Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**КУРСОВОЙ ПРОЕКТ**

**Дисциплина:** Алгоритмы и структуры данных

**Тема:** Разработка GUI приложения для алгоритма нахождения всевозможных путей из верхнего левого в нижний правый элемент матрицы.

Выполнила

студентка гр. №3530903/80003 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зайцева Е.А.

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Глухих М.И.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург   
2019

Оглавление

[Техническое задание 3](#_Toc27354573)

[Метод решения 4](#_Toc27354574)

[Листинг 5](#_Toc27354575)

[**Main.java** 5](#_Toc27354576)

[**Controller.java** 7](#_Toc27354577)

[**Logic.java** 18](#_Toc27354578)

[**Sample.fxml** 21](#_Toc27354579)

[Тестирование 22](#_Toc27354580)

[Скриншоты работы программы 24](#_Toc27354581)

# Техническое задание

Поле NxM заполняется целыми положительными числами. Целое число в каждой клетке указывает какой длины должен быть шаг из нее. Все шаги могут быть только вправо или вниз. Найти все возможные пути из левого верхнего угла в правый нижний.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 5 | 1 |
| 1 | 2 | 4 | 3 | 2 |
| 3 | 5 | 1 | 1 | 3 |
| 3 | 4 | 2 | 1 | 0 |

GitHub репозиторий проекта: https://github.com/LizaZaytseva/matrix

# Метод решения

В проекте использована концепция MVC (Model-View-Controller) для структурирования кода и отделения логики от визуализации.

Main – осуществляет запуск приложения.

Logic - осуществляет бизнес-логику программы. В нем присутствует только метод result(), представляющий из себя рекурсивный алгоритм поиска всевозможных решений.

Sample.fxml – графическое представление главного окна, меню.

Controller – осуществляет обработку событий (выбор и чтение входного файла, выбор раздела HELP => About, завершение работы программы).

Порядок работы программы:

При запуске программы в меню главного окна выбирается входной файл (File => Choose file).

Проверяются строки входного файла на верность формата (с помощью регулярных выражений) и соответствие заданного и реального размеров матрицы.

Если ошибок нет, записываются размеры матрицы (str – количество строк, stl – количество столбцов) и её элементы (в ArrayList table), если обнаруживается неверный формат входных данных, выводится сообщение об ошибке.

После идет преобразование списка table в двумерный массив map[][] и запускается метод result() класса Logic, реализующий поиск всевозможных путей.

Метод result() принимает на входе массив элементов map, координаты элемента, из которого будет шаг в данный ход, размеры матрицы str и stl, и строку m, в которой записываются порядковые номера элементов, через которые проходит данный путь. Поиск всевозможных решений представляет собой рекурсивный алгоритм(функция result()), суть которого заключается в совершении шагов вправо и вниз на заданное в матрице количество клеток, если такие шаги не выходят за пределы матрицы. При шаге, номер элемента, в который был совершен шаг, записывается в строку m. Результатом метода result() является ArrayList resArr из строк со всевозможными путями.

Если пути существуют, то список, содержащий порядковые номера всех элементов матрицы, через которые проходил путь, форматируется и выводится в окне, если же путей нет, выводится сообщение об этом.

# Листинг

# **Main.java**

package sample;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.control.Menu;

import javafx.scene.control.MenuBar;

import javafx.scene.control.MenuItem;

import javafx.scene.input.KeyCombination;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.StackPane;

import javafx.stage.Modality;

import javafx.stage.Stage;

import java.awt.\*;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import static javafx.fxml.FXMLLoader.load;

public class Main extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));

primaryStage.setTitle("MATRIX");

Scene scene = (new Scene(root, 1100, 700));

primaryStage.setScene(scene);

primaryStage.setResizable(false);

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

# **Controller.java**

package sample;

mport javafx.application.Platform;

import javafx.fxml.FXML;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.control.Menu;

import javafx.scene.control.MenuBar;

import javafx.scene.control.MenuItem;

import javafx.scene.layout.\*;

import javafx.scene.text.Font;

import javafx.stage.FileChooser;

import javafx.stage.Modality;

import javafx.stage.Stage;

import java.io.\*;

import java.net.URL;

import java.util.ArrayList;

import java.util.ResourceBundle;

public class Controller {

@FXML

private ResourceBundle resources;

@FXML

private MenuBar menuBar;

@FXML

private URL location;

@FXML

public Menu HelpMenu;

@FXML

public Menu FileMenu;

@FXML

private MenuItem About;

@FXML

private MenuItem Close;

@FXML

public MenuItem ChooseFile;

@FXML

public AnchorPane AnchorPane;

@FXML

public GridPane gridPane = null;

Label res;

ArrayList<String> resArr = new ArrayList<>();

Logic x = new Logic();

File file;

