Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

ОТЧЕТ

Вариант 20

к лабораторной работе по дисциплине:

|  |
| --- |
| Объектно–ориентированное программирование |
| “Разработка приложения с оконным графическим |
| Пользовательским интерфейсом” |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | АСУб-19-1 |  |  |  | Даниленко А.С. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В, |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск 2020 г.

**Содержание**

[**1 Постановка задачи** 3](#_Toc59052105)

[**2 Проектирование классов** 4](#_Toc59052106)

[**3 Проектирование графического интерфейса пользователя** 7](#_Toc59052107)

[**4 Таблица тестов** 8](#_Toc59052108)

[**5 Результаты тестирования** 9](#_Toc59052109)

[**6 Исходный код** 13](#_Toc59052110)

[Words: 13](#_Toc59052111)

[Calculator: 14](#_Toc59052112)

[Matrix: 15](#_Toc59052113)

[MainFrame: 18](#_Toc59052114)

[Controller: 21](#_Toc59052115)

[**Список использованных источников** 26](#_Toc59052116)

## **1 Постановка задачи**

1. Дана матрица. Необходимо сгенерировать новую матрицу, элементы не первой строки которой равны произведению соответствующих элементов исходной матрицы на первый элемент столбца.

2. Вычислите значение арифметического выражения. Два операнда и знак операции задаются в командной строке при запуске программы.

3. Выберите все слова из строки, содержащей слова, разделенные пробелами и запятыми, которые полностью состоят из одного символа.

Провести объектную декомпозицию задачи из индивидуального задания и разработать класс, содержащий соответствующие поля для хранения необходимых данных и методы, обеспечивающие достаточную для решения задачи функциональность класса. Разработать класс, реализующий текстовое меню, позволяющее осуществлять выбор вариантов выполнения индивидуального задания. Создание объекта класса решающего задание и вызов его методов должны осуществляться из класса реализующего меню. При выполнении задания предусмотреть методы для организации ввода и выводы данных как на консоль, так и в текстовые файлы. Работа с файлами может быть организована методами класса реализующего пользовательское меню, либо в рамках специально разработанного класса. Для всех проверяемых исключений должны быть реализованы обработчики, в которых выводится сообщение о возникновении исключительной ситуации и предпринимаются меры для продолжения работы программы в штатном режиме.

## **2 Проектирование классов**

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Проектирование классов | | | |
| **№** | **Название** | **Назначение** | **Поля (атрибуты)** |
| 1 | public Main | Главный класс | - |
| 2 | public Words | Класс 3 задания | private String line - строка  private String[] array - разбитая на слова строка |
| 3 | public Calculator | Класс 2 задания | private double a - первое число  private double b - втоое число  private String operation - знак операции |
| 4 | public Matrix | Класс 1 задания | private int height - высота матрицы  private int width - ширина матрицы  private int[][] matrix - матрица |
| 5 | package-private MainFrame | Вид | protected JComboBox<String> chooseTaskComboBox - поле выбора задания  protected JTextField calcFirstNumber - первое число в зад 2  protected JTextField calcSecondNumber - второе число в зад 2  protected JTextField calcOperation - операция в зад 2;  protected JTextField wordSentece - сторка в зад 3  protected JTextField matrHeight - высота матрицы в зад 1  protected JTextField matrWidth - ширина матрицы в зад 1  protected JTextField matrNumbers - матрица в зад 1  private JPanel fieldsPanel - панель, содержащая поля ввода  protected JButton doTaskButton - кнопка выполнения задания  protected JMenuItem saveButton - кнопка сохранения  protected JMenuItem loadButton - кнопка загрузки  protected JFileChooser fileChooser - выбор файла  protected JLabel resultSentenceTextLabel - результат |
| 6 | public Controller | Контроллер | private final MainFrame View - вид  private File file - кэш файла  private boolean isThereAnyFrames = false проверка, что открыто только одно доп окно  private Matrix firstTask = null - первое зад  private Calculator secondTask = null - второе зад  private Words thirdTask = null - третье зад |

**Main:**

Методы:

public static void main(String[] args) – начало программы и запуск меню.

