

Кафедра «Комп'ютерні інформаційні технології»

Лабораторна робота № 16

з дисципліни «Об'єктно-орієнтоване програмування»

на тему: «Серіалізація в Java»

Виконав:
студент гр.ПЗ1911
Сафонов Д. Є.
Прийняла:
Демидович І.М.

Дніпро, 2021

Тема. Серіалізація в Java.

Завдання. Написати об'єктно-орієнтовану програму на мові Java, яка використовує серіалізацію та десеріалізацію для збереження та зчитування пов’язаних через посилання об’єктів у файл. Підготувати та виконати експерименти для перевірки збереження даних в повному обсязі та перевірки контролю версій класів, що десеріалізуються.

Діаграма класів.

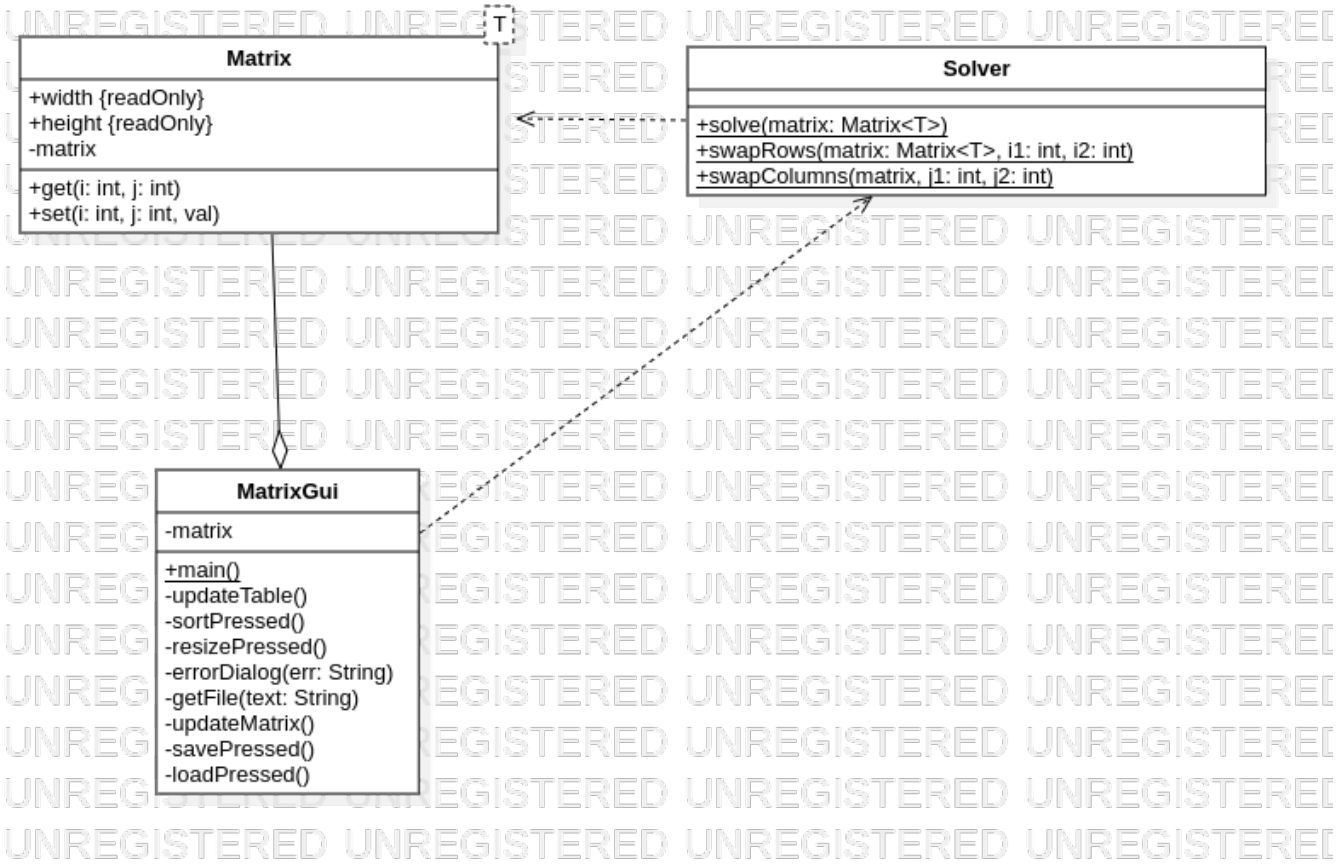


Рисунок 1

Опис потоків, які використовуються в програмі.

