МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

Тема: Визуализация алгоритма Уоршелла

Студент гр. 7383 _____ Бергалиев М.

Студентка гр. 7383 _____ Тян Е.

Студентка гр. 7383 _____ Прокопенко Н.

Руководитель _____ Размочаева Н.В.

Санкт-Петербург

2019

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Бергалиев М. группы 7383

Студент Тян Е. группы 7383

Студент Прокопенко Н. группы 7383

Тема практики: визуализация алгоритма Уоршелла

Задание на практику:
Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.
Алгоритм: Уоршелла построения транзитивного замыкания.

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета: 12.07.2019

Дата защиты отчета: 12.07.2019

Студент	_____	Бергалиев М.
Студентка	_____	Тян Е.
Студентка	_____	Прокопенко Н.
Руководитель	_____	Размочаева Н.В.

АННОТАЦИЯ

Целью практики являлось создание визуализатора алгоритма. На протяжении практики работа велась в команде по итеративному плану. Каждый член команды имел свои обязанности. Был написан код визуализируемого алгоритма. Был создан удобный пользовательский интерфейс. Было добавлено логирование всех компонент программы. Было проведено тестирование для проверки корректности составляющих программы.

SUMMARY

The aim of the practice was to create an algorithm visualizer. Throughout the practice, work was carried out in a team on an iterative plan. Each member of the team had their own responsibilities. The code of the visualized algorithm was written. A user friendly interface has been created. Logging of all program components was added. Testing was conducted to verify the correctness of the program components.

СОДЕРЖАНИЕ

	Введение	5
1.	Требования к программе	6
1.1.	Исходные требования к программе	6
1.2.	Уточнение требований после сдачи прототипа	7
1.3.	Уточнение требований после сдачи 1-ой версии	7
2.	План разработки и распределение ролей в бригаде	8
2.1.	План разработки	8
2.2.	Распределение ролей в бригаде	8
3.	Особенности реализации	9
3.1.	Использованные структуры данных	9
3.2.	Основные методы	9
3.3.	Работа графического интерфейса	9
4.	Тестирование	13
4.1	Тестирование графического интерфейса	13
4.2	Тестирование кода алгоритма	13
	Заключение	15
	Список использованных источников	16
	Приложение А. Исходный код – только в электронном виде	17

ВВЕДЕНИЕ

Целью практики является освоение навыков разработки программ в команде.

Задачей практики является создание визуализатора алгоритма Уоршелла построения транзитивного замыкания на языке Java.

Алгоритм Уоршелла оперирует матрицей смежности графа. На каждом шаге алгоритм умножает матрицу смежности на себя и складывает ее с матрицей на предыдущем шаге. В полученной матрице все ненулевые значения заменяются на единицы. Алгоритм продолжает работу, пока матрицы на текущем и предыдущем шагах не окажутся равными. [1]

Алгоритм Уоршелла не является самым эффективным, поэтому на практике используется редко. Построение транзитивного замыкания может использоваться для нахождения отношений между объектами на основе данного. Например, на отношении «является родителем» можно построить отношение «является предком», на отношении «непосредственно используется в» «используется в». [2]

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные требования к программе

1.1.1. Требования к вводу исходных данных

Ввод исходных данных должен происходить в отдельном окне.

На вход подаются ребра ориентированного графа в следующем формате:

<вершина_из_которой_исходит_ребро> <вершина_в_которую_входит_ребро>

Именем вершины могут быть строчные буквы латинского алфавита(от 'a' до 'z'). Ограничение количества символов связано с тем, что при большом количестве вершин граф будет непригодным для восприятия пользователем.

1.1.2. Требования к визуализации

На каждом шаге алгоритма в окне отображается текущее графическое представление и матрица смежности графа.

Пользовательский интерфейс содержит несколько кнопок для управления состоянием программы. А именно следующие:

- Ввод исходных данных: при нажатии на данную кнопку вызывается отдельное окно, в котором пользователь может ввести входные данные в формате, описанном выше.
- Считать из файла: при нажатии вызывается окно, где можно выбрать любой текстовый файл, откуда будет производиться считывание.
- Следующий шаг: при нажатии выполняет следующий шаг алгоритма и отображает результат в окне.
- Предыдущий шаг: при нажатии возвращает предыдущее состояние выполнения алгоритма.
- Окончательный результат: при нажатии программа отображает состояние графа после окончания алгоритма.
- К начальному состоянию: при нажатии возвращает состояние графа к исходному (к состоянию графа при вводе входным данных).

1.2. Уточнение требований после прототипа

Программа должна подсвечивать изменившиеся ячейки в матрице смежности графа.

1.3. Уточнение требований после 1-ой версии

Пользователю должен быть доступен ввод исходных данных из произвольного файла.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

План разработки представлен в табл. 1.

Таблица 1 — План разработки

Шаги разработки	Результаты шага разработки	Сроки реализации
Разработка спецификации	Определены требования к программе	3 июля
Прототип	Демонстрация окна пользовательского интерфейса программы без функционала.	4 июля
Итерация 1	Написаны код алгоритма и тестирующая программа к нему. Кнопка "Ввод исходных данных" работает так, как описано выше с обработкой всех некорректных ситуаций.	6 июля
Итерация 2(Версия программы №1)	Усовершенствован алгоритм до пошагового исполнения и написана тестирующая программа к нему. Состояние графа выводится в главное или отдельное окно, кнопки "Следующий шаг", "Предыдущий шаг", "Окончательный результат", "К начальному состоянию" модифицируют его.	8 июля
Итерация 3(Версия программы №2)	Вместо текстового состояния графа выводится графическое. Описаны тестовые случаи для проверки графического отображения. Добавлено логирование.	10 июля
Финальная версия	Исправлены недочеты, написан отчет.	12 июля

2.2. Распределение ролей в бригаде

Бергалиев М. - архитектор, ответственный за назначение задач, написание алгоритма, логирование.

Тян Е. - ответственна за GUI.