**Words:**

Методы:

public void saveAs(String name) - сохранение объекта в файл

public String getLine() - геттер строки

public String doTask() - выполнить задание

public void setLine(String line) - сеттер строки

**Calculator:**

Методы:

public void setName(String compName) — сеттер названия компании

public double getA() - геттер первого числа

public double getB() - геттер второго числа

public String getOperation() - геттер операции

public void saveAs(String name) - сохранение объекта

public String answer() - выполнение задания

**Matrix:**

Методы:

public int getHeight() - геттер высоты матрицы

public int getWidth() - геттер ширины матрицы

public int[][] getMatrix() - геттер матрицы

public void saveAs(String name) - сохранения объекта

public String doTask() - выполнение задания

**MainFrame:**

Методы:

private void repaintFields() - повторный рендер полей

public boolean validateFields() - валидация полей

public boolean validateNumber(String text) - проверка, что вводимый текст – числа

public boolean validateSoloNumber(String text) - проверка, что вводимый текст - число

**Controller:**

Методы:

private void doTask() - выполнить задание

private void createDialog(JFrame mainFrame, String text, int width) - создать диалоговое окно

## **3 Проектирование графического интерфейса пользователя**



Рисунок 1 - Главное окно программы

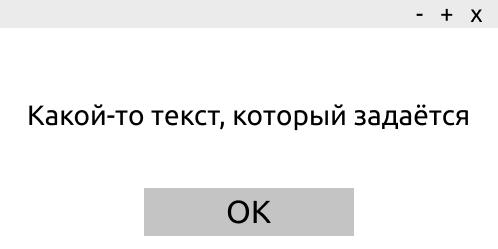


Рисунок 2 - Диалоговое окно программы

## **4 Таблица тестов**

Таблица 2 – Таблица тестов.

|  |  |  |
| --- | --- | --- |
| **№** | **Действие** | **Ожидаемый результат** |
| 1 | Выбор Задание 1 и решение | Решение Задания 1 |
| 2 | Выбор Задание 2 и решение | Решение Задания 2 |
| 3 | Выбор Задание 3 и решение | Решение Задания 3 |
| 4 | Вызвать ошибку ввода | Появление диалогового окна |
| 5 | Нажать на File - Save | Сохранение в файл |
| 6 | Нажать на File — Load | Загрузка из файла |
| 7 | Нажать на File — Exit | Завершение программы |