@FXML

public void initialize() throws IOException {

Close.setOnAction(event -> Platform.exit()); // Закрытие программы из меню

//Правила работы программы

About.setOnAction(event -> {

try {

Label secondLabel = new Label("Правила пользования программой!\n"+

"Поле M\*N заполняется целыми положительными числами, целое\n" +

"число в каждой клетке указывает, какой длины должен быть шаг из нее.\n" +

"Все шаги могут быть вправо или вниз. Программа находит ВСЕ возможные\n" +

"пути из левого верхнего угла в правый нижный\n" +

"Примечание:\n" +

"Для работы программы необходимо выбрать исходный файл, в котором\n" +

"в первой строке должен быть указан размер матрицы, например, 'm\*n',\n" +

"а потом сама матрица. Числа в матрице должны быть записаны\n" +

"через пробел.");

secondLabel.setFont(new Font("courier new", 18));

StackPane secondaryLayout = new StackPane();

secondaryLayout.getChildren().add(secondLabel);

Scene secondScene = new Scene(secondaryLayout, 800, 250);

//Создание главного окна

Stage newWindow = new Stage();

newWindow.setTitle("Rules");

newWindow.setScene(secondScene);

newWindow.initModality(Modality.WINDOW\_MODAL);

newWindow.show();

} catch (Exception e) {

e.printStackTrace();

}

});

}

//создание листа, хранящего все значения из входного файла

ArrayList<String> table = new ArrayList<>();

public void openAndReadFile(javafx.event.ActionEvent actionEvent) {

// диалоговое окно для выбора исходного файла

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Open Document");

// проверка исходного файла на верность типа

FileChooser.ExtensionFilter extFilter =

new FileChooser.ExtensionFilter("TXT files (\*.txt)", "\*.txt");

fileChooser.getExtensionFilters().add(extFilter);

Stage newWindow = new Stage();

file = fileChooser.showOpenDialog(newWindow);

try {

//чтение входного файла

FileReader fr = new FileReader(file);

BufferedReader reader = new BufferedReader(fr);

String line = reader.readLine();

int k = 0;// счетчик строк

int str = 0;// количество строк

int stl = 0;// количество столбцов

//очищаем листы от старых данных

table.clear();

resArr.clear();

//разбор входного файла на матрицу и параметры матрицы, а также проверка на верность входных данных

while (line != null) {

if ((k == 0) && (line.matches("\\d+\\\*\\d+"))) {

String [] size = line.split("\\" +

"\*");

//количество строк в таблице

str = Integer.valueOf(size[0]);

//количество столбцов в таблице

stl = Integer.valueOf(size[1]);

}

else if ((k != str && line.matches("[\\d+\\s]\*") && ((line.split("\\s")).length == stl)) ||

(k == str && line.matches("[\\d+\\s]\*0$") && ((line.split("\\s")).length == stl))) {

//добавление элементов матрицы в лист

for (String st : line.split("\\s")){

table.add(st);

}

}

//окно ошибки, в случае неверных входных данных

else {

Label secondLabel = new Label(" Ошибка входных данных!\n" +

"Проверьте входной файл на наличие ошибок.\n" +

"Про необходимый формат вы можете прочитать\n" +

"в разделе HELP.");

secondLabel.setFont(new Font("courier new", 18));

StackPane secondaryLayout = new StackPane();

secondaryLayout.getChildren().add(secondLabel);

Scene secondScene = new Scene(secondaryLayout, 500, 150);

Stage exWindow = new Stage();

exWindow.setTitle("Input file is incorrect");

exWindow.setScene(secondScene);

newWindow.initModality(Modality.WINDOW\_MODAL);

exWindow.show();

throw new Exception();

}

line = reader.readLine();

k++;

};

if (k - 1 != str) {

Label secondLabel = new Label(" Ошибка входных данных!\n" +

"Проверьте входной файл на наличие ошибок.\n" +

"Про необходимый формат вы можете прочитать\n" +

"в разделе HELP.");

secondLabel.setFont(new Font("courier new", 18));

StackPane secondaryLayout = new StackPane();

secondaryLayout.getChildren().add(secondLabel);

Scene secondScene = new Scene(secondaryLayout, 500, 150);

Stage exWindow = new Stage();

exWindow.setTitle("Input file is incorrect");

exWindow.setScene(secondScene);

newWindow.initModality(Modality.WINDOW\_MODAL);

exWindow.show();

throw new Exception();

}

try {

//создание двумерного массива, хранещего значения матрицы

int[][] map = new int[str][stl];

int l = 0;

for (int i = 0; i < str; i++) {

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

map[i][j] = Integer.valueOf(table.get(l));

l++;

}

}

String m = "";

//использование класса, осуществляющего логику программы

resArr = x.result(map, 0, 0, str, stl, m);

String finalResult = "";

int r = 1;

//форматируем полученный результат

for (String s : resArr) {

finalResult += r + ") "+ s + " \n";

r++;

}

//обновление окна после работы программы

if ( gridPane == null) gridPane = new GridPane();

else {

gridPane.getChildren().clear();

gridPane = new GridPane();

AnchorPane.getChildren().remove(res);

}

//отрисовка таблицы

ColumnConstraints cc = new ColumnConstraints();

cc.setFillWidth(true);

cc.setHgrow(Priority.ALWAYS);

for (int i = 0; i < stl; i++) {

gridPane.getColumnConstraints().add(cc);