Прокопенко Н. - ответственна за написание тестов.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Использованные структуры данных

Для описания графа используется матрица смежности. Поскольку граф неориентированный, в ячейках матрицы хранятся булевы значения. Был написан класс булевой матрицы BoolMatrix.

Класс Graph, описывающий граф, представляется булевой матрицей и набором вершин.

Класс WarshallAlgorithm, хранит в себе граф, а так же его состояния по ходу алгоритма для удобного возвращения к ним.

3.2. Основные методы

В классе WarshallAlgorithm представлены основные методы алгоритма.

Метод transitiveClosure совершает весь алгоритм от текущего состояния до конечного.

Метод stepUp совершает один шаг алгоритма, причем если он уже был совершен ранее, то состояние графа восстанавливается из истории.

Метод stepDown возвращается на предыдущий шаг алгоритма, состояние графа восстанавливается из истории.

Метод toStart возвращает изначальное состояние графа.

Метод toFinalResult доводит состояние графа до конечного.

Для визуализации работы алгоритма использовались методы GraphDraw, setMatrix, createDialog класса MyFrom.

Метод GraphDraw рисует граф, используя данные о текущем состоянии объекта класса WarshallAlgorithm.

Метод setMatrix строит матрицу смежности для заданного объекта класса WarshallAgorithm и подсвечивает изменившиеся ячейки.

Метод createDialog создает окно ввода исходных данных.

3.3. Работа графического интерфейса

UML-диаграмма, иллюстрирующая работу программы представлена на рис. 1.

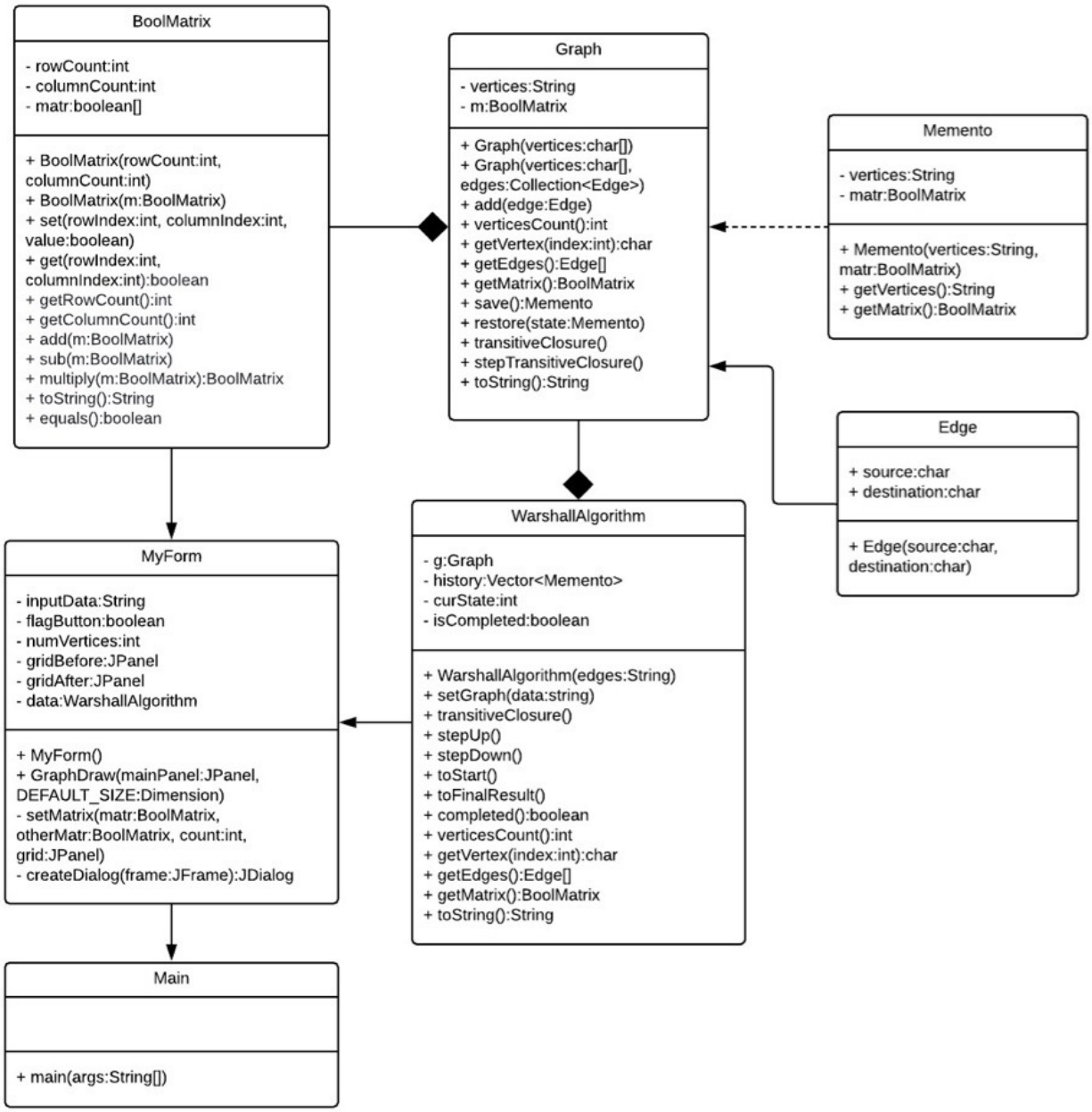

Рисунок 1 — UML-диаграмма

При запуске программы выводится главное окно с двумя кнопками, предлагающими варианты ввода исходных данных: «Считать из файла» или «Ввод исходных данных», представленные на рис. 2.