## **5 Результаты тестирования**

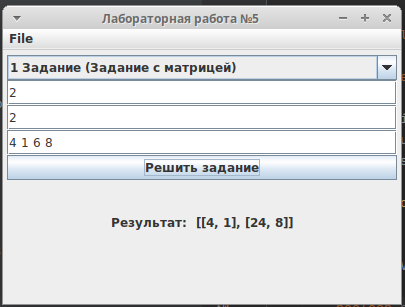


Рисунок 3 - Тест №1

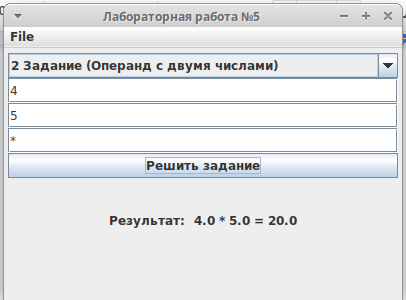


Рисунок 4 - Тест №2

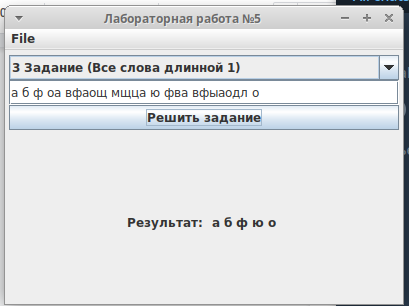


Рисунок 5 - Тест №3

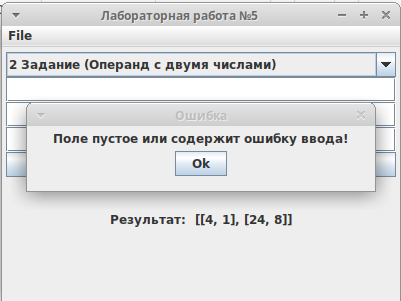


Рисунок 6 - Тест №4

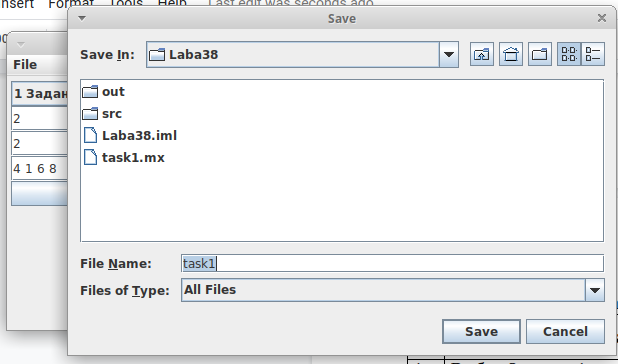


Рисунок 7 - Тест №5

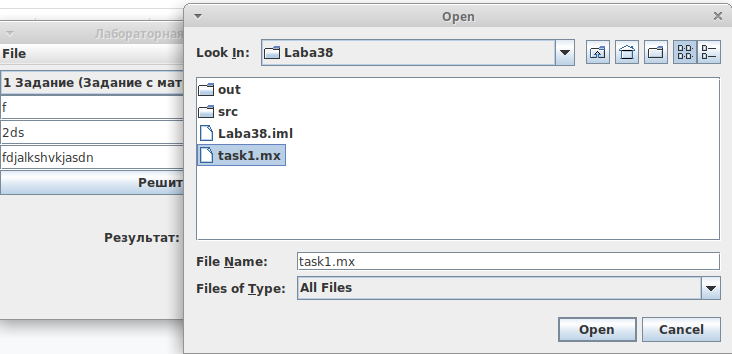


Рисунок 8 - Тест №5

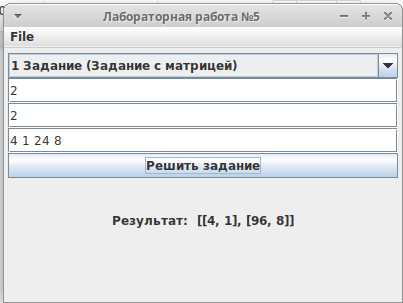


Рисунок 9 - Тест №5

## **6 Исходный код**

### Words:

package Lab\_3;  
import java.io.\*;  
public class Words implements Serializable  
{  
 private String line;  
 private String[] array;  
 public Words()  
 {  
 System.*out*.print("Введите обрабатываемые слова через пробел или запятую: ");  
 line = ExceptionHelper.*readString*();  
 doTask();  
 }  
 public Words(String sentence)  
 {  
 line = sentence;  
 }  
 public void saveAs(String name)  
 {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name + ".ws")))  
 {  
 oos.writeObject(this);  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println(exception.getMessage());  
 }  
 }  
 public String getLine()  
 {  
 return line;  
 }  
 public String doTask()  
 {  
 String[] buffer = line.split("[, ]");  
 String tempLine = "";  
 for (int i = 0; i < buffer.length; i++)  
 if (buffer[i].length() == 1)  
 tempLine = tempLine + buffer[i] + " ";  
 return tempLine;  
 }  
 public void print()  
 {  
 System.*out*.print("Все односимвольные слова: ");  
 for (int i = 0; i < array.length; i++)  
 System.*out*.print(array[i] + ' ');  
 System.*out*.println();  
 }  
 public boolean filled()  
 {  
 return line != null;  
 }  
 public void setLine(String line) {  
 this.line = line;  
 }  
}

### Calculator:

package Lab\_3;

import java.io.\*;

public class Calculator implements Serializable

{

private double a;

private double b;

private String operation;

public Calculator(double a, double b, String oper) {

this.a = a;

this.b = b;

this.operation = oper;

}

public double getA() {

return this.a;

}

public double getB() {

return b;

}

public String getOperation() {

return operation;

}

public Calculator()

{

System.out.print("Через пробел введите два рабочих числа: ");

double[] temp = ExceptionHelper.readDoubleArray(2);

a = temp[0];

b = temp[1];

System.out.print("Введите знак операции: ");

operation = ExceptionHelper.readString();

}

public Calculator(String name)

{

try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name)))

{

Calculator temp = (Calculator)ois.readObject();

System.out.print("Файл найден. Начинаю сборку.\n");

a = temp.a;

b = temp.b;

operation = temp.operation;

}

catch(Exception exception)

{

System.out.println("Объект с указанным именем не найден.");

