Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

|  |
| --- |
| Институт информационных технологий и анализа данных |

наименование института

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

по дисциплине:

|  |
| --- |
| **ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ Программирование** |
| **«Разработка приложения с оконным графическим** |
| **пользовательским интерфейсом»** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил | АСУб-20-2 |  |  |  | Арбакова А.В. |
|  | шифр группы |  | подпись |  | Фамилия И.О. |
| Проверил |  |  |  |  | Бакшеева Е.Н. |
|  | должность |  | подпись |  | Фамилия И.О. |

Иркутск 2021 г.

**Содержание**

[1. Постановка задачи 3](#_Toc134478866)

[2. Проектирование графического интерфейса пользователя 3](#_Toc134478867)

[3. Проектирование классов 6](#_Toc134478868)

[4. Таблица тестов 6](#_Toc134478869)

[5. Результаты тестирования 8](#_Toc134478870)

[6. Исходный код 11](#_Toc134478871)

[7. Список использованных источников 16](#_Toc134478872)

1. **Постановка задачи**

Для выполнения задания необходимо разработать графический пользовательский интерфейс средствами java для работы с классами. Графический пользовательский интерфейс должен предоставлять возможности работы со всеми возможными функциями программы (ввод и вывод данных в соответствующей форме, изменение условий выполнения задания, работа с файлами и т.д.). Состав используемых компонентов определяется самостоятельно, но для них должны быть предусмотрены соответствующие планировщики раскладки компонентов в основных контейнерах. Классы, реализующие графический пользовательский интерфейс, должны содержать только обращения к методам классов для выполнения заданий.

Задачи из 1 лабораторной работы по 1 варианту:

1. Вычислите сумму диагоналей квадратной матрицы.

2. Определите, является ли заданная строка символов числом.

3. Выполнить сложение двух матриц.

1. **Проектирование графического интерфейса пользователя**

При запуске приложения открывается «Главное меню» (рис. 1), содержащее следующие пункты:

1. Задание 1. Вычислить сумму диагоналей матрицы.
2. Задание 2. Является ли строка числом.
3. Задание 3. Вычислить сумму матриц.

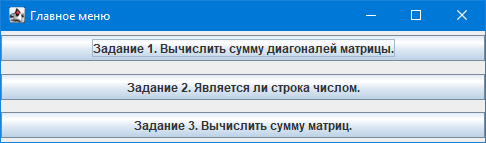


Рисунок 1 – Интерфейс главного меню

При выборе первого пункта, выполняющего задание 1, открывается окно «Задание 1» (рис. 2), в котором следует ввести размерность матрицы и нажать на кнопку «Построить матрицу». После нажатия на эту кнопку, происходит случайное заполнение матрицы и вычисление сумм диагоналей – главной и побочной. При нажатии кнопки «Вернуться в меню» происходит возращение в окно «Главное меню» (рис. 1).

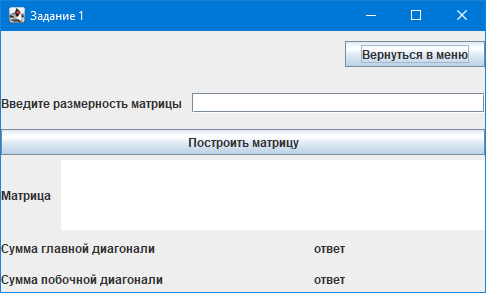


Рисунок 2 – Интерфейс окна «Задание 1»

При выборе второго пункта, выполняющего задание 2, открывается окно «Задание 2» (рис. 3), в котором следует ввести строку и нажать на кнопку «Вывести ответ». После нажатия на эту кнопку, происходит определение строки на соответствие числу. При нажатии кнопки «Вернуться в меню» происходит возращение в окно «Главное меню» (рис. 1).

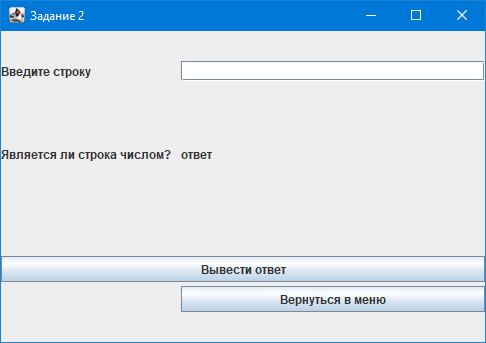


Рисунок 3 – Интерфейс окна «Задание 2»

При выборе третьего пункта, выполняющего задание 3, открывается окно «Задание 3» (рис. 4), в котором следует ввести размерность матрицы и нажать на кнопку «Построить таблицы». После нажатия на эту кнопку, происходит случайное заполнение матриц А и В и вычисление суммы матриц. При нажатии кнопки «Вернуться в меню» происходит возращение в окно «Главное меню» (рис. 1).

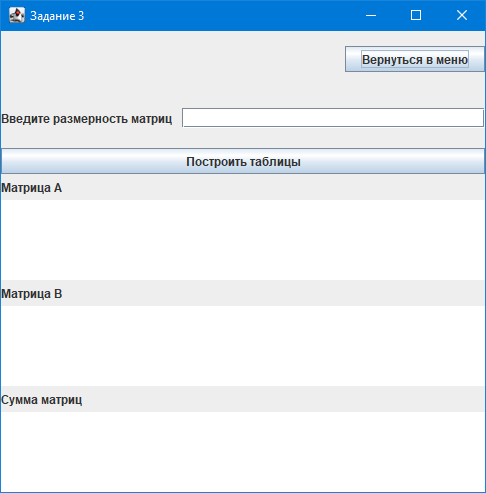


Рисунок 4 – Интерфейс окна «Задание 3»

При некорректном вводе значения размерности матрицы, всплывает окно, оповещающее о наличии ошибке в вводе (рис. 5).