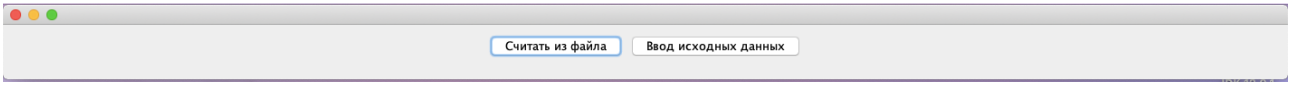

Рисунок 2 — главное окно программы

При нажатии на кнопку «Считать из файла» отображается окно, показанное на рис. 3, где можно выбрать любой текстовый файл.

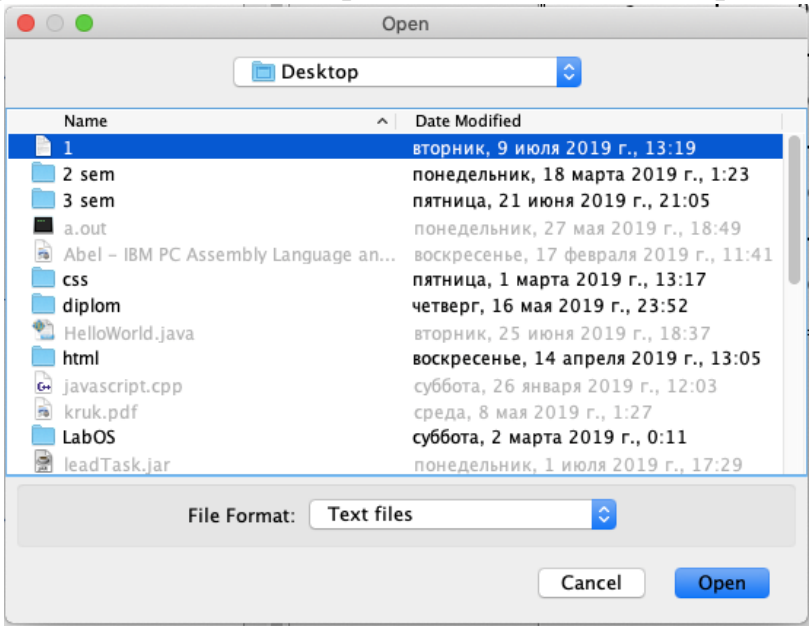

Рисунок 3 — окно выбора файла

При нажатии на кнопку «Ввод исходных данных» отображается окно вода данных, представленное на рис. 4.

При неверности введенных данных выводится сообщение об ошибке, представленное на рис. 5.

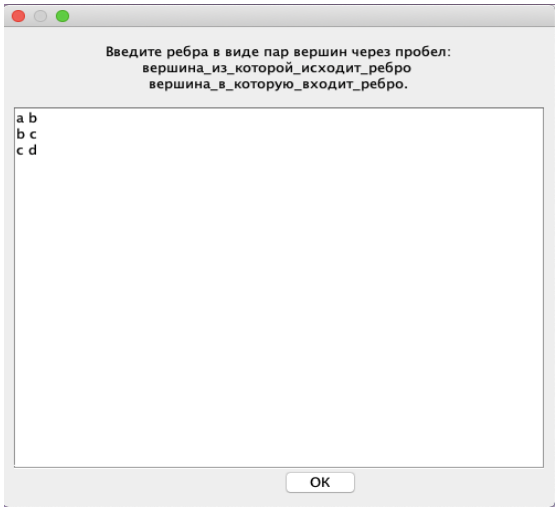

Рисунок 4 — окно ввода данных

Рисунок 5 — сообщение об ошибке

При корректном считывании выводится окно, приведенное на рис. 6, где располагаются кнопки «Следующий шаг», «Предыдущий шаг», «Окончательный результат», «К начальному состоянию». Также в данном окне выводится представление графа после применения шага алгоритма и матричное представление до и после применения шага алгоритма. Функционал представленных кнопок соответствует требованиям.