}

}

public void saveAs(String name)

{

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name + "clc")))

{

oos.writeObject(this);

}

catch(Exception exception)

{

System.out.println(exception.getMessage());

}

}

public String answer()

{

switch (operation)

{

case "+":

return a + " " + operation + " " + b + " = " + (a + b) + '\n';

case "-":

return a + " " + operation + " " + b + " = " + (a - b) + '\n';

case "\*":

return a + " " + operation + " " + b + " = " + (a \* b) + '\n';

case "/":

return a + " " + operation + " " + b + " = " + (a / b) + '\n';

default:

return "Операция недействительна\n";

}

}

public boolean filled()

{

return operation != null;

}

}

### Matrix:

package Lab\_3;  
import java.io.\*;  
import java.util.Arrays;  
  
public class Matrix implements Serializable  
{  
 private int[][] matrix;  
 private int height = 0;  
 private int width = 0;  
 public Matrix()  
 {  
 System.*out*.print("Введите высоту матрицы: ");  
 height = ExceptionHelper.*readInt*();  
 while (height < 2)  
 {  
 System.*out*.print("Пожалуйста, введите число более 1: ");  
 height = ExceptionHelper.*readInt*();  
 }  
 System.*out*.print("Введите ширину матрицы: ");  
 width = ExceptionHelper.*readInt*();  
 while (width < 2)  
 {  
 System.*out*.print("Пожалуйста, введите число более 1: ");  
 width = ExceptionHelper.*readInt*();  
 }  
 System.*out*.println("Далее строка за строкой введите матрицу размерностью " + height + "x" + width);  
 matrix = new int[height][width];  
 for (int i = 0; i < height; i++)  
 {  
 int[] temp = ExceptionHelper.*readIntArray*(width);  
 for (int j = 0; j < width; j++)  
 matrix[i][j] = temp[j];  
 }  
 }  
 public Matrix(int height, int width, String matr)  
 {  
 this.height = height;  
 this.width = width;  
 String[] tempMatr = matr.split(" ");  
 matrix = new int[height][width];  
 int counter = 0;  
 for (int i = 0; i < height; i++)  
 {  
 for (int j = 0; j < width; j++)  
 {  
 matrix[i][j] = Integer.*parseInt*(tempMatr[counter]);  
 counter++;  
 }  
 }  
 }  
 public int getHeight() {  
 return height;  
 }  
 public int getWidth() {  
 return width;  
 }  
 public int[][] getMatrix() {  
 return matrix;  
 }  
 public void multiplyLine()  
 {  
 System.*out*.print("Выберите изменяемую строку(2-" + height + "): ");  
 int line = ExceptionHelper.*readInt*() - 1;  
 while (line < 1 || line + 1 > height)  
 {  
 System.*out*.print("Пожалуйста, введите число от 2 до " + height + ": ");  
 line = ExceptionHelper.*readInt*() - 1;  
 }  
 for (int i = 0; i < width; i++)  
 matrix[line][i] \*= matrix[0][i];  
 }  
 public void printMatrix()  
 {  
 System.*out*.print("Ваша матрица:\n");  
 for (int i = 0; i < height; i++)  
 {  
 for (int j = 0; j < width; j++)  
 System.*out*.print(matrix[i][j] + " ");  
 System.*out*.print('\n');  
 }  
 }  
 public void saveAs(String name)  
 {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name + ".mx")))  
 {  
 oos.writeObject(this);  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println(exception.getMessage());  
 }  
 }  
 public String doTask()  
 {  
 for (int i = 1; i < this.height; i++)  
 {  
 for (int j = 0; j < this.width; j++)  
 this.matrix[i][j] \*= this.matrix[0][j];  
 }  
 return Arrays.*deepToString*(matrix);  
 }  
 public boolean filled()  
 {  
 return matrix != null;  
 }  
}