}

RowConstraints rc = new RowConstraints();

rc.setFillHeight(true);

rc.setVgrow(Priority.ALWAYS);

for (int i = 0; i < str; i++) {

gridPane.getRowConstraints().add(rc);

}

gridPane.setHgap(0);

gridPane.setVgap(0);

gridPane.setGridLinesVisible(true);

//заполнение таблицы

int y = 0;

int n = 1;

int[][] arr = new int[str][stl];

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

for (int j = 0; j < arr[i].length; j++) {

Label lb = new Label(" " + table.get(y) + " " + "(" + n + ") ");

lb.setFont(Font.font(25));

gridPane.add(lb, j , i);

y++;

n++;

}

}

//характеристики окна

gridPane.autosize();

gridPane.setLayoutX(50);

gridPane.setLayoutY(90);

//результат работы программы

if (!finalResult.isEmpty()){

res = new Label("Всевозможные пути:\n" + finalResult);

} else {

res = new Label("Нет решений! Попробуйте ввести другую матрицу!");

}

Label mLabel = new Label("Заданная матрица:");

mLabel.setFont(new Font("courier new", 22));

//выбор координат расположения матрицы

mLabel.setLayoutX(50);

mLabel.setLayoutY(50);

//расположение результата работы программы, формула расчитывается след. способом:

// 50 - отступ от края, stl - количество столбцов, 80 - расстояние, выделенное на каждый столбец, 70 - отступ от края матрицы

res.setLayoutX(50 + stl \* 80 + 70);

res.setLayoutY(50);

res.setFont(new Font("courier new", 22));

AnchorPane.getChildren().add(mLabel);

AnchorPane.getChildren().add(res);

AnchorPane.getChildren().add(gridPane);

} catch (Exception e) {

e.printStackTrace();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

# **Logic.java**

package sample;

import java.util.ArrayList;

public class Logic {

private ArrayList<String> finish = new ArrayList<>();

public ArrayList<String> result(int[][] arr, int startA, int startB, int str, int stb, String res){

//значение элемента матрицы = > длина шага

int x = arr[startA][startB];

//вычисление порядкого номера элемента в листе

int num = stb \* (startA) + startB + 1;

int f = str \* stb; //клетка окончания программы

String st = String.valueOf(num);

//проверяем, можно ли сделать ход вниз

if (startA + x < str) {

//записываем промежуточный результат

res+= st + " ";

//если возможен ход в итоговую клетку, то записываем результат

if (startA + x == str - 1 && startB == stb - 1) {

res += String.valueOf(f);

}

//если ход в итоговую клетку невозможен, запускаем наш метод в клетке, в которую осуществлен переход

else result(arr, startA + x, startB, str, stb, res);

}

//проверяем, можно ли сделать ход вправо

if (startB + x < stb) {

//записываем промежуточный результат

res+= st + " ";

//если возможен ход в итоговую клетку, то записываем результат

if (startB + x == stb - 1 && startA == str - 1) {

res += String.valueOf(f);

}

//если ход в итоговую клетку невозможен, запускаем наш метод в клетке, в которую осуществлен переход

else result(arr, startA, startB + x, str, stb, res);

}

int s = 0;

if (!res.isEmpty()) {

int splL = res.split(" ").length;

s = Integer.valueOf(res.split(" ")[splL - 1]);

}

//проверяем, является ли последний ход, ходом в финальную клетку

if (f == s) {

String[] r = res.split(" ");

//проверяем на повтор промежуточные точки, повторяющиеся удаляем => последствие логики поиска ходов

for (int i = 0; i < r.length - 1; i++) {

if (r[i].equals(r[i + 1])) {

res = res.replaceFirst(r[i] + " ", "");

}

}

finish.add(res);

}

return finish;

}

}

# **Sample.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane fx:id="AnchorPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="610.0" prefWidth="971.0" style="-fx-background-color: ffe4c4;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller">

<children>

<MenuBar layoutY="-2.0" prefHeight="32.0" prefWidth="1200.0" style="-fx-background-color: #FFFFF0;">

<menus>

<Menu fx:id="FileMenu" mnemonicParsing="false" text="File">

<items>

<MenuItem fx:id = "ChooseFile" mnemonicParsing="false" onAction="#openAndReadFile" text="Choose file" />

<MenuItem fx:id = "Close" mnemonicParsing="false" text="Close" />

</items>

</Menu>

<Menu fx:id="HelpMenu" mnemonicParsing="false" text="Help">

<items>

<MenuItem fx:id = "About" mnemonicParsing="false" text="About" />

</items>

</Menu>

</menus>

</MenuBar>

</children>

</AnchorPane>

# Тестирование

Класс TestsMatrix тестирует исключительно класс Logic и его единственный метод result().

package tests;

import org.junit.Test;

import sample.Logic;

import static org.junit.jupiter.api.Assertions.\*;

public class TestsMatrix {

Logic logic = new Logic();