ObjectInputStream - клас, який десеріалізує примітивні дані та об’єкти класів, які реалізують інтерфейс Serializable або Externalizable, який є спеціалізацією інтерфейсу Serializable.

ObjectOutputStream - клас, який серіалізує примітивні дані та об’єкти класів, які реалізують інтерфейс Serializable або Externalizable, який є спеціалізацією інтерфейсу Serializable, і записує їх у файл.

FileInputStream - клас для зчитування байтів з файлів.

FileOutputStream - клас для запису байтів у файл.

Текст програми.

“MatrixGui.java”

```
import com.dazzlemon.oop16.Matrix;
import com.dazzlemon.oop16.Solver;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.filechooser.FileSystemView;

public class MatrixGui extends javax.swing.JFrame {
    public MatrixGui() {
        initComponents();
        //
        updateTable();
        //
    }

    private void updateTable() {
        var m = new Integer[matrix.height][matrix.width];
        for (int i = 0; i < matrix.height; i++) {
            m[i] = new Integer[matrix.width];
            for (int j = 0; j < matrix.width; j++) {
                m[i][j] = matrix.get(i, j);
            }
        }

        var names = new String[matrix.width];
        for (int j = 0; j < matrix.width; j++) {
            names[j] = String.valueOf(j);
        }
        table.setModel(new javax.swing.table.DefaultTableModel(m, names));
        table.setShowGrid(true);
        table.setTableHeader(null);
        jScrollPane1.setViewportView(table);
    }

    private void sortButtonActionPerformed(java.awt.event.ActionEvent evt) {
        updateMatrix();
```

```
        var indices = Solver.solve(matrix);
        var x = indices.x;
        var y = indices.y;

        Solver.swapRows(matrix, 0, x);
        Solver.swapColumns(matrix, 0, y);

        updateTable();
    }

    private void resizeButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int w = Integer.parseInt(columnsText.getText());
        int h = Integer.parseInt(rowsText.getText());
        var newMatrix = new Matrix<>(h, w, 0);
        for (int i = 0; i < h && i < matrix.height; i++) {
            for (int j = 0; j < w && j < matrix.width; j++) {
                var mij = Integer.parseInt(table
                                    .getModel()
                                    .getValueAt(i, j)
                                    .toString());
                newMatrix.set(i, j, mij);
            }
        }
        matrix = newMatrix;
        updateTable();
    }

    private void updateMatrix() {
        for (int i = 0; i < matrix.height; i++) {
            for (int j = 0; j < matrix.width; j++) {
                var mij = Integer.parseInt(table
                                    .getModel()
                                    .getValueAt(i, j)
                                    .toString());
                matrix.set(i, j, mij);
            }
        }
    }

    private void errorDialog(String err) {
        JOptionPane.showMessageDialog(
                new JFrame(),
                err,
                "Error",
                JOptionPane.ERROR_MESSAGE
        );
    }
```

```
    private void saveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        var file = getFile("save");
        if (file == null) {
            return;
        }
        if (!file.getName().endsWith(".ser")) {
            errorDialog("INCORRECT FILE EXTENSION!");
            return;
        }
        if (!file.exists()) {
            try {
                file.createNewFile();
            } catch (IOException e) {
                errorDialog("ERROR OCCURED WHILE CREATING FILE!");
                return;
            }
        }
        if (!file.canWrite()) {
            errorDialog("CAN\'T WRITE TO THAT FILE!");
            return;
        }

        updateMatrix();
        try (var oos = new ObjectOutputStream(new FileOutputStream(file))) {
            oos.writeObject(matrix);
        } catch (IOException e) {
            errorDialog("ERROR OCCURED WHILE WRITING TO THE FILE!");
        }
    }

    private void loadButtonActionPerformed(java.awt.event.ActionEvent evt) {
        var file = getFile("load");
        if (file == null) {
            return;
        }
        if (!file.getName().endsWith(".ser")) {
            errorDialog("INCORRECT FILE EXTENSION!");
            return;
        }
        if (!file.exists()) {
            errorDialog("FILE DOESN\'T EXIST!");
            return;
        }
        if (!file.canRead()) {
            errorDialog("CAN\'T READ THAT FILE!");
            return;
```

```
        }
        try (var ois = new ObjectInputStream(new FileInputStream(file))) {
            matrix = (Matrix<Integer>)ois.readObject();
            // if class ver was incorrect or
            // it was other class exception would be thrown
            updateTable();
        } catch (IOException e) {
            errorDialog("ERROR OCCURED WHILE READING THE FILE!");
        } catch (Exception e) {
            errorDialog(e.getMessage());
        }
    }

    private File getFile(String text) {
        var j = new JFileChooser(FileSystemView
                                 .getFileSystemView()
                                 .getHomeDirectory());
        var f = new FileNameExtensionFilter("ser", "ser");
        j.setFileFilter(f);
        j.setAcceptAllFileFilterUsed(false);
        var r = j.showDialog(null, text);
        if (r == JFileChooser.APPROVE_OPTION) {
            return j.getSelectedFile();
        }
        return null;
    }

    private Matrix<Integer> matrix = new Matrix<Integer>(4, 4, 0);
}
```