### MainFrame:

package Interface;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.regex.Pattern;  
  
class MainFrame extends JFrame {  
  
 protected JComboBox<String> chooseTaskComboBox;  
  
 protected JTextField calcFirstNumber;  
 protected JTextField calcSecondNumber;  
 protected JTextField calcOperation;  
 protected JTextField wordSentece;  
 protected JTextField matrHeight;  
 protected JTextField matrWidth;  
 protected JTextField matrNumbers;  
  
 private JPanel fieldsPanel;  
  
 protected JButton doTaskButton;  
 protected JMenuItem saveButton;  
 protected JMenuItem loadButton;  
 protected JFileChooser fileChooser;  
 protected JLabel resultSentenceTextLabel;  
  
 protected MainFrame()  
 {  
 // Инициализация окна  
 super("Лабораторная работа №5");  
 this.setSize(400, 300);  
 this.setLocation(200, 200);  
 this.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 this.setLayout(new FlowLayout());  
  
 // Инициализация меню бара  
 var menuBar = new JMenuBar();  
 var menu = new JMenu("File");  
 saveButton = new JMenuItem("Save");  
 loadButton = new JMenuItem("Load");  
 JMenuItem exitButton = new JMenuItem("Exit");  
 exitButton.addActionListener(new ActionListener()  
 {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 System.*exit*(0);  
 }  
 });  
 menu.add(saveButton);  
 menu.add(loadButton);  
 menu.add(exitButton);  
 menuBar.add(menu);  
 this.setJMenuBar(menuBar);  
  
 fileChooser = new JFileChooser();  
 JPanel resultPanel = new JPanel();  
 resultPanel.setLayout(new FlowLayout());  
 fieldsPanel = new JPanel();  
 fieldsPanel.setLayout(new GridLayout(6, 1));  
  
 // Инициализация компонентов ввода  
 String[] choices = {"1 Задание (Задание с матрицей)", "2 Задание (Операнд с двумя числами)", "3 Задание (Все слова длиной 1)"};  
 chooseTaskComboBox = new JComboBox<>(choices);  
 chooseTaskComboBox.setPreferredSize(new Dimension(390, 24));  
 calcFirstNumber = new JTextField();  
 calcFirstNumber.setPreferredSize(new Dimension(90, 24));  
 calcFirstNumber.setToolTipText("Введите первое число");  
 calcSecondNumber = new JTextField();  
 calcSecondNumber.setPreferredSize(new Dimension(90, 24));  
 calcSecondNumber.setToolTipText("Введите второе число");  
 calcOperation = new JTextField();  
 calcOperation.setPreferredSize(new Dimension(90, 24));  
 calcOperation.setToolTipText("Введите операнд");  
 wordSentece = new JTextField();  
 wordSentece.setPreferredSize(new Dimension(390, 24));  
 wordSentece.setToolTipText("Введите слова, разделённые пробелами или запятыми");  
 matrHeight = new JTextField();  
 matrHeight.setPreferredSize(new Dimension(90, 24));  
 matrHeight.setToolTipText("Введите высоту матрицы");  
 matrWidth = new JTextField();  
 matrWidth.setPreferredSize(new Dimension(90, 24));  
 matrWidth.setToolTipText("Введите ширину матрицы");  
 matrNumbers = new JTextField();  
 matrNumbers.setPreferredSize(new Dimension(180, 24));  
 matrNumbers.setToolTipText("Введите матрицу");  
  
  
 JLabel resultSentenceLabel = new JLabel("Результат: ");  
 resultSentenceTextLabel = new JLabel("Нет результата");  
  
 doTaskButton = new JButton("Решить задание");  
  
 chooseTaskComboBox.setSelectedIndex(0);  
 repaintFields();  
  
 resultPanel.add(resultSentenceLabel);  
 resultPanel.add(resultSentenceTextLabel);  
  
 chooseTaskComboBox.addActionListener(new ActionListener()  
 {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 repaintFields();  
 }  
 });  
  
 // Добавление компонентов в окно  
 this.add(fieldsPanel);  
 this.add(resultPanel);  
 }  
  
 private void repaintFields()  
 {  
 switch (chooseTaskComboBox.getSelectedIndex())  
 {  
 case 0:  
 {  
 fieldsPanel.removeAll();  
 fieldsPanel.add(chooseTaskComboBox);  
 fieldsPanel.add(matrHeight);  
 fieldsPanel.add(matrWidth);  
 fieldsPanel.add(matrNumbers);  
 fieldsPanel.add(doTaskButton);  
 this.validate();  
 this.repaint();  
 break;  
 }  
 case 1:  
 {  
 fieldsPanel.removeAll();  
 fieldsPanel.add(chooseTaskComboBox);  
 fieldsPanel.add(calcFirstNumber);  
 fieldsPanel.add(calcSecondNumber);  
 fieldsPanel.add(calcOperation);  
 fieldsPanel.add(doTaskButton);  
 this.validate();  
 this.repaint();  
 break;  
 }  
 case 2:  
 {  
 fieldsPanel.removeAll();  
 fieldsPanel.add(chooseTaskComboBox);  
 fieldsPanel.add(wordSentece);  
 fieldsPanel.add(doTaskButton);  
 this.validate();  
 this.repaint();  
 break;  
 }  
 }  
 }  
  