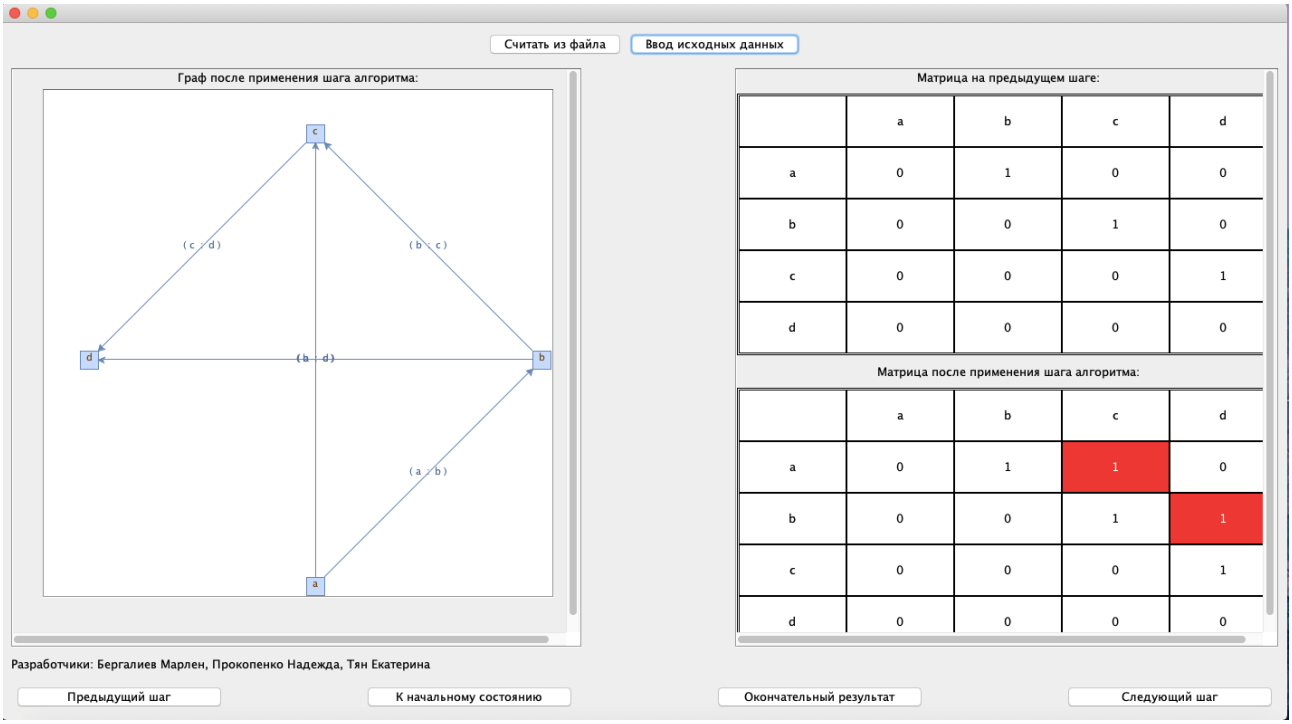

Рисунок 6 — основное окно программы

4. ТЕСТИРОВАНИЕ

4.1. Тестирование графического интерфейса

Запуск и тестирование графического интерфейса программы происходило в среде разработки IntelliJ IDEA. [3] Все кнопки графического интерфейса данной программы работают корректно. В ходе тестирования никаких ошибок выявлено не было.

4.2. Тестирование кода алгоритма

Программа тестировалась в среде разработки IntelliJ IDEA. Было проведено модульное тестирование с помощью библиотеки Junit [4], для проверки корректности работы программы исходя из её спецификации и согласно плану тестирования. Были проведены тесты на работу программы вне диапазона допустимых вершин. Также были проведены тесты алгоритма на граничных значениях и на случайных данных. Были проверены функции перехода к начальному и конечному результату, перехода на следующий и предыдущий шаг. Во всех случаях результаты тестов были положительные.

В тестах реализованы следующие функции:

`testsetGraphData()` – тест, проверяющий обработку ошибки ввода данных и правильность построения матрицы смежности для данного графа.

`testsetGraphData()` – тест, проверяющий правильность работы алгоритма Уоршелла построения транзитивного замыкания.

`test_stepUp()` – тест, проверяющий правильность перехода к следующему шагу алгоритма.

`test_stepDown()` – тест, проверяющий правильность перехода к предыдущему шагу алгоритма.

`test_toFinalResult()` – тест, проверяющий правильность перехода к конечному состоянию графа с любого шага алгоритма.

`test_toStart()` – тест, проверяющий правильность перехода к начальному состоянию графа с любого шага алгоритма.

`test_file()` – тест, проверяющий правильность открытия, считывания данных из файла, создания графа и обработки любой ошибки.

ЗАКЛЮЧЕНИЕ

В ходе практической работы требовалось построить визуализатор алгоритма. В качестве результата практической работы был разработан и написан визуализатор алгоритма Уоршелла построения транзитивного замыкания на языке Java. Были освоены навыки разработки программ в команде. Был написан код реализуемого алгоритма на языке программирования java. Был разработан пользовательский итерфейс для визуального представления работы алгоритма. Было проведено тестирование программы на наличие и отсутствие ошибок работы алгоритма и обработки исключительных ситуаций. Также были написаны тесты для проверки корректности работы графического пользовательского интерфейса. Было добавлено логирование всех компонентов программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Зайцева, О.Н. Математические методы в приложениях. Дискретная математика: учеб. пособие / А.Н. Нуриев, П.В. Малов, Казан. нац. исслед. технол. ун-т, О.Н. Зайцева .— Казань : КНИТУ, 2014 .— 173 с.

2. Транзитивное замыкание // Википедия. URL: https://ru.wikipedia.org/wiki/Транзитивное_замыкание (дата обращения: 11.07.2019).