Logic logic2 = new Logic();

Logic logic3 = new Logic();

String res = "";

String res2 = "";

//матрица с 1 возможным путём

int[][] arr = new int[][]{

{1, 2, 2},

{2, 1, 3}

};

//матрица без возможных путей

int[][] arr2 = new int[][]{

{10, 12, 13, 14},

{2, 1, 3, 5},

{3, 2, 2, 2},

{2, 6, 3, 2},

{1, 2, 2, 6},

{2, 1, 3, 1}

};

//матрица с 6 возможными путями

int[][] arr3 = new int[][]{

{1, 1, 1},

{1, 1, 1},

{1, 1, 1}

};

@Test

public void LogicTests(){

assertEquals(0, logic2.result(arr2, 0, 0, 6,4, res2).size() );

assertEquals("[1 4 6]", logic.result(arr, 0, 0, 2,3, res).toString());

assertEquals( "[1 4 7 8 9, 1 4 5 8 9, 1 4 5 6 9, 1 2 5 8 9, 1 2 5 6 9, 1 2 3 6 9]", logic3.result(arr3, 0, 0, 3,3, res2).toString());

}

}

# Скриншоты работы программы

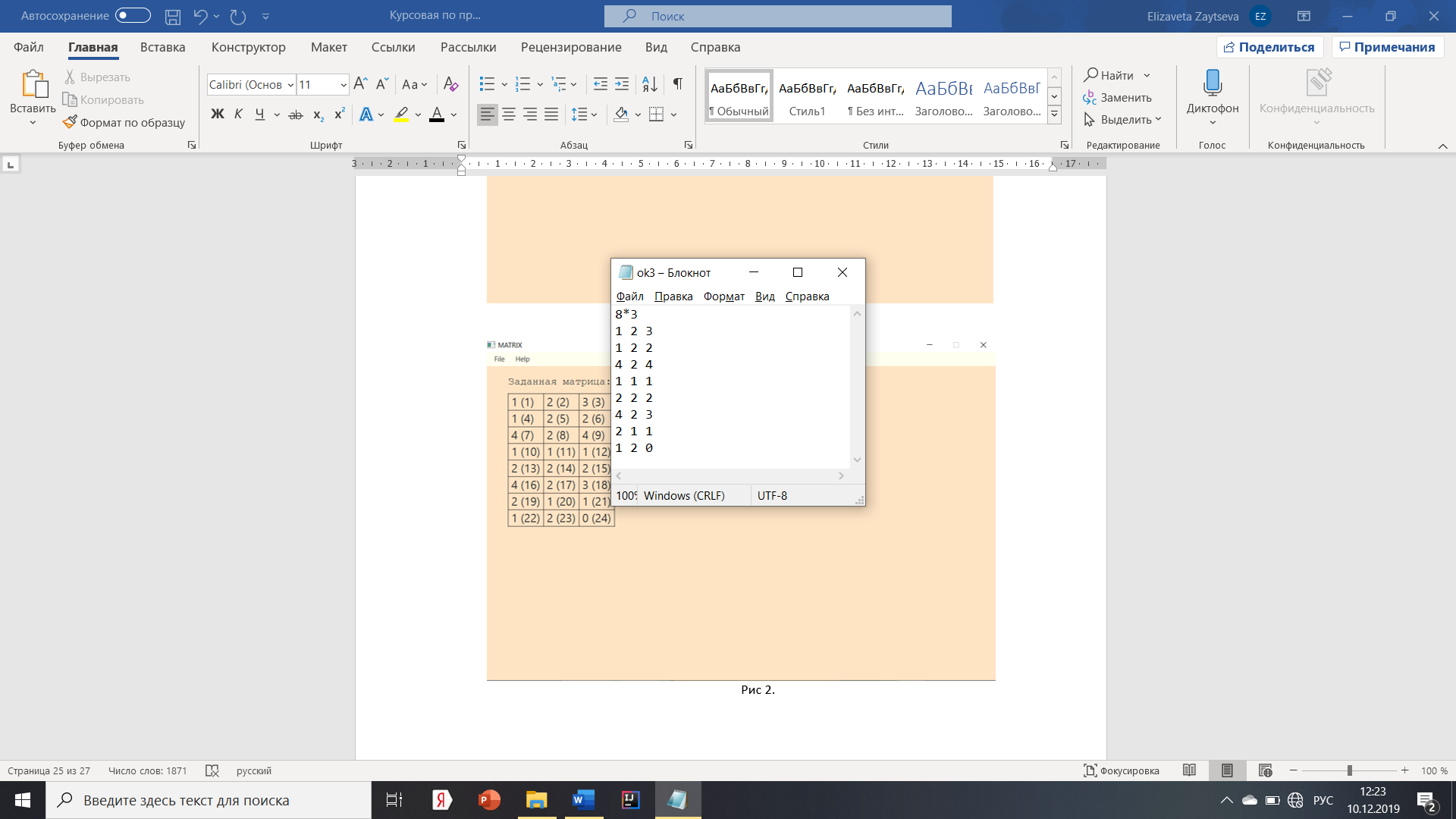
Рис.1 Главное окно и меню программы

Рис.2 Входной файл верного формата

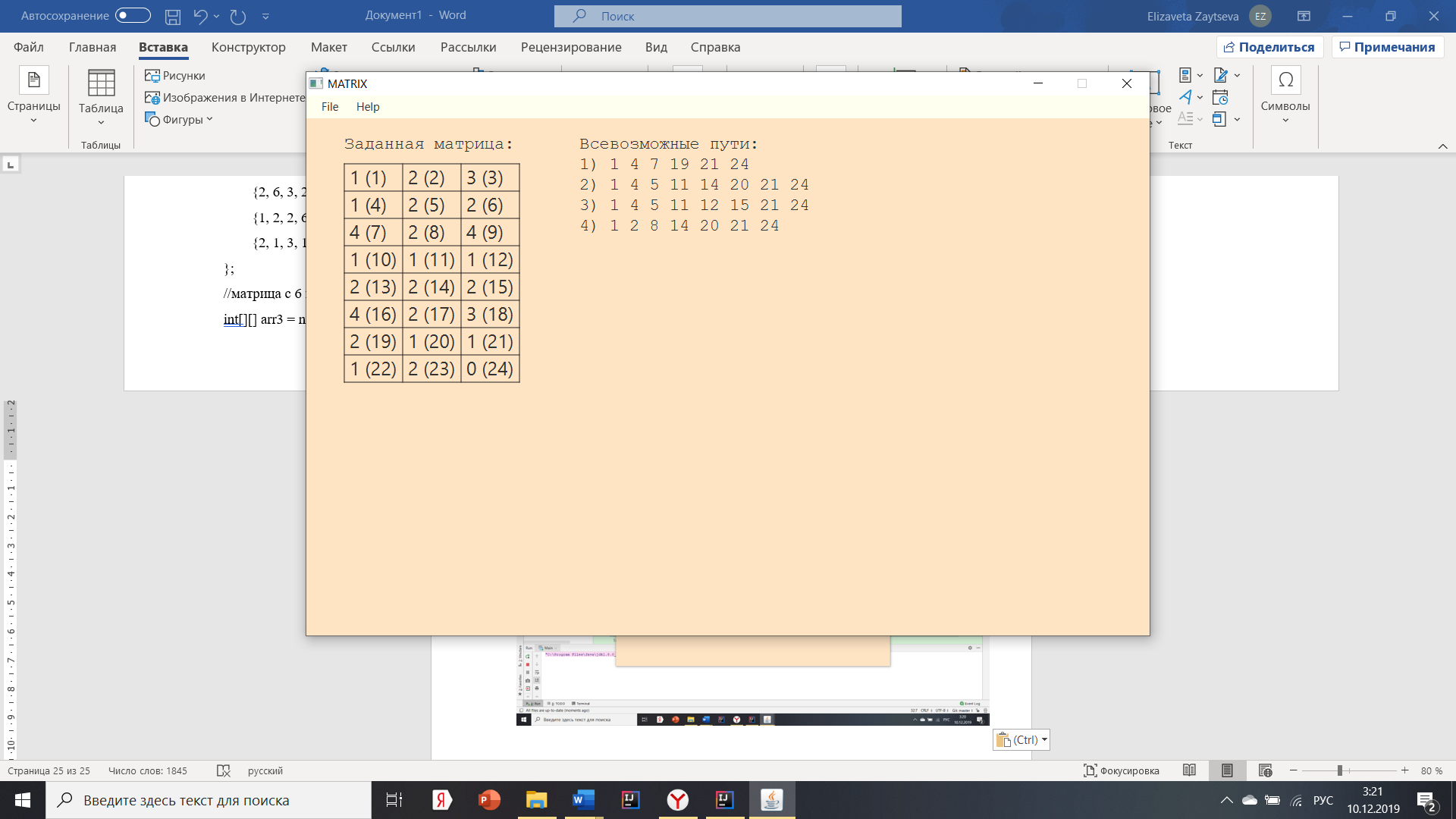
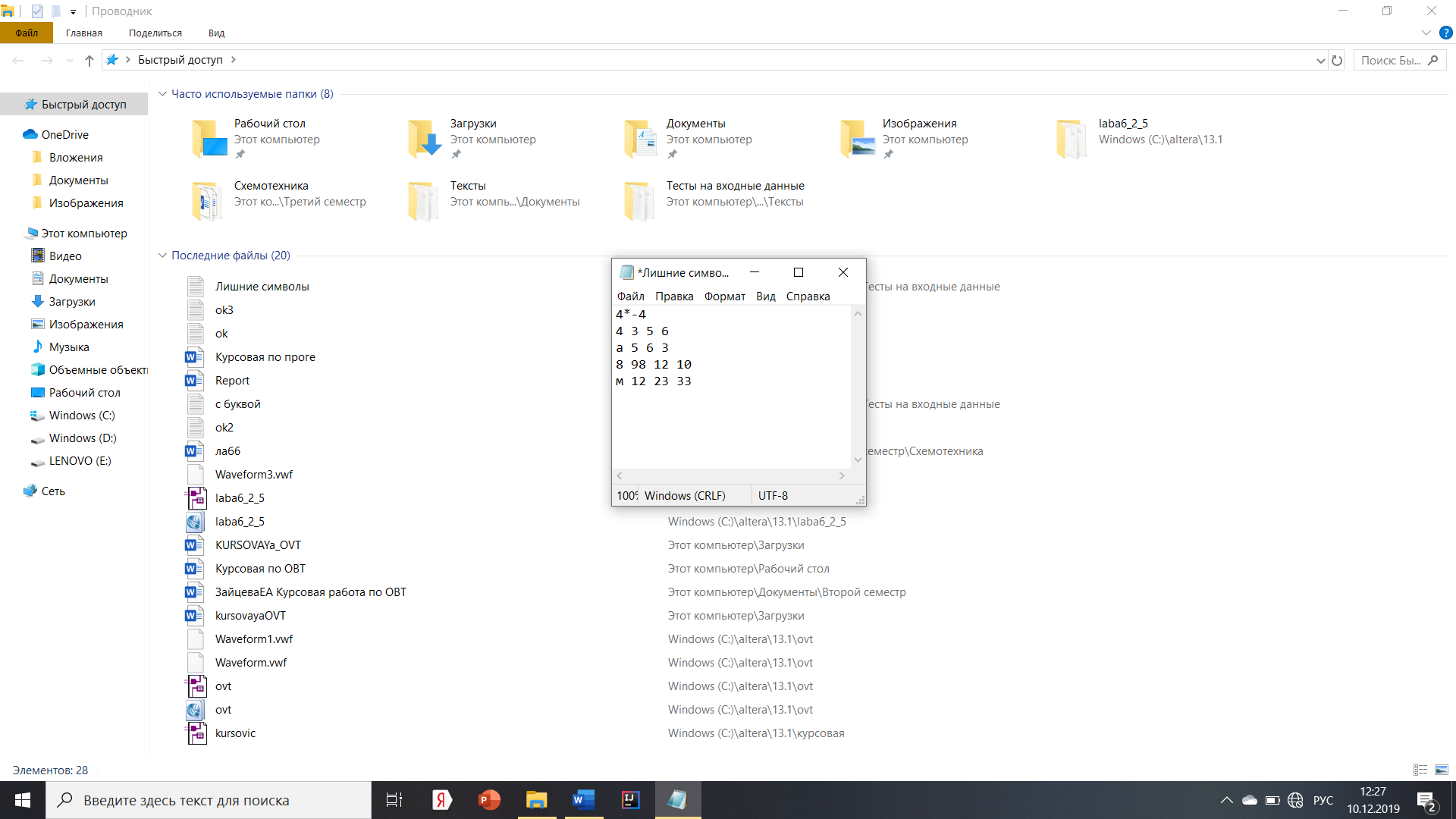
Рис.3 Результаты работы программы