 // Валидация полей ввода  
 public boolean validateFields()  
 {  
 switch (chooseTaskComboBox.getSelectedIndex())  
 {  
 case 0:  
 {  
 if  
 (  
 !validateSoloNumber(matrHeight.getText()) || !validateSoloNumber(matrWidth.getText())  
 || !validateNumber(matrNumbers.getText()) || matrHeight.getText().isEmpty()  
 || matrWidth.getText().isEmpty() || matrNumbers.getText().isEmpty()  
 )  
 return false;  
 break;  
 }  
 case 1:  
 {  
 if  
 (  
 calcFirstNumber.getText().isEmpty() || calcSecondNumber.getText().isEmpty()  
 || calcOperation.getText().isEmpty() || !validateSoloNumber(calcFirstNumber.getText())  
 || !validateSoloNumber(calcSecondNumber.getText())  
 )  
 return false;  
 break;  
 }  
 case 2:  
 {  
 if (wordSentece.getText().isEmpty())  
 return false;  
 break;  
 }  
 }  
 return true;  
 }  
 public boolean validateNumber(String text)  
 {  
 return Pattern.*matches*("[\\d\\s]+", text);  
 }  
 public boolean validateSoloNumber(String text)  
 {  
 return Pattern.*matches*("[\\d\\s]", text);  
 }  
}

### Controller:

package Interface;  
  
import Lab\_3.\*;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.\*;  
  