“Matrix.java”

```
package com.dazzlemon.oop16;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class Matrix <T> implements Serializable {
   private static final long serialVersionUID = 1L;//class v1

   public final int width;
   public final int height;
   private List<List<T>> matrix;


   public Matrix(int height, int width, T filler) {
         this.height = height;
         this.width  = width;

         this.matrix = new ArrayList<>(height);
         for (int i = 0; i < height; i++) {
         this.matrix.add(i, new ArrayList<>(width));
         for (int j = 0; j < width; j++) {
                  this.matrix.get(i).add(filler);
         }
         }
   }


   public void set(int i, int j, T val) {
         this.matrix.get(i).set(j, val);
   }


   public T get(int i, int j) {
         return this.matrix.get(i).get(j);
   }
}
```

“Solver.java”

```
package com.dazzlemon.oop16;

import java.awt.Point;

public class Solver {
   public static <T extends Comparable<T>> Point solve(Matrix<T> matrix) {
         int imax = 0;
         int jmax = 0;
         for (int i = 0; i < matrix.height; i++) {
         for (int j = 0; j < matrix.width; j++) {
                  if (matrix.get(i, j).compareTo(matrix.get(imax, jmax)) > 0) {
               imax = i;
               jmax = j;
                  }
         }
         }
         return new Point(imax, jmax);
   }

   public static <T> void swapRows(Matrix<T> matrix, int i1, int i2) {
      for (int j = 0; j < matrix.width; j++) {
         var tmp = matrix.get(i1, j);
         matrix.set(i1, j, matrix.get(i2, j));
         matrix.set(i2, j, tmp);
         }
   }

   public static <T> void swapColumns(Matrix<T> matrix, int j1, int j2) {
      for (int i = 0; i < matrix.height; i++) {
         var tmp = matrix.get(i, j1);
         matrix.set(i, j1, matrix.get(i, j2));
         matrix.set(i, j2, tmp);
         }
   }
}
```

Приклад роботи програми.