3. Petar Tahchiev. «JUnit in Action». «Manning Publications», 2010.

4. Andy Hunt, Dave Thomas. «Pragmatic Unit Testing in Java with Junit». «Pragmatic Bookshelf», 2003.

ПРИЛОЖЕНИЕ А
НАЗВАНИЕ ПРИЛОЖЕНИЯ

Main.java:

```
import java.util.logging.*;

import javax.swing.*;

import javax.swing.filechooser.*;

import java.awt.*;

import java.awt.event.*;

import java.io.*;

import com.mxgraph.layout.mxCircleLayout;

import com.mxgraph.swing.mxGraphComponent;

import org.jgrapht.ListenableGraph;

import org.jgrapht.ext.JGraphXAdapter;

import org.jgrapht.graph.DefaultDirectedGraph;

import org.jgrapht.graph.DefaultEdge;

import org.jgrapht.graph.DefaultListenableGraph;


public class Main {

    private static Logger log = Logger.getLogger(Main.class.getName());

    public static void main(String[] args) {

        try {

            LogManager.getLogManager().readConfiguration(

                                                    Main.class.getResource-
AsStream("../logging.properties"));

        } catch (Exception e) {

            System.err.println("Could not setup logger configuration: " +
e.toString());

        }

        try{
```

```
            MyForm win = new MyForm();
        }
        catch(Exception e) {
            log.log(Level.SEVERE, "Error: ", e);
        }
    }
}


class MyForm{
    private static Logger log = Logger.getLogger(MyForm.class.getName());
    private String inputData = "";
    private boolean flagButton = false;
    private int numVertices = 0;
    private JPanel gridBefore = new JPanel();
    private JPanel gridAfter = new JPanel();
    private WarshallAlgorithm data;
    public MyForm(){
        log.fine("Creating main window");
        JFrame frame  = new JFrame();
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        JPanel pan = new JPanel(new BorderLayout(10, 10));
        pan.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        JPanel buttonPanel = new JPanel(new GridBagLayout());
        buttonPanel.setPreferredSize(new Dimension(1400, 30));
        buttonPanel.setMaximumSize(new Dimension(1400, 30));
```

```
JButton fileButton = new JButton("Считать из файла");
GridBagConstraints constraints2 = new GridBagConstraints();
constraints2.gridwidth = 2;
constraints2.gridheight = 1;
buttonPanel.add(fileButton, constraints2);

JButton firstButton = new JButton("Ввод исходных данных");
GridBagConstraints constraints1 = new GridBagConstraints();
constraints1.gridwidth = 2;
constraints1.gridheight = 1;
buttonPanel.add(firstButton, constraints1);
pan.add(buttonPanel, BorderLayout.NORTH);

JPanel lastPanel = new JPanel(new BorderLayout());
lastPanel.setPreferredSize(new Dimension(1400, 60));
lastPanel.setMinimumSize(new Dimension(1400, 60));
lastPanel.setMaximumSize(new Dimension(1400, 60));
JPanel pane = new JPanel(new GridLayout(1, 4, 150, 0));
pane.setPreferredSize(new Dimension(1400, 30));
pane.setMinimumSize(new Dimension(1400, 30));

JButton prevBut = new JButton("Предыдущий шаг");
JButton nextBut = new JButton("Следующий шаг");
JButton finishBut = new JButton("Окончательный результат");
JButton initBut = new JButton("К начальному состоянию");
pane.add(prevBut);
pane.add(initBut);
pane.add(finishBut);
```

```
        pane.add(nextBut);

        lastPanel.add(new JLabel("Разработчики: Бергалиев Марлен, Прокопенко Надежда,
Тян Екатерина"), BorderLayout.NORTH);

        lastPanel.add(pane, BorderLayout.SOUTH);

        pan.add(lastPanel, BorderLayout.SOUTH);


        JPanel panelMatrix = new JPanel();

        panelMatrix.setLayout(new BoxLayout(panelMatrix, BoxLayout.PAGE_AXIS));

        panelMatrix.setPreferredSize(new Dimension(600, 640));

        panelMatrix.setMinimumSize(new Dimension(600, 640));

        panelMatrix.setMaximumSize(new Dimension(600, 710));

        JScrollPane scroll = new JScrollPane(panelMatrix);

        scroll.setPreferredSize(new Dimension(600, 640));

        scroll.setMinimumSize(new Dimension(600, 640));

        scroll.setMaximumSize(new Dimension(600, 710));

        pan.add(scroll, BorderLayout.EAST);


        JLabel labelInitial = new JLabel();

        labelInitial.setSize(580, 30);

        labelInitial.setAlignmentX(Component.CENTER_ALIGNMENT);

        panelMatrix.add(labelInitial);

        panelMatrix.add(Box.createRigidArea(new Dimension(0, 10)));

        panelMatrix.add(gridBefore);

        panelMatrix.add(Box.createRigidArea(new Dimension(0, 10)));

        JLabel labelTerminal = new JLabel();

        labelTerminal.setAlignmentX(Component.CENTER_ALIGNMENT);

        labelTerminal.setSize(580, 30);
```

```
panelMatrix.add(labelTerminal);
panelMatrix.add(Box.createRigidArea(new Dimension(0, 10)));
panelMatrix.add(gridAfter);

JPanel panelGraph = new JPanel();
panelGraph.setLayout(new BoxLayout(panelGraph, BoxLayout.PAGE_AXIS));
panelGraph.setPreferredSize(new Dimension(630, 640));
panelGraph.setMinimumSize(new Dimension(630, 640));
panelGraph.setMaximumSize(new Dimension(650, 710));

JPanel paintedGraphPanel = new JPanel();
paintedGraphPanel.setPreferredSize(new Dimension(630, 590));
paintedGraphPanel.setMaximumSize(new Dimension(650, 650));
paintedGraphPanel.setMinimumSize(new Dimension(630, 590));

JLabel labelGraph = new JLabel();
labelGraph.setAlignmentX(Component.CENTER_ALIGNMENT);
labelGraph.setSize(630, 30);
JScrollPane scrollG = new JScrollPane(panelGraph);
scrollG.setPreferredSize(new Dimension(630, 590));
scrollG.setMinimumSize(new Dimension(630, 590));
scrollG.setMaximumSize(new Dimension(650, 650));
panelGraph.add(labelGraph);
panelGraph.add(paintedGraphPanel);
pan.add(scrollG, BorderLayout.WEST);

fileButton.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) {
                log.fine("Choosing file");
                JFileChooser chooser = new JFileChooser();
                FileNameExtensionFilter filter = new FileNameExtensionFilter(
                                                                        "Text
files", "txt");
                chooser.setFileFilter(filter);
                int returnVal = chooser.showOpenDialog(null);
                if(returnVal == JFileChooser.APPROVE_OPTION) {
                    String filePath = "" + chooser.getCurrentDirectory();
                    String fileName = chooser.getSelectedFile().getName();
                    File mainFile = new File(filePath, fileName);
                    BufferedReader reader = null;
                    try {
                        reader = new BufferedReader(new FileReader(mainFile.getAbso-
lutePath()));
                        StringBuilder builder = new StringBuilder();
                        String currentLine = reader.readLine();
                        while (currentLine != null) {
                            builder.append(currentLine);
                            builder.append("\n");
                            currentLine = reader.readLine();
                        }
                        reader.close();
                        inputData = builder.toString();
                        flagButton = true;
                        panelGraph.setVisible(flagButton);
```

```
                    labelInitial.setText("Матрица на предыдущем шаге:");
                    labelTerminal.setText("Матрица после применения шага алго-
ритма:");