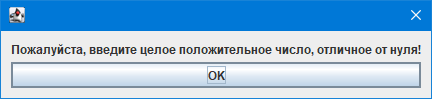


Рисунок 5 – Интерфейс окна, оповещающее об ошибке ввода

1. **Проектирование классов**

Таблица 1 – Классы.

|  |  |
| --- | --- |
| lab5 | Главный класс, из которого вызываются остальные классы |
| MainMenu | Класс-меню |
| Task1 | Класс, представляющий графический вид 1-го задания |
| Task 2 | Класс, представляющий графический вид 2-го задания |
| Task 3 | Класс, представляющий графический вид 3-го задания |
| Matrix | Класс, описывающий свойства матрицы и её функции и использующийся для заданий 1 и 3 |
| Sentence | Класс, использующийся во 2-м задании для проверки строки |

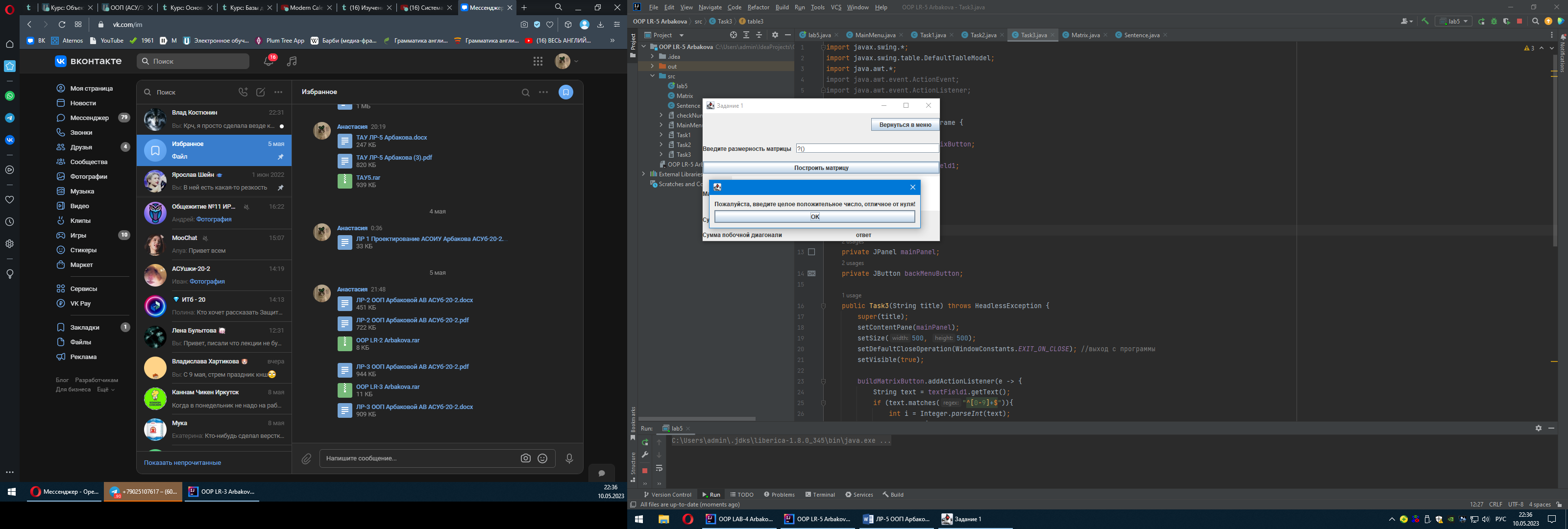


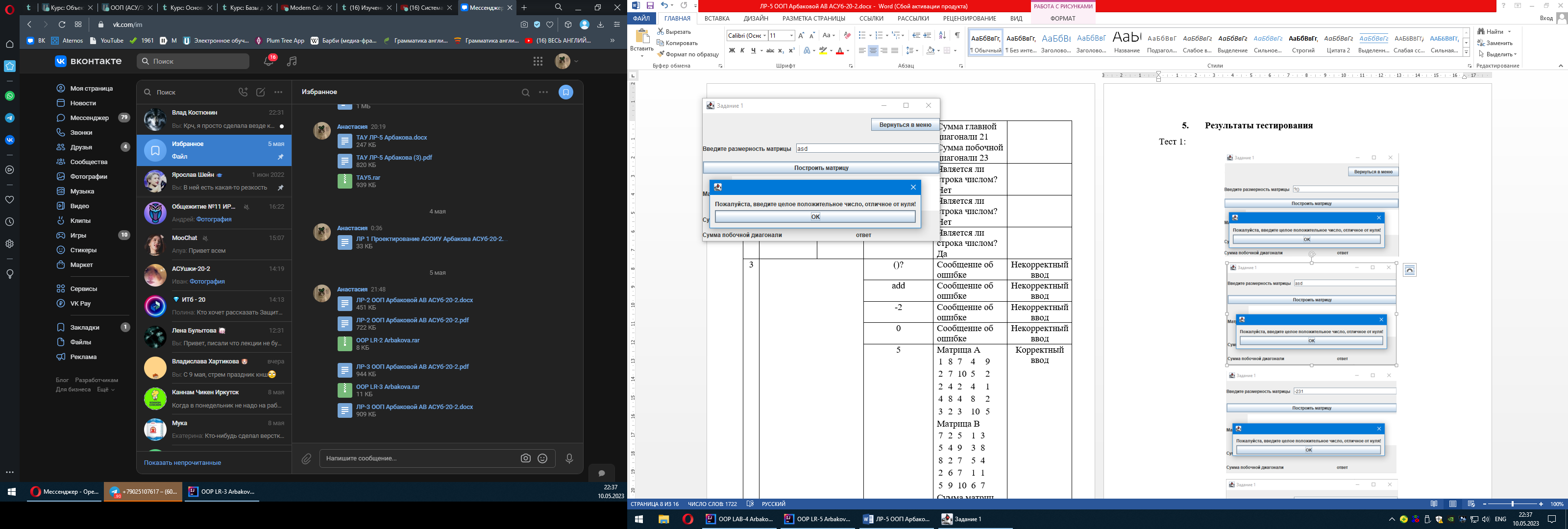
1. **Таблица тестов**

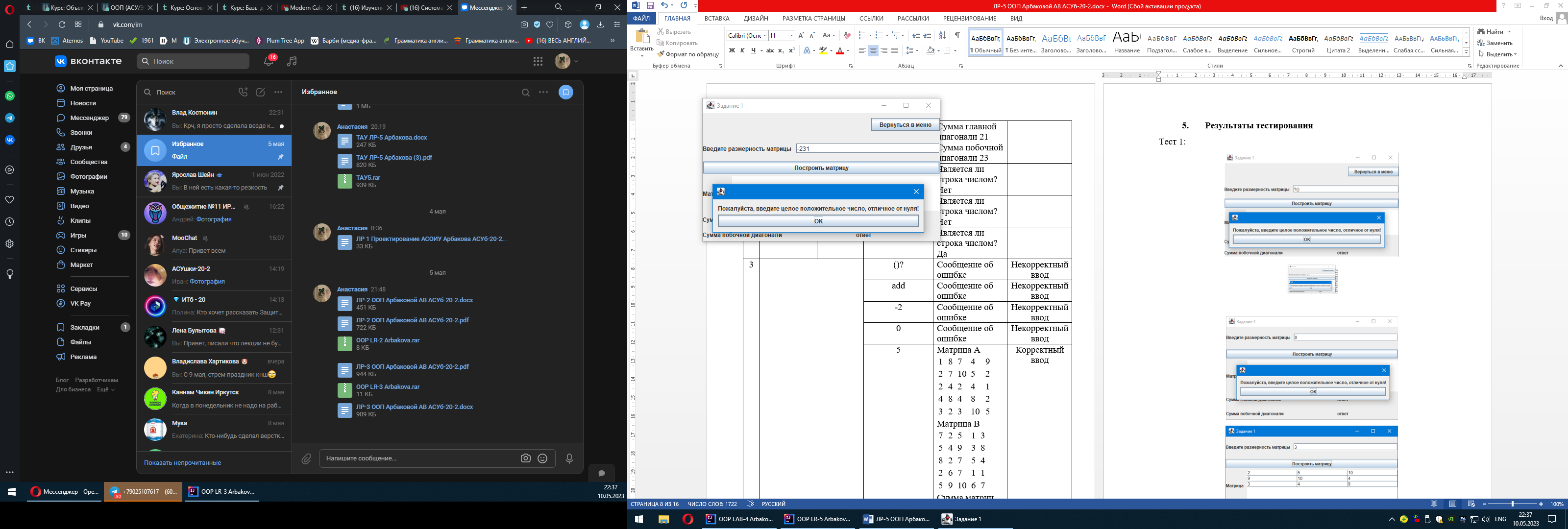
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Вводимые данные | | | Результат | Примечание |
| Размерность матрицы | Строка | Размерность матриц |
| 1 | ?() |  | | Сообщение об ошибке | Некорректный ввод |
| asd | Сообщение об ошибке | Некорректный ввод |
| -231 | Сообщение об ошибке | Некорректный ввод |