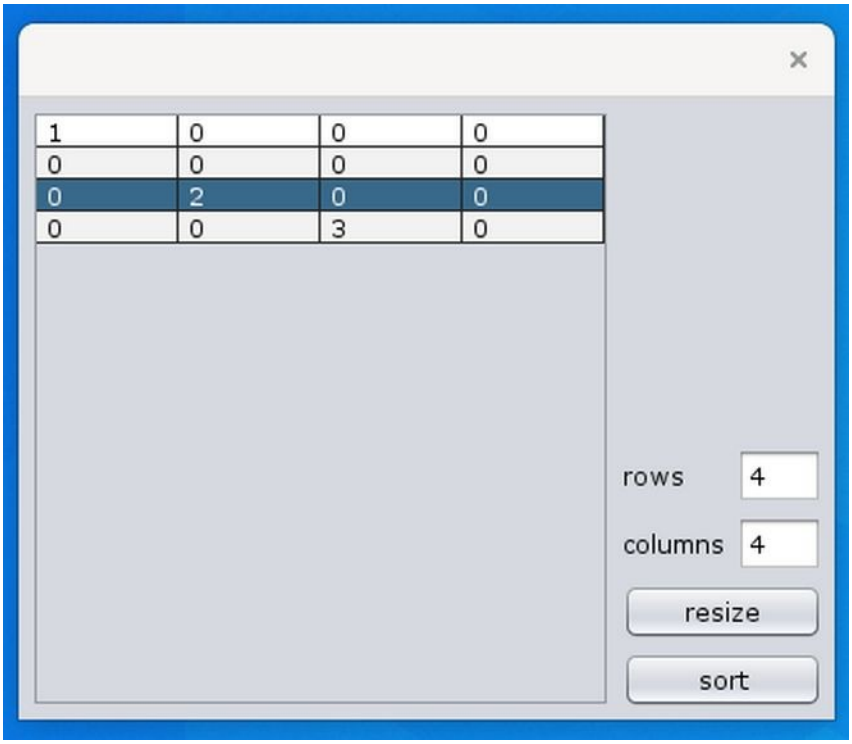


Рисунок 2(Тест 1. Підготовка)

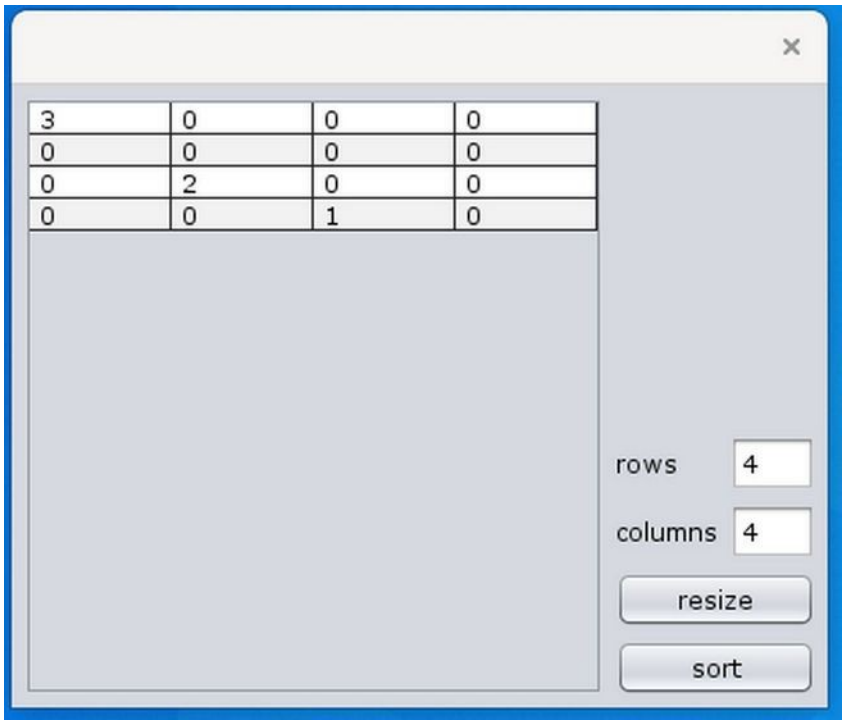


Рисунок 3(Тест 1. Стовпці та Рядки були замінені так, що найбільший єлемент знаходиться за індексами 0, 0)