                    labelGraph.setText("Граф после применения шага алгоритма:");
                    initBut.setVisible(flagButton);
                    finishBut.setVisible(flagButton);
                    nextBut.setVisible(flagButton);
                    prevBut.setVisible(flagButton);
                    labelTerminal.setVisible(flagButton);
                    gridAfter.setVisible(flagButton);
                    lastPanel.setVisible(flagButton);
                    scroll.setVisible(flagButton);
                    scrollG.setVisible(flagButton);
                    try {
                        data = new WarshallAlgorithm(inputData);
                        data.setGraphData(inputData);
                        int count = data.verticesCount();
                        BoolMatrix m = new BoolMatrix(count, count);
                        for (int i = 0; i < count; i++) {
                            for (int j = 0; j < count; j++) {
                                m.set(i, j, false);
                            }
                        }
                        setMatrix(m, data.getMatrix(), count, gridBefore);
                        BoolMatrix tmpMatr = new BoolMatrix(data.getMatrix());
                        data.stepUp();
                        tmpMatr.sub(data.getMatrix());
                        setMatrix(tmpMatr, data.getMatrix(), count, gridAfter);
```

```
                    numVertices = count;
                    GraphDraw(paintedGraphPanel, new Dimension(630, 590));
                    frame.getContentPane().revalidate();
                    frame.getContentPane().repaint();
                    frame.pack();
                }catch(IllegalArgumentException i){
                    log.log(Level.INFO, "Exception: ", i);
                    JOptionPane.showMessageDialog(frame, "Wrong input data.");
                }
            } catch (FileNotFoundException ex) {
                log.log(Level.INFO, "Exception: ", ex);
                ex.printStackTrace();
            } catch (IOException ex) {
                log.log(Level.WARNING, "Exception: ", ex);
                ex.printStackTrace();
            }

        }
    }
});

firstButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        log.fine("Input data entered");
        JDialog dialog = createDialog(frame);
        dialog.setVisible(true);
        panelGraph.setVisible(flagButton);
```

```
labelInitial.setText("Матрица на предыдущем шаге:");
labelTerminal.setText("Матрица после применения шага алгоритма:");
labelGraph.setText("Граф после применения шага алгоритма:");
initBut.setVisible(flagButton);
finishBut.setVisible(flagButton);
nextBut.setVisible(flagButton);
prevBut.setVisible(flagButton);
labelTerminal.setVisible(flagButton);
gridAfter.setVisible(flagButton);
lastPanel.setVisible(flagButton);
scroll.setVisible(flagButton);
scrollG.setVisible(flagButton);
try{
    data = new WarshallAlgorithm(inputData);
    data.setGraphData(inputData);
    int count = data.verticesCount();
    BoolMatrix m = new BoolMatrix(count, count);
    for(int i = 0; i < count; i++){
        for(int j = 0; j < count; j++){
            m.set(i, j, false);
        }
    }
    setMatrix(m, data.getMatrix(), count, gridBefore);
    BoolMatrix tmpMatr = new BoolMatrix(data.getMatrix());
    data.stepUp();
    tmpMatr.sub(data.getMatrix());
    setMatrix(tmpMatr, data.getMatrix(), count, gridAfter);
```

```
                numVertices = count;
                GraphDraw(paintedGraphPanel, new Dimension(630, 590));
                frame.getContentPane().revalidate();
                frame.getContentPane().repaint();
                frame.pack();
            }catch(IllegalArgumentException i){
                log.log(Level.INFO, "Exception: ", i);
                JOptionPane.showMessageDialog(frame, "Wrong input data.");
            }

        }
    });

    nextBut.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            log.fine("Next button pressed");
            labelInitial.setText("Матрица на предыдущем шаге:");
            labelTerminal.setText("Матрица после применения шага алгоритма:");
            labelGraph.setText("Граф после применения шага алгоритма:");
            initBut.setVisible(true);
            prevBut.setVisible(true);
            labelTerminal.setVisible(true);
            gridAfter.setVisible(true);
            fileButton.setVisible(true);
            BoolMatrix m = new BoolMatrix(numVertices, numVertices);
            initBut.setVisible(true);
            for(int i = 0; i < numVertices; i++){
```

```
            for(int j = 0; j < numVertices; j++){
                m.set(i, j, false);
            }
        }
        setMatrix(m, data.getMatrix(), numVertices, gridBefore);
        BoolMatrix matr = new BoolMatrix(data.getMatrix());
        data.stepUp();
        matr.sub(data.getMatrix());
        setMatrix(matr, data.getMatrix(), numVertices, gridAfter);
        GraphDraw(paintedGraphPanel, new Dimension(630, 610));
        WarshallAlgorithm tmp = new WarshallAlgorithm(inputData);
        tmp.toFinalResult();
        if(tmp.getMatrix().equals(data.getMatrix())){
            nextBut.setVisible(false);
        }
    }
});

prevBut.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        log.fine("Prev button pressed");
        labelInitial.setText("Матрица на предыдущем шаге:");
        labelTerminal.setText("Матрица после применения шага алгоритма:");
        labelGraph.setText("Граф после применения шага алгоритма:");
        initBut.setVisible(true);
        finishBut.setVisible(true);
        nextBut.setVisible(true);
```

```
            labelTerminal.setVisible(true);
            gridAfter.setVisible(true);
            BoolMatrix matr = new BoolMatrix(data.getMatrix());
            BoolMatrix otherMatr = new BoolMatrix(data.getMatrix());
            data.stepDown();
            BoolMatrix m = new BoolMatrix(numVertices, numVertices);
            for(int i = 0; i < numVertices; i++){
                for(int j = 0; j < numVertices; j++){
                    m.set(i, j, false);
                }
            }
            matr.sub(data.getMatrix());
            finishBut.setVisible(true);
            setMatrix(matr, otherMatr, numVertices, gridAfter);
            setMatrix(m, data.getMatrix(), numVertices, gridBefore);
            GraphDraw(paintedGraphPanel, new Dimension(630, 610));
            WarshallAlgorithm tmp = new WarshallAlgorithm(inputData);
            tmp.toStart();
            if(tmp.getMatrix().equals(data.getMatrix())){
                prevBut.setVisible(false);
            }
        }
    });

    finishBut.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            log.fine("Finish button pressed");
            labelInitial.setText("Матрица смежности окончательного результата:");
```

```
            labelGraph.setText("Граф окончательного результата:");
            labelTerminal.setVisible(false);
            gridAfter.setVisible(false);
            nextBut.setVisible(false);
            finishBut.setVisible(false);
            initBut.setVisible(true);
            prevBut.setVisible(true);
            BoolMatrix m = new BoolMatrix(numVertices, numVertices);
            for(int i = 0; i < numVertices; i++){