public class Controller  
{  
 private final MainFrame View = new MainFrame();  
 private File file;  
 private boolean isThereAnyFrames = false;  
 private Matrix firstTask = null;  
 private Calculator secondTask = null;  
 private Words thirdTask = null;  
 public Controller()  
 {  
 View.setVisible(true);  
 boolean exit = true;  
 // Добавляем действие кнопки "Решить задание"  
 View.doTaskButton.addActionListener(new ActionListener()  
 {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent)  
 {  
 if (!View.validateFields())  
 {  
 isThereAnyFrames = true;  
 createDialog(View, "Поле пустое или содержит ошибку ввода!", 350);  
 return;  
 }  
 doTask();  
 }  
 });  
 // действие кнопки сохранения  
 View.saveButton.addActionListener(new ActionListener()  
 {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent)  
 {  
 switch (View.chooseTaskComboBox.getSelectedIndex())  
 {  
 case 0:  
 {  
 if (firstTask != null)  
 {  
 View.fileChooser.showSaveDialog(View);  
 firstTask.saveAs(View.fileChooser.getSelectedFile().getAbsolutePath());  
 }  
 else  
 {  
 isThereAnyFrames = true;  
 createDialog(View, "Задание не было выполнено!", 200);  
 }  
 break;  
 }  
 case 1:  
 {  
 if (secondTask != null)  
 {  
 View.fileChooser.showSaveDialog(View);  
 secondTask.saveAs(View.fileChooser.getSelectedFile().getAbsolutePath());  
 }  
 else  
 {  
 isThereAnyFrames = true;  
 createDialog(View, "Задание не было выполнено!", 200);  
 }  
 break;  
 }  
 case 2:  
 {  
 if (thirdTask != null)  
 {  
 View.fileChooser.showSaveDialog(View);  
 thirdTask.saveAs(View.fileChooser.getSelectedFile().getAbsolutePath());  
 }  
 else  
 {  
 isThereAnyFrames = true;  
 createDialog(View, "Задание не было выполнено!", 200);  
 }  
 break;  
 }  
 }  
 }  
 });  
 // Действие кнопки загрузки  
 View.loadButton.addActionListener(new ActionListener()  
 {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent)  
 {  
 View.fileChooser.showOpenDialog(View);  
 file = View.fileChooser.getSelectedFile();  
 switch (View.chooseTaskComboBox.getSelectedIndex())  
 {  
 case 0:  
 {  
 if (!file.getAbsolutePath().endsWith(".mx"))  
 {  
 isThereAnyFrames = true;  
 createDialog(View, "Неправильный тип файла", 150);  
 return;  
 }  
 try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file.getAbsolutePath())))  
 {  
 firstTask = (Matrix) ois.readObject();  
 View.matrHeight.setText(String.*valueOf*(firstTask.getHeight()));  
 View.matrWidth.setText(String.*valueOf*(firstTask.getWidth()));  
 int[][] temp = firstTask.getMatrix();  
 String tempLine = "";  
 for (int i = 0; i < firstTask.getHeight(); i++)  
 for (int j = 0; j < firstTask.getWidth(); j++)  
 tempLine = tempLine + temp[i][j] + " ";  
 View.matrNumbers.setText(tempLine);  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println(exception.getMessage());  
 }  
 break;  
 }  
 case 1:  
 {  
 if (!file.getAbsolutePath().endsWith(".clc"))  
 {  
 isThereAnyFrames = true;  
 createDialog(View, "Неправильный тип файла", 150);  
 return;  
 }  
 try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file.getAbsolutePath())))  
 {  
 secondTask = (Calculator) ois.readObject();  
 View.calcFirstNumber.setText(String.*valueOf*(secondTask.getA()));  
 View.calcSecondNumber.setText(String.*valueOf*(secondTask.getB()));  
 View.calcOperation.setText(secondTask.getOperation());  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println(exception.getMessage());  
 }  
 break;  
 }  
 case 2:  
 {  
 if (!file.getAbsolutePath().endsWith(".ws"))  
 {  
 isThereAnyFrames = true;  
 createDialog(View, "Неправильный тип файла", 150);  
 return;  
 }  
 try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file.getAbsolutePath())))  
 {  
 thirdTask = (Words) ois.readObject();  
 View.wordSentece.setText(thirdTask.getLine());  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println(exception.getMessage());  
 }  
 break;  
 }  
 }  
 }  
 });  
 }  
// Выполнение задания в зависимости от выбранного задания  
 private void doTask()  
 {  
 switch (View.chooseTaskComboBox.getSelectedIndex())  
 {  
 case 0:  
 {  
 firstTask = new Matrix  
 (  
 Integer.*parseInt*(View.matrHeight.getText()),  
 Integer.*parseInt*(View.matrWidth.getText()),  
 View.matrNumbers.getText()  
 );  
 View.resultSentenceTextLabel.setText(firstTask.doTask());  
 break;  
 }  
 case 1:  
 {  
 secondTask = new Calculator  
 (  
 Double.*parseDouble*(View.calcFirstNumber.getText()),  
 Double.*parseDouble*(View.calcSecondNumber.getText()),  
 View.calcOperation.getText()  
 );  
 View.resultSentenceTextLabel.setText(secondTask.answer());  
 break;  
 }  
 case 2:  
 {  
 thirdTask = new Words(View.wordSentece.getText());  
 View.resultSentenceTextLabel.setText(thirdTask.doTask());  
 break;  
 }  
 }  
 }  
 // Создание диалогового окна  
 private void createDialog(JFrame mainFrame, String text, int width)  
 {  
 final JDialog temp = new JDialog(mainFrame, "Ошибка");  
 temp.add(new JLabel(text));  
 temp.setLayout(new FlowLayout());  
 JButton okBtn = new JButton("Ok");  
 temp.add(okBtn);  
 okBtn.addActionListener(new ActionListener()  
 {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent)  
 {  
 isThereAnyFrames = false;  
 temp.dispose();  
 }  
 });  
 temp.setLocationRelativeTo(mainFrame);  
 temp.setLocation(temp.getX() - width / 2, temp.getY() - 50);  
 temp.setSize(width, 90);  
 temp.setVisible(true);  
 }  
}

## **Список использованных источников**

1. Аршинский В.Л. Объектно-ориентированное программирование. Метод. указания по выполнению лабораторных работ / В.Л. Аршинский. – Иркутск: Изд-во ИРНИТУ, 2015. – 23 c.