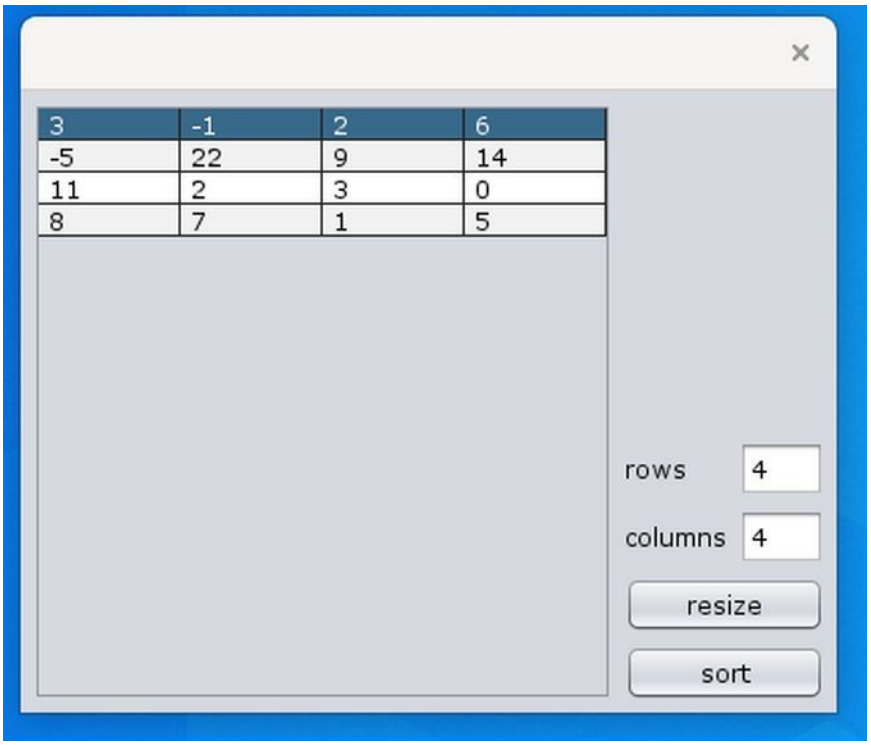


Рисунок 4(Тест 2. Підготовка)

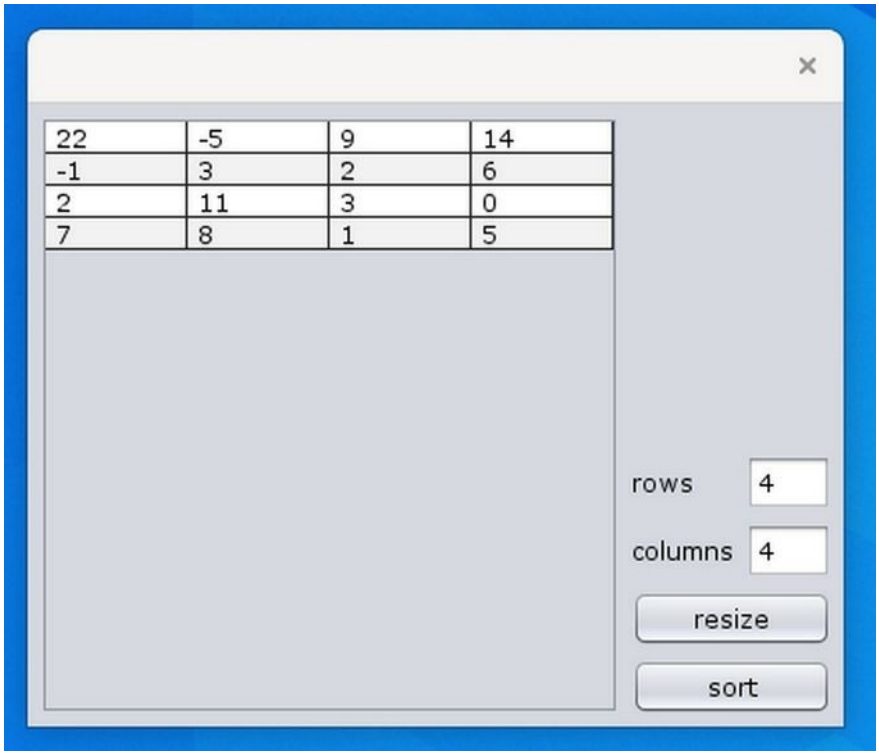


Рисунок 5(Тест 2. Найбільший елемент за індексом 0, 0)