| 0 | Сообщение об ошибке | Некорректный ввод |
| 3 | Матрица   |  |  |  | | --- | --- | --- | | 2 | 3 | 1 | | 5 | 3 | 5 | | 7 | 4 | 2 |   Сумма главной диагонали 7  Сумма побочной диагонали 11 | Корректный ввод |
| 2 |  | ?\*( |  | Является ли строка числом? Нет |  |
| asd | Является ли строка числом? Нет |  |
| 3123 | Является ли строка числом? Да |  |
| 3 |  | | ()? | Сообщение об ошибке | Некорректный ввод |
| add | Сообщение об ошибке | Некорректный ввод |
| -2 | Сообщение об ошибке | Некорректный ввод |
| 0 | Сообщение об ошибке | Некорректный ввод |
| 5 | Матрица А   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 4 | 10 | 3 | 4 | 7 | | 5 | 4 | 2 | 1 | 6 | | 5 | 4 | 6 | 3 | 9 | | 4 | 1 | 6 | 7 | 7 | | 7 | 4 | 2 | 5 | 10 |   Матрица В   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 8 | 4 | 1 | 8 | 6 | | 8 | 6 | 1 | 1 | 2 | | 6 | 2 | 6 | 6 | 3 | | 1 | 5 | 2 | 4 | 1 | | 8 | 8 | 3 | 3 | 1 |   Сумма матриц   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 12 | 14 | 4 | 12 | 13 | | 13 | 10 | 3 | 2 | 8 | | 11 | 6 | 12 | 9 | 12 | | 5 | 6 | 8 | 11 | 8 | | 15 | 12 | 5 | 8 | 11 | | Корректный ввод |