                for(int j = 0; j < numVertices; j++){
                    m.set(i, j, false);
                }
            }
            data.toFinalResult();
            GraphDraw(paintedGraphPanel, new Dimension(630, 610));
            setMatrix(m, data.getMatrix(), numVertices, gridBefore);
        }
    });

    initBut.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            log.fine("Init button pressed");
            finishBut.setVisible(true);
            nextBut.setVisible(true);
            BoolMatrix m = new BoolMatrix(numVertices, numVertices);
            for(int i = 0; i < numVertices; i++){
                for(int j = 0; j < numVertices; j++){
```

```
                    m.set(i, j, false);
                }
            }
            data.toStart();
            setMatrix(m, data.getMatrix(), numVertices, gridBefore);
            GraphDraw(paintedGraphPanel, new Dimension(630, 610));
            labelTerminal.setVisible(false);
            labelInitial.setText("Матрица смежности начального состояния:");
            labelGraph.setText("Граф начального состояния:");
            prevBut.setVisible(false);
            initBut.setVisible(false);
            gridAfter.setVisible(false);
        }
    });
    panelGraph.setVisible(flagButton);
    scroll.setVisible(flagButton);
    scrollG.setVisible(flagButton);
    lastPanel.setVisible(flagButton);
    pan.revalidate();
    pan.repaint();
    pan.setOpaque(true);
    frame.setContentPane(pan);
    frame.pack();
    frame.setVisible(true);
}

public void GraphDraw(JPanel mainPanel, Dimension DEFAULT_SIZE){
    log.fine("Drawing graph");
```

```
        mainPanel.removeAll();

        mainPanel.revalidate();

        mainPanel.repaint();


        mainPanel.setPreferredSize(DEFAULT_SIZE);

        mainPanel.setMinimumSize(DEFAULT_SIZE);

        mainPanel.setMaximumSize(DEFAULT_SIZE);

        ListenableGraph <String, DefaultEdge> g = new DefaultListenableGraph<>(new De-
faultDirectedGraph<>(DefaultEdge.class));


        JGraphXAdapter<String, DefaultEdge> jgxAdapter = new JGraphXAdapter<>(g);

        mxGraphComponent component = new mxGraphComponent(jgxAdapter);

        component.setConnectable(false);

        component.getGraph().setAllowDanglingEdges(false);

        component.createHorizontalScrollBar();

        component.createVerticalScrollBar();

        mainPanel.add(component);


        String[] verteces = new String[numVertices];

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

            String v = "\t" + data.getVertex(i) + "\t";

            verteces[i] = v;

            g.addVertex(v);

        }


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

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

                if(data.getMatrix().get(i, j) == true){

                    g.addEdge(verteces[i], verteces[j]);
```

```
                }

            }

        }

        mxCircleLayout layout = new mxCircleLayout(jgxAdapter);

        int radius = 250;

        layout.setX0(DEFAULT_SIZE.width / 16.0);

        layout.setY0(DEFAULT_SIZE.height / 16.0);

        layout.setRadius(radius);

        layout.setMoveCircle(true);


        layout.execute(jgxAdapter.getDefaultParent());

        log.fine("Finished drawing graph");

    }


    private void setMatrix(BoolMatrix matr, BoolMatrix otherMatr, int count, JPanel
grid){

        log.fine("Setting Matrix");

        grid.removeAll();

        grid.revalidate();

        grid.repaint();

        grid.setPreferredSize(new Dimension(600, 270));

        grid.setMaximumSize(new Dimension(600, 320));

        grid.setMinimumSize(new Dimension(600, 270));

        grid.setBackground(Color.white);

        grid.setLayout(new GridLayout(count + 1, count + 1));

        grid.setBorder(BorderFactory.createLineBorder(Color.black));


        JLabel labelEmpty = new JLabel("");

        labelEmpty.setHorizontalAlignment(JLabel.CENTER);
```

```
labelEmpty.setVerticalAlignment(JLabel.CENTER);
labelEmpty.setBorder(BorderFactory.createLineBorder(Color.black));
grid.add(labelEmpty);

for(int j = 0; j < count; j++){
    char c = data.getVertex(j);
    JLabel label = new JLabel("" + c);
    label.setHorizontalAlignment(JLabel.CENTER);
    label.setVerticalAlignment(JLabel.CENTER);
    label.setBorder(BorderFactory.createLineBorder(Color.black));
    grid.add(label);
}

for(int i = 0; i < count; i++){
    char c = data.getVertex(i);
    JLabel labelVert = new JLabel("" + c);
    labelVert.setHorizontalAlignment(JLabel.CENTER);
    labelVert.setVerticalAlignment(JLabel.CENTER);
    labelVert.setBorder(BorderFactory.createLineBorder(Color.black));
    grid.add(labelVert);
    for(int j = 0; j < count; j++){
        int k = otherMatr.get(i, j) ? 1 : 0;
        JLabel label = new JLabel("" + k);
        if(matr.get(i, j) == true){
            label.setBackground(Color.red);
            label.setForeground(Color.white);
            label.setOpaque(true);
        }
```

```
                label.setHorizontalAlignment(JLabel.CENTER);
                label.setVerticalAlignment(JLabel.CENTER);
                label.setBorder(BorderFactory.createLineBorder(Color.black));
                grid.add(label);
            }
        }
        return;
    }

    private JDialog createDialog(JFrame frame){
        log.fine("Creating input dialog");
        JDialog dialog = new JDialog(frame);
        dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        dialog.setModal(true);
        dialog.setSize(500, 500);
        JTextArea area = new JTextArea(20, 30);

        JLabel label = new JLabel("<html><div style='text-align: center;'>" + "Введите
ребра в виде пар вершин через пробел: " + "<br>" + " вершина_из_которой_исходит_ребро
вершина_в_которую_входит_ребро." + "</div></html>");
        label.setPreferredSize(new Dimension(480, 60));
        label.setMaximumSize(new Dimension(480, 60));
        label.setMinimumSize(new Dimension(480, 60));
        label.setAlignmentX(Component.CENTER_ALIGNMENT);
        label.setOpaque(true);

        JPanel panel = new JPanel();
        panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
        panel.setLayout(new BoxLayout(panel, BoxLayout.PAGE_AXIS));
```

```
panel.setMaximumSize(new Dimension(480, 480));
panel.setMinimumSize(new Dimension(480, 480));
panel.setPreferredSize(new Dimension(480, 480));

panel.add(label);
panel.add(Box.createRigidArea(new Dimension(480,10)));
JScrollPane scroll = new JScrollPane(area);
scroll.setPreferredSize(new Dimension(480, 360));
scroll.setMaximumSize(new Dimension(480, 360));
scroll.setMinimumSize(new Dimension(480, 360));
scroll.setOpaque(true);
panel.add(scroll);

JButton firstButton = new JButton("OK");
panel.setMaximumSize(new Dimension(60, 30));
panel.setMinimumSize(new Dimension(60, 30));
panel.setPreferredSize(new Dimension(60, 30));
panel.add(firstButton);
firstButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        inputData = area.getText();
        if(inputData.length() > 0){
            flagButton = true;
        }
        dialog.dispose();
    }
});
```

```
        panel.add(Box.createGlue());
        panel.setOpaque(true);
        dialog.setContentPane(panel);
        dialog.setVisible(true);
        return dialog;
    }
};
```