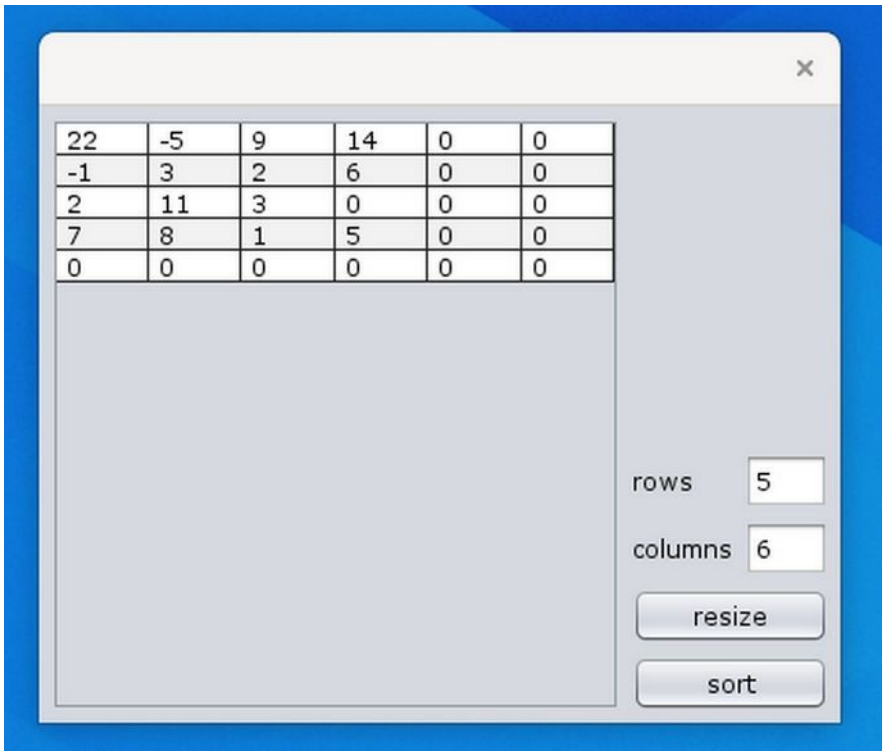


Рисунок 6(збільшення розміру)