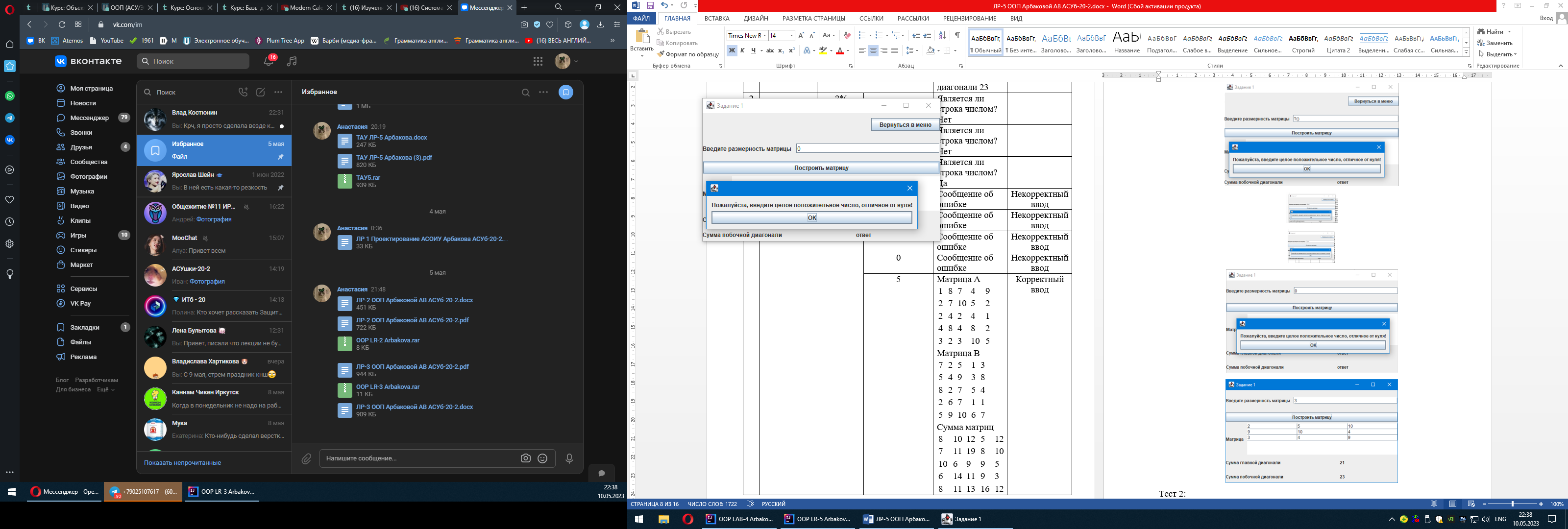
1. **Результаты тестирования**

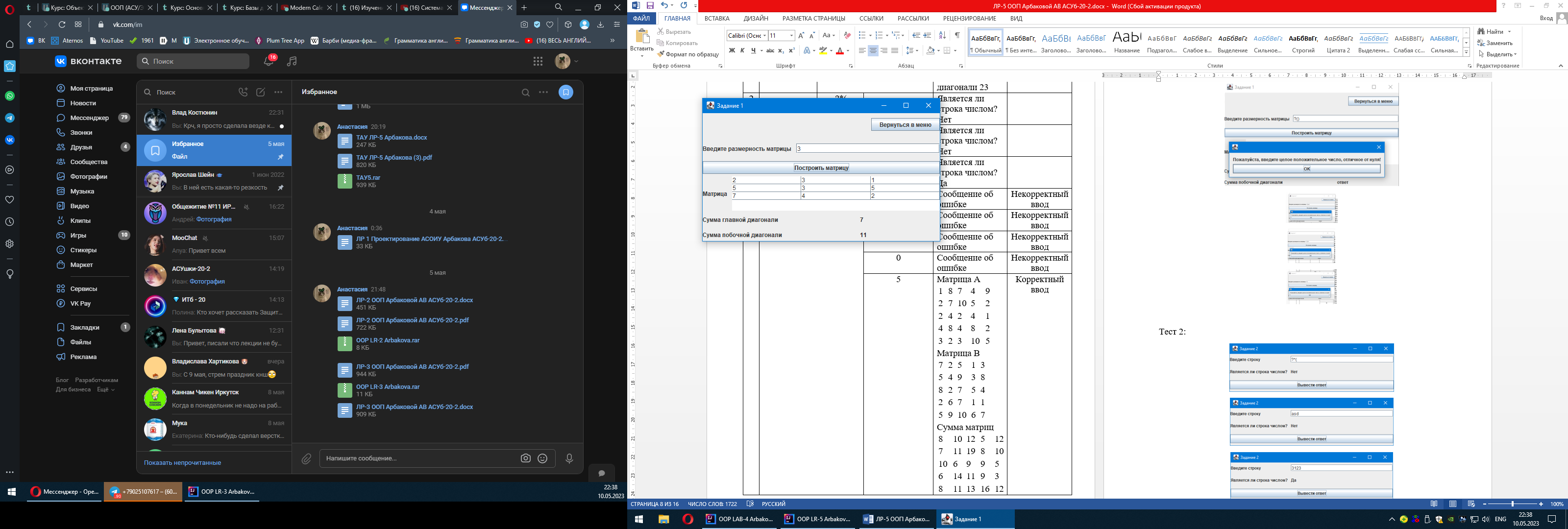
Тест 1:



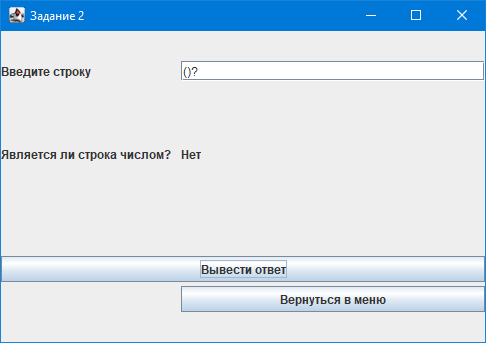


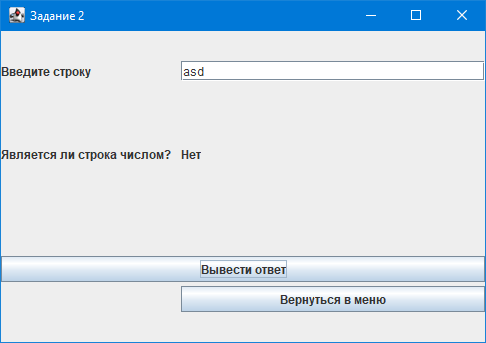


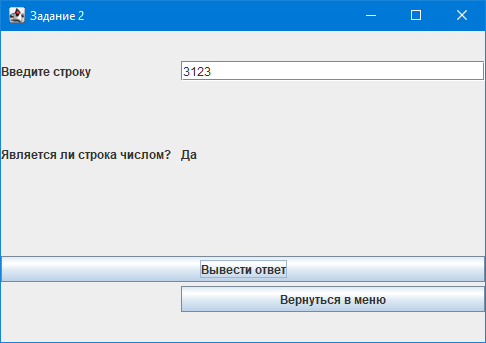




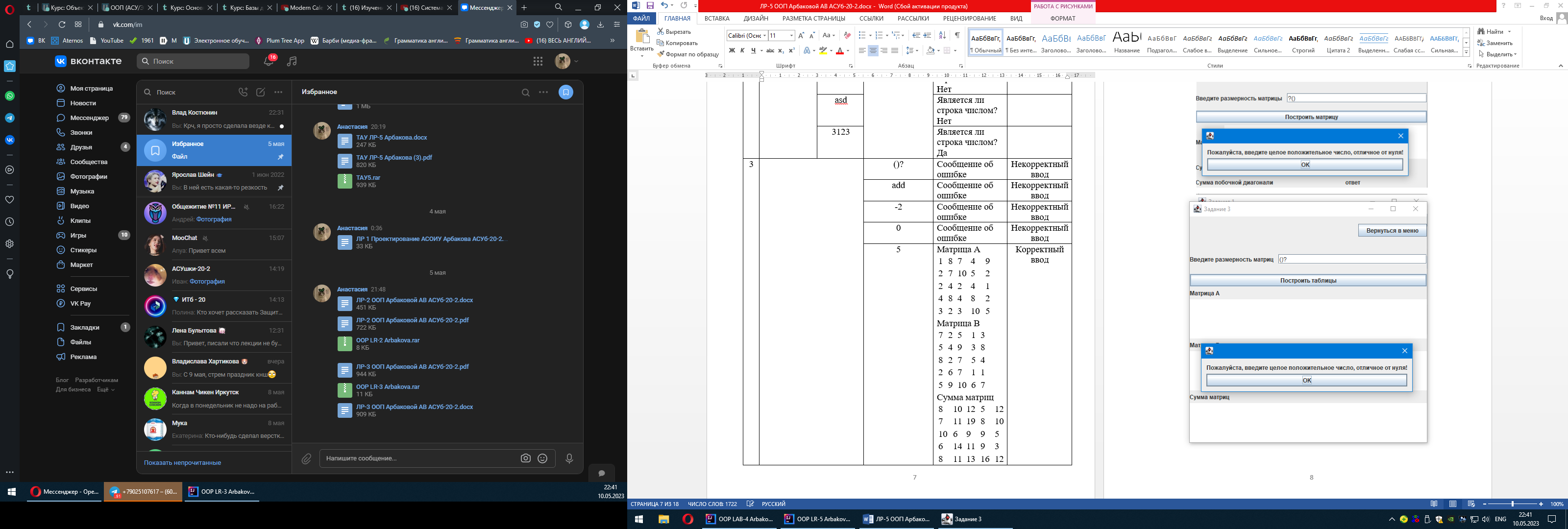
Тест 2:

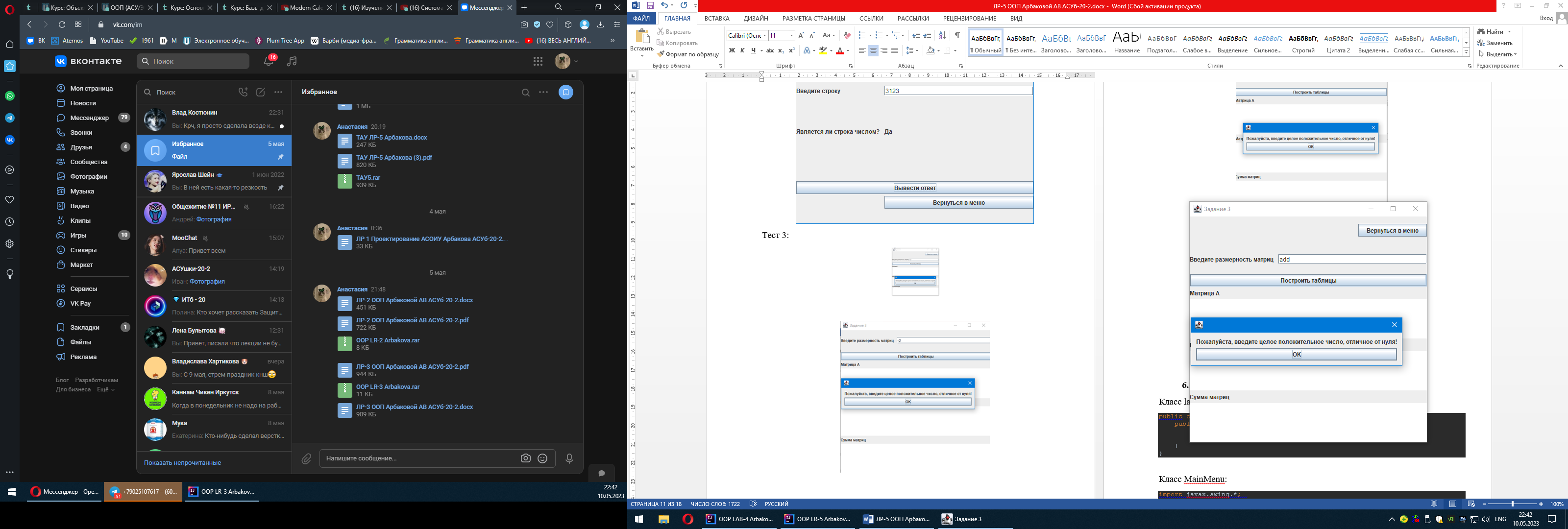


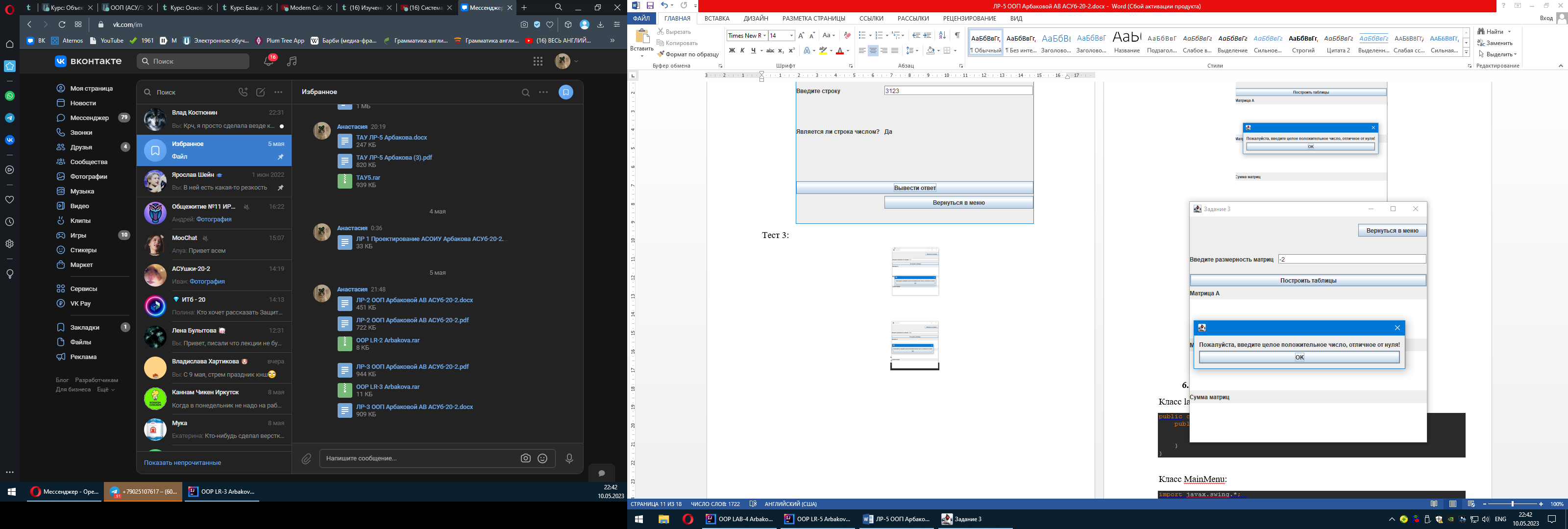


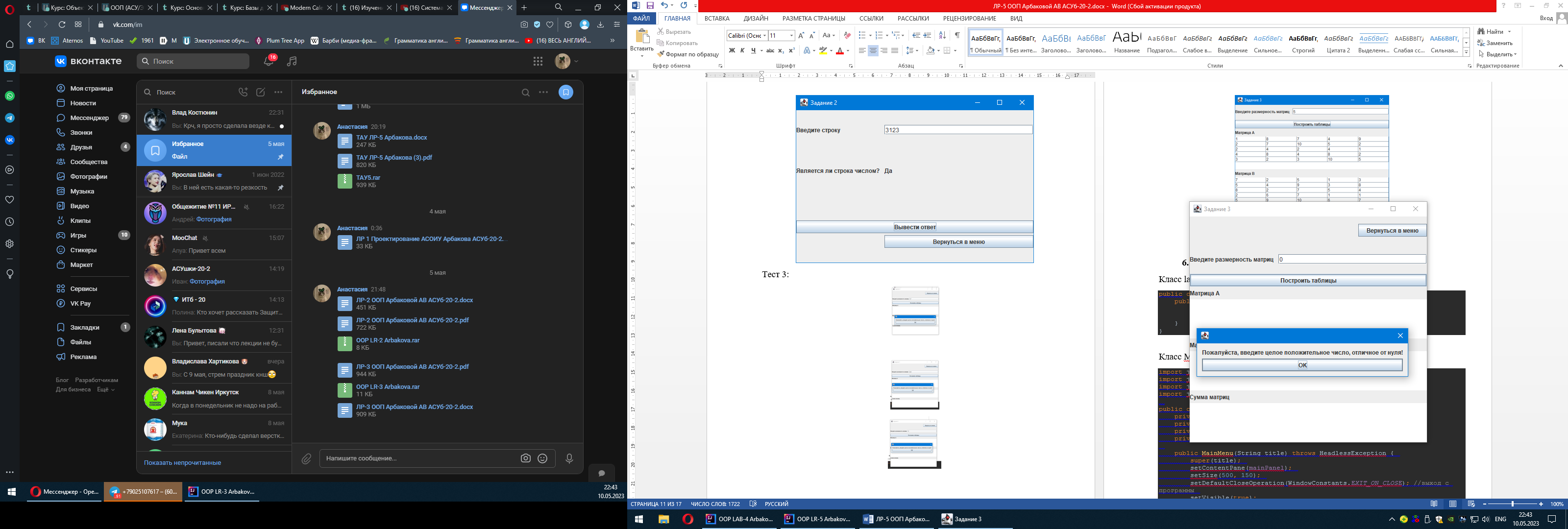


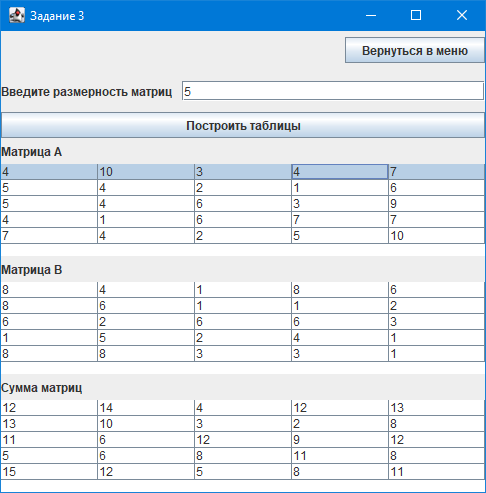
Тест 3:











1. **Исходный код**

Класс lab5:

public class lab5 {  
 public static void main(String[] args) {  
 MainMenu menu = new MainMenu("Главное меню");  
  
 }  
}

Класс MainMenu:

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class MainMenu extends JFrame{  
 private JButton task1Button;  
 private JButton task2Button;  
 private JButton task3Button;  
 private JPanel mainPanel;  
  
 public MainMenu(String title) throws HeadlessException {  
 super(title);  
 setContentPane(mainPanel);  
 setSize(500, 150);  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*); //выход с программы  
 setVisible(true);  
  
 task1Button.addActionListener(e -> {  
 Task1 task = new Task1("Задание 1");  
 dispose();  
 });  
 task2Button.addActionListener(e -> {  
 Task2 task = new Task2("Задание 2");  
 dispose();  
 });  
 task3Button.addActionListener(e -> {  
 Task3 task = new Task3("Задание 3");  
 dispose();  
 });  
 }  
}

Класс Task1:

import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class Task1 extends JFrame{  
 private JTextField textField1;  
 private JTable table1;  
 private JButton buildMatrixButton;  
 private JPanel mainPanel;  
 private JLabel Answer1;  
 private JLabel Answer2;  
 private JButton backMenuButton;  
  
 public Task1(String title) throws HeadlessException {  
 super(title);  
 setContentPane(mainPanel);  
 setSize(500, 300);  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*); //выход с программы  
  
  
 setVisible(true);  
  
 buildMatrixButton.addActionListener(e -> {  
 String text = textField1.getText();  
 if (text.matches("^[0-9]+$")){  
 int i = Integer.*parseInt*(text);  
 if (i!=0){  
 Matrix matrix = new Matrix(i);  
 String[][] matrixString = new String[i][i];  