Рис.4 Входной файл неверного формата

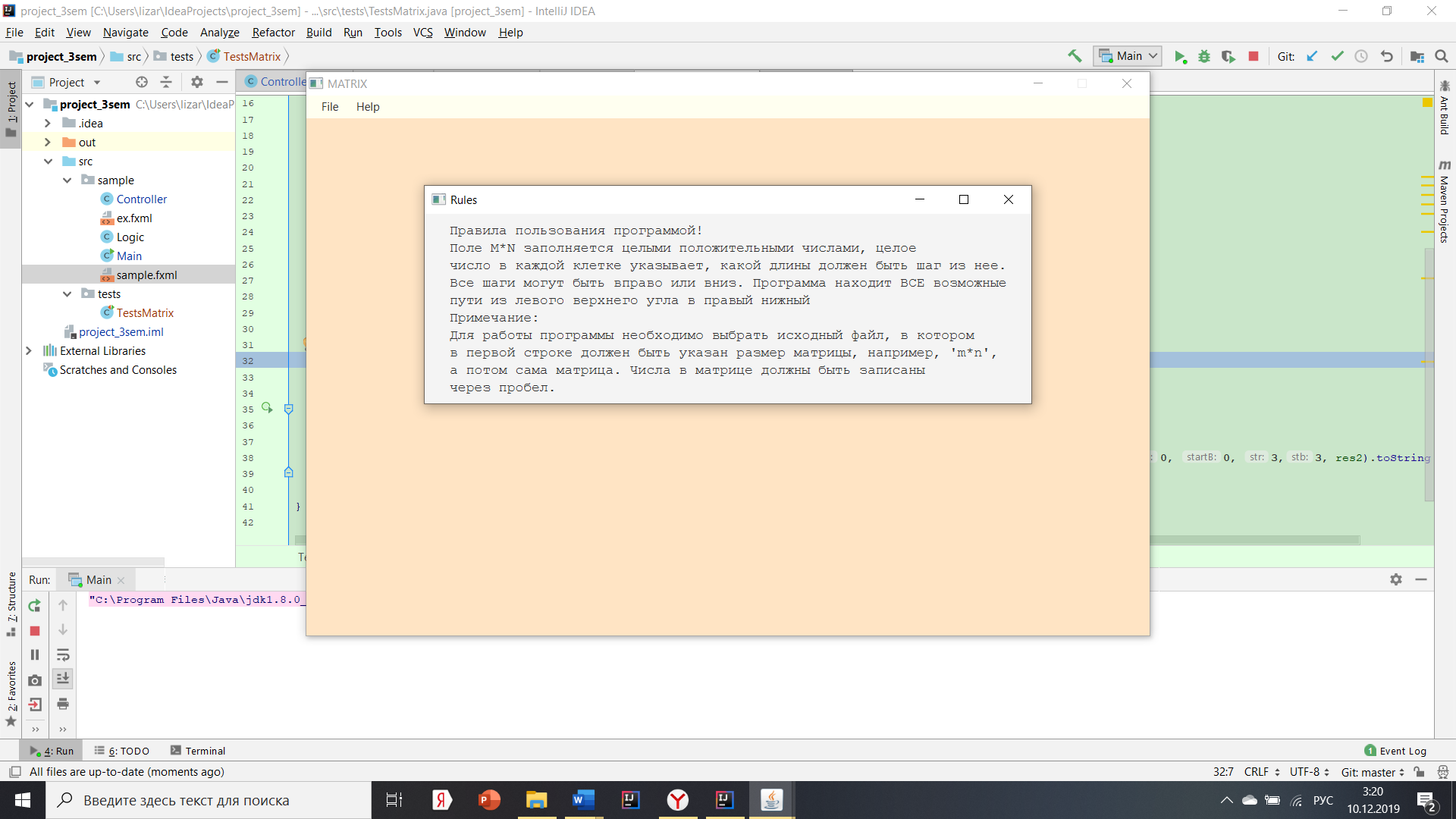
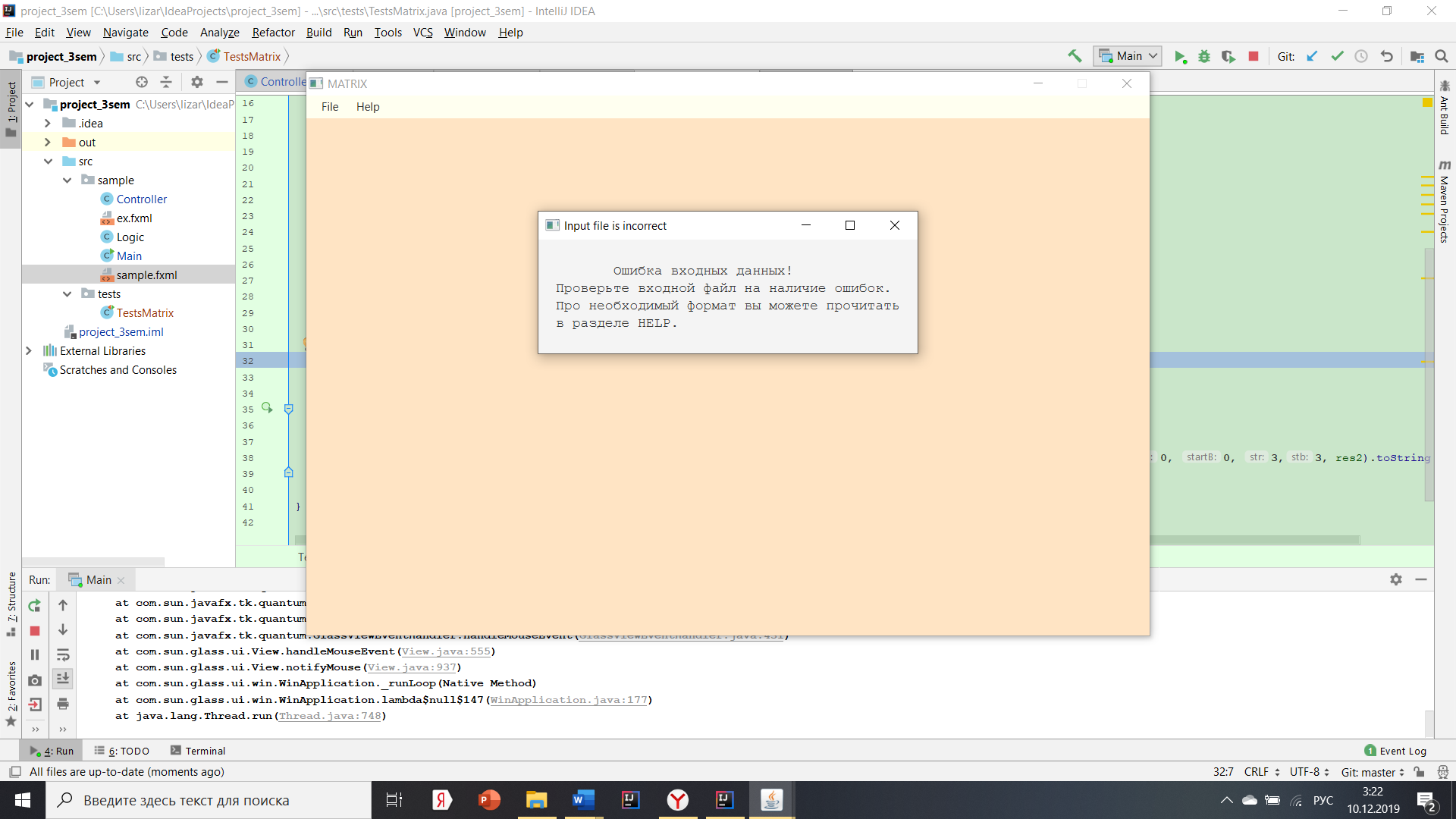
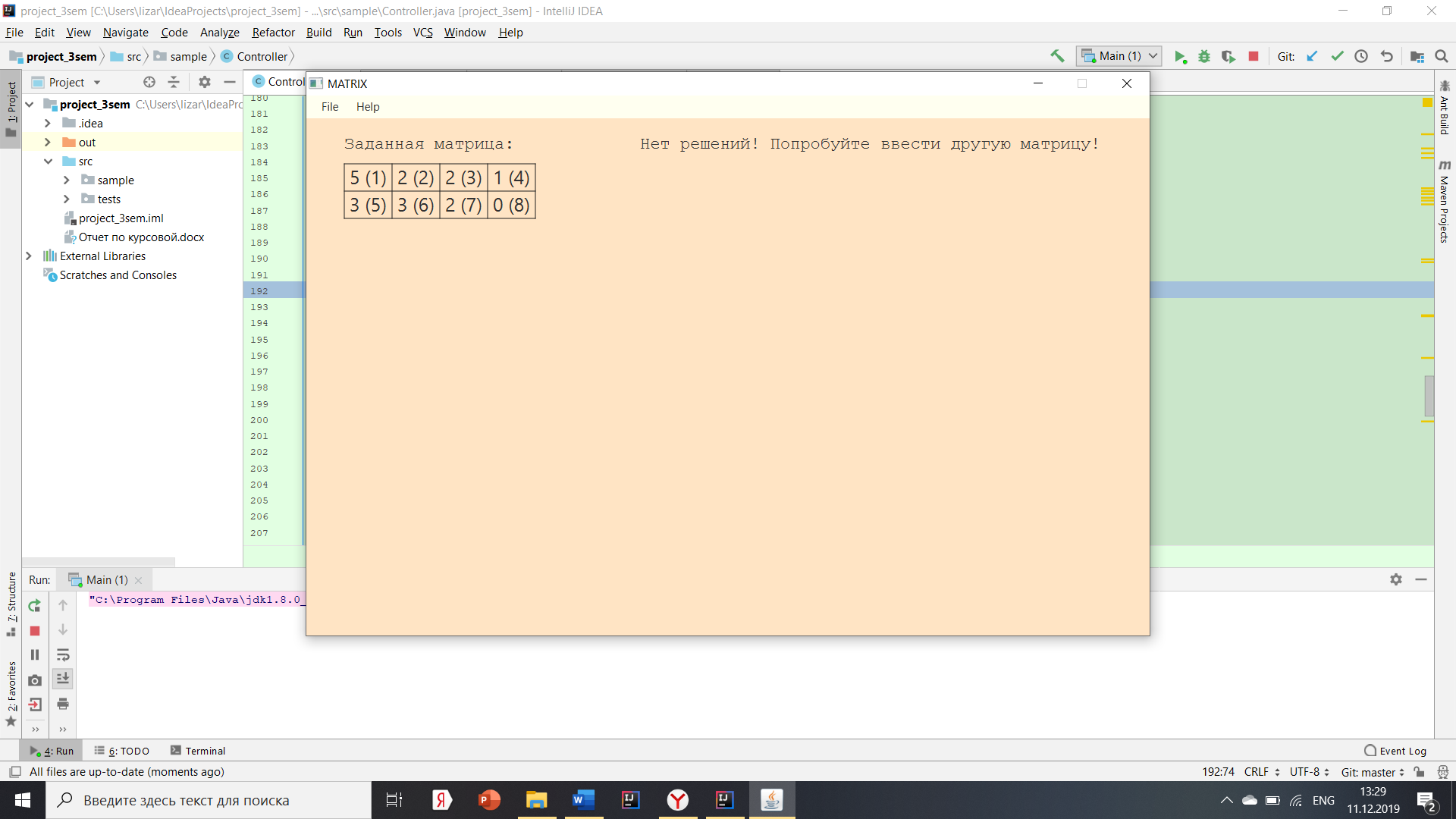
Рис.5 Сообщение о неверном формате входного файла

Рис.6 Окно Rules (при выборе Help => About) с правилами работы программы и информацией о необходимом формате входных данных

Рис.7 Результат работы программы при отсутствии решений.