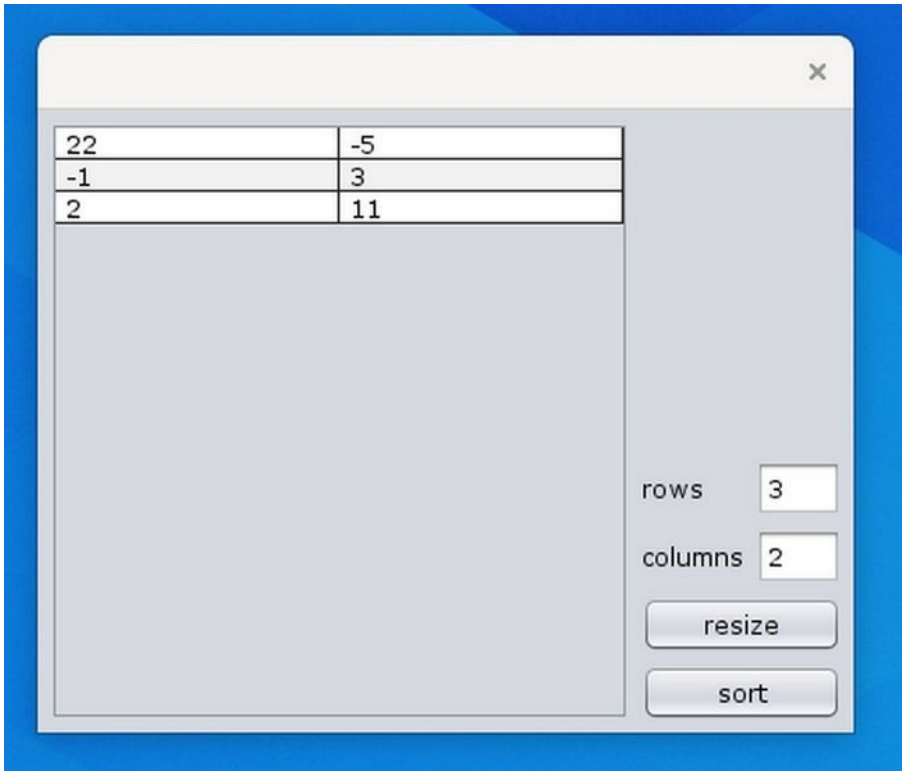


Рисунок 7(зменьшення розміру)

Перші 6 скріншотів були скопійовані з попередньої лабораторної роботи через те, що та частина функціоналу яку вони показують ніяк не змінилась.

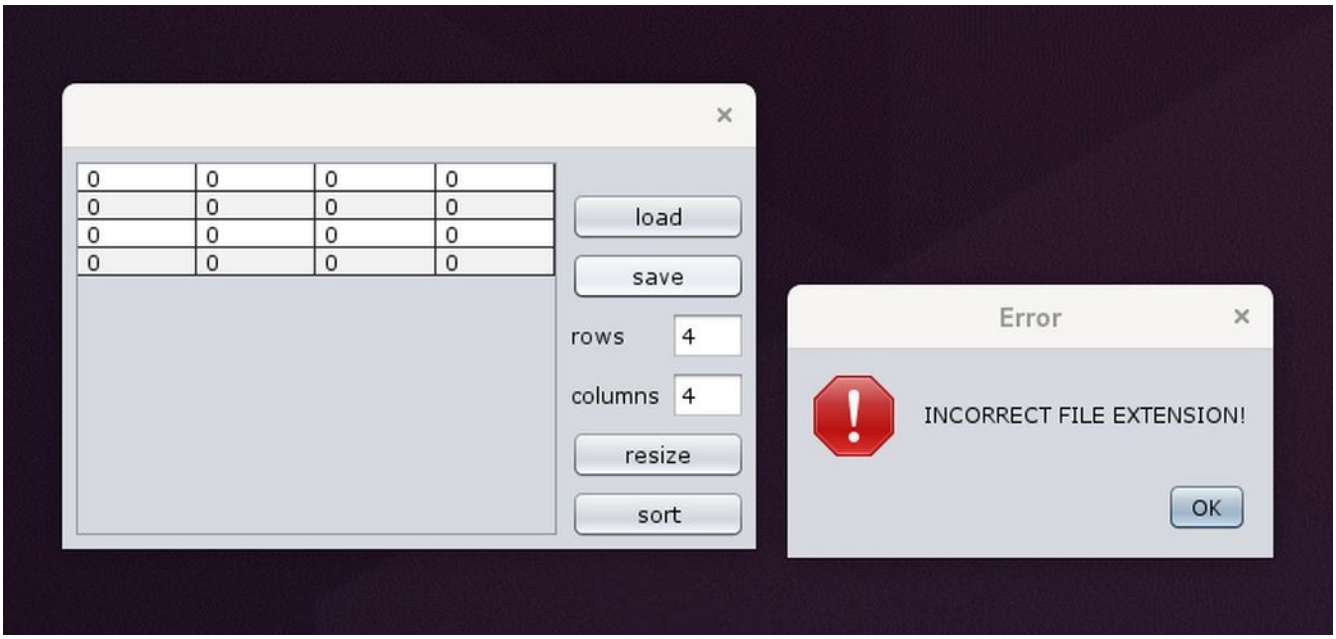


Рисунок 8(Приклад вікна помилки при зчитуванні або записі у файл)