 for (int j = 0; j < i; j++) {  
 for (int k = 0; k < i; k++) {  
 matrixString[j][k] = Integer.*toString*(matrix.matrix[j][k]);  
 }  
 }  
 table1.setModel(new DefaultTableModel(matrixString, matrixString[0]));  
  
 Answer1.setText(Integer.*toString*(matrix.sumMainDiagonal()));  
 Answer2.setText(Integer.*toString*(matrix.sumSideDiagonal()));  
 } else new checkNumber();  
 } else new checkNumber();  
 });  
 backMenuButton.addActionListener(e -> {  
 MainMenu menu = new MainMenu("Главное меню");  
 dispose();  
 });  
 }  
}

Класс Task2:

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class Task2 extends JFrame {  
 private JTextField textField1;  
 private JButton getAnswerButton;  
 private JPanel mainPanel;  
 private JLabel Answer;  
 private JButton backMenuButton;  
  
 public Task2(String title) throws HeadlessException {  
 super(title);  
 setContentPane(mainPanel);  
 setSize(500, 350);  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*); //выход с программы  
 setVisible(true);  
  
 getAnswerButton.addActionListener(e -> {  
 String text = textField1.getText();  
 Sentence sentence = new Sentence(text);  
 Answer.setText(sentence.isNumeric() ? "Да" : "Нет");  
 });  
 backMenuButton.addActionListener(e -> {  
 MainMenu menu = new MainMenu("Главное меню");  
 dispose();  
 });  
 }  
}

Класс Task3:

import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class Task3 extends JFrame {  
 private JButton buildMatrixButton;  
 private JTextField textField1;  
 private JTable table1;  
 private JTable table2;  
 private JTable table3;  
 private JPanel mainPanel;  
 private JButton backMenuButton;  
  
 public Task3(String title) throws HeadlessException {  
 super(title);  
 setContentPane(mainPanel);  
 setSize(500, 500);  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*); //выход с программы  
 setVisible(true);  
  