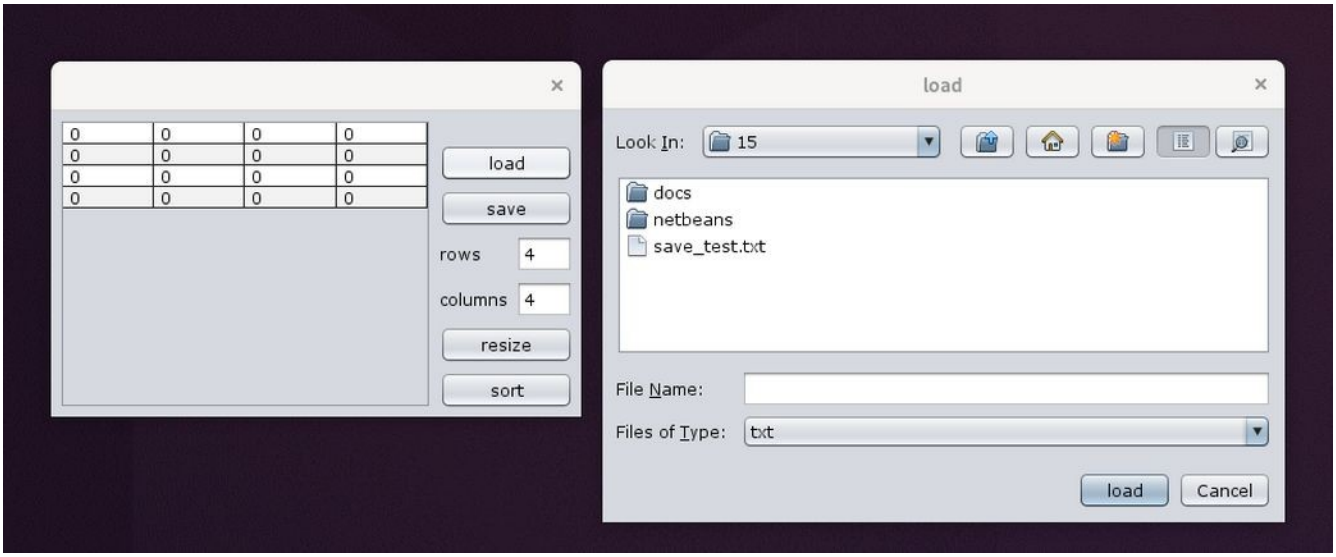


Рисунок 9(Приклад вікна вибору файлу)

Не були продемонстровані усі варіації помилок, та вибору файлів, бо їх було б забагато, а різниці між ними майже немає.

Використані старі скріншоти через те, що усі зміни непомітні для кінцевого користувача(окрім вибору файлу - там був змінений формат файлу з .txt на .ser).

Висновки.

Розроблена програма заснована на програмах із попередніх лабораторних робіт(13, 14, 15). Клас Solver ніяк не змінився з ЛР13, клас Matrix майже не змінився, був доданий функціонал серіалізації та десеріалізації, клас MatrixFileIO був видалений через непотрібність, клас MatrixGui майже не змінився, були змінені тільки обробники кнопок збереження та завантаження матриці з файлу, щоб використовувати класи ObjectInputStream та ObjectOutputStream замість MatrixFileIO.

Загалом такий спосіб збереження об’єктів є зручнішим ніж простий запис у файл через те, що автоматизується перевірка версій класу, і майже непотрібно розписувати логіку збереження об’єкту, натомість, якщо потрібна складна логіка збереження її можливо прописати самостійно хоча це і не є обов’язковим. У деякому сенсі серіалізація є аналогом json та xml. Але з деякими різницями - серіалізацію неможливо використовувати без класів, коли json можна використовувати для яких завгодно комбінацій простих даних, хоча в назві сказано, що він призначений для запису об’єктів(JavaScriptObjectNotation). XML також можна використовувати для запису яких завгодно даних.