 buildMatrixButton.addActionListener(e -> {  
 String text = textField1.getText();  
 if (text.matches("^[0-9]+$")){  
 int i = Integer.*parseInt*(text);  
 if (i!=0){  
 Matrix matrix1 = new Matrix(i);  
 Matrix matrix2 = new Matrix(i);  
 int[][] sumMatrix = matrix1.sumMatrix(matrix2.matrix);  
 String[][] matrixString1 = new String[i][i];  
 String[][] matrixString2 = new String[i][i];  
 String[][] matrixString3 = new String[i][i];  
 for (int j = 0; j < i; j++) {  
 for (int k = 0; k < i; k++) {  
 matrixString1[j][k] = Integer.*toString*(matrix1.matrix[j][k]);  
 matrixString2[j][k] = Integer.*toString*(matrix2.matrix[j][k]);  
 matrixString3[j][k] = Integer.*toString*(sumMatrix[j][k]);  
 }  
 }  
 table1.setModel(new DefaultTableModel(matrixString1, matrixString1[0]));  
 table2.setModel(new DefaultTableModel(matrixString2, matrixString2[0]));  
 table3.setModel(new DefaultTableModel(matrixString3, matrixString3[0]));  
 } else new checkNumber();  
 } else new checkNumber();  
 });  
 backMenuButton.addActionListener(e -> {  
 MainMenu menu = new MainMenu("Главное меню");  
 dispose();  
 });  
 }  
}

Класс Matrix:

import java.util.Scanner;  
public class Matrix {  
 int size; //размер матрицы  
 int[][] matrix; //переменная matrix  
 public Matrix(int size) { //конструктор класса  
 this.size = size;  
 this.matrix = getRandMatrix(size); //получение рандомной матрицы с указанным размером  
 }  
 public int[][] getRandMatrix(int size) { //функция - получения рандомной матрицы  
 int[][] matrix = new int[size][size];  
 for (int i = 0; i < size; i++) {  
 for (int j = 0; j < size; j++) {  
 double k = Math.*random*() \* 10 + 1;  
 matrix[i][j] = (int) k;//изменение типов данных  
 }  
 } return matrix;  
 }  
 public int[][] sumMatrix(int[][] matrixB) { //функция - сумма матриц - для 3 задания  
 int[][] sumMatrix = new int[size][size];  
 System.*out*.println("\nМатрица А:\n");  
 outputMatrix(this.matrix); //вызов функции - вывод матрицы А  
 System.*out*.println("\nМатрица В:\n");  
 outputMatrix(matrixB); //вызов функции - вывод матрицы В  
 System.*out*.print("\nСумма матриц:\n");  
 for (int i = 0; i < size; i++) {  
 for (int j = 0; j < size; j++) {  
 sumMatrix[i][j] = matrix[i][j] + matrixB[i][j];  
 System.*out*.print(sumMatrix[i][j] + " ");  
 } System.*out*.print("\n");  
 }  
 return sumMatrix;  
 }  
 public int sumMainDiagonal() { //функция - сумма диагоналей матрицы - для 1 задания  
 int sumMain = 0;  
 System.*out*.println("\nМатрица:\n");  
 outputMatrix(this.matrix); //вызов функции - вывод матрицы  
 System.*out*.print("\nСумма элементов главной диагонали: "); //сумма главной диагонали  
 for (int i = 0; i < size; i++) {  
 for (int j = 0; j < size; j++) {  
 if (i == j) {  
 sumMain = sumMain + this.matrix[i][j];  
 }  
 }  
 }  
 return sumMain; //вывод суммы главной диагонали  
 }  
 public int sumSideDiagonal(){  
 int sumSide = 0;  
 System.*out*.print("\nСумма элементов побочной диагонали: "); //сумма побочной диагонали  
 for (int i = 0; i < size; i++) {  
 for (int j = size - 1; j >= 0; j--) {  
 if (i + j == size - 1) {  
 sumSide = sumSide + matrix[i][j];  
 }  
 }  
 }  
 return sumSide; //вывод суммы побочной диагонали  
 }  
 private void outputMatrix(int[][] matrix) { //функция вывода матрицы на экран  
 for (int i = 0; i < size; i++) {  
 for (int j = 0; j < size; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 } System.*out*.print("\n");  
 }  
 }  
 public static int inputNumber(int leftGranitcha, int rightGranitcha) { //функция ввода размерности матрицы  
 String input = "";  
 int x = 0;  
 boolean isCorrect = true;  
 Scanner console = new Scanner(System.*in*);  
 do {  
 try {  
 System.*out*.print("Введите число от " + leftGranitcha + " до " + rightGranitcha + ": ");  
 input = console.nextLine();  
 x = Integer.*parseInt*(input);  
 if ((x>=leftGranitcha) && (x <= rightGranitcha)){  
 break;  
 } else { System.*out*.print("\nНекорректный ввод!\n");}  
 }catch(NumberFormatException e) {  
 isCorrect = false;  
 System.*out*.print("\nОшибка! Введите число!\n");  
 }  
 } while (true);  
 return x;  
 }  
}

Класс Sentence:

import java.util.Scanner;  
public class Sentence {  
 String sentence; //переменная типа строки класса Sentence  
 public Sentence(String text){ //начало конструктора класса Sentence на вход строка-текст  
 this.sentence = text; //присвоение переменной string sentence класса - значение text  
 }  
 Scanner console = new Scanner(System.*in*);  
 public boolean isNumeric() { // функция правда-ложь является ли числом  
 try {  
 Double.*parseDouble*(sentence); // попытка перевода строки в число  
 return true; // если получилось - правда  
 } catch (NumberFormatException e) {  
 return false; // не получилось  
 }  
 }  
}

1. **Список использованных источников**
2. Шилдт, Герберт Java 8. Руководство для начинающих / Герберт Шилдт. - М.: Вильямс, 2015. - 720 c.
3. Берд, Барри Java для чайников / Барри Берд. - М.: Диалектика / Вильямс, 2013. - 521 c.
4. Савитч, Уолтер Язык Java. Курс программирования / Уолтер Савитч. - М.: Вильямс, 2015. - 928 c.