WarshallAlgorithm.java:

```
import java.util.*;
import java.util.logging.*;

public class WarshallAlgorithm {
    private static Logger log = Logger.getLogger(WarshallAlgorithm.class.getName());
    private Graph g;
    private Vector<Graph.Memento> history;
    private int curState;
    private boolean isCompleted;

    public WarshallAlgorithm(String edges) {
        setGraphData(edges);
    }

    public void setGraphData(String data) {
        log.fine("Starting parsing string to create Graph");
        isCompleted = false;
        curState = 0;
        history = new Vector<Graph.Memento>();
        Scanner s = new Scanner(data);
        s.useDelimiter("\n");
```

```
        String token;
        StringBuilder alp = new StringBuilder();
        Vector<Graph.Edge> edges = new Vector<Graph.Edge>();
        int i;
        char source, destination;
        while(s.hasNext(" *[a-z] +[a-z] *")) {
            token = s.next(" *[a-z] +[a-z] *");
            log.fine(String.format("Parsing token: %s", token));
            i = 0;
            while(Character.isSpaceChar(token.charAt(i)))
                ++i;
            source = token.charAt(i);
            if(alp.indexOf(String.valueOf(source)) == -1)
                alp.append(source);
            i++;
            while(Character.isSpaceChar(token.charAt(i)))
                ++i;
            destination = token.charAt(i);
            if(alp.indexOf(String.valueOf(destination)) == -1)
                alp.append(destination);
            edges.add(new Graph.Edge(source, destination));
        }
        if(s.hasNext()) {
            IllegalArgumentException e = new IllegalArgumentException("Incorrect input
data");
            throw e;
        }
```

```
    g = new Graph(alp.toString().toCharArray(), edges);
    history.add(g.save());
}

public void transitiveClosure() {
    g.transitiveClosure();
}

public void stepUp() {
    log.fine("Going to next step");
    if(curState + 1 < history.size()) {
         g.restore(history.elementAt(curState + 1));
    }
    else {
        if(isCompleted && curState + 1 == history.size())
            return;
        isCompleted = g.stepTransitiveClosure();
        if(isCompleted)
            return;
        history.add(g.save());
    }
    ++curState;
}

public void stepDown() {
    log.fine("Going to previous step");
    if(curState == 0)
        return;
    g.restore(history.elementAt(curState - 1));
```

```
    --curState;
}

public void toStart() {
    log.fine("Going to starting step");
    g.restore(history.elementAt(0));
    curState = 0;
}

public void toFinalResult() {
    log.fine("Going to final result");
   g.restore(history.elementAt(history.size()-1));
   if(!isCompleted) {
       while(!(isCompleted = g.stepTransitiveClosure()))
           history.add(g.save());
   }
   curState = history.size() - 1;
}

public boolean completed() {
    return isCompleted;
}

public int verticesCount() {
    return g.verticesCount();
}

public char getVertex(int Index) {
    return g.getVertex(Index);
```

```
    }

    public Graph.Edge[] getEdges() {
        return g.getEdges();
    }

    public BoolMatrix getMatrix() {
        return g.getMatrix();
    }

    public String toString() {
        return g.toString();
    }
}
```

Graph.java:

```
import java.util.*;
import java.util.logging.*;

public class Graph{
    private static Logger log = Logger.getLogger(Graph.class.getName());
    public static class Edge{
        public char source;
        public char destination;
        public Edge(char source, char destination) {
            this.source = source;
            this.destination = destination;
        }
    }
    public class Memento {
```

```
    private String vertices;
    private BoolMatrix matr;
    public Memento(String vertices, BoolMatrix matr) {
        this.matr = new BoolMatrix(matr);
        this.vertices = new String(vertices);
    }
    public String getVertices() {
        return vertices;
    }
    public BoolMatrix getMatrix() {
        return new BoolMatrix(matr);
    }
}

private String vertices;
private BoolMatrix m;

public Graph(char[] vertices) {
    this.vertices = String.valueOf(vertices);
    m = new BoolMatrix(vertices.length, vertices.length);
}

public Graph(char[] vertices, Collection<Edge> edges) {
    this.vertices = String.valueOf(vertices);
    m = new BoolMatrix(vertices.length, vertices.length);
    Iterator<Edge> i = edges.iterator();
    while(i.hasNext())
        add(i.next());
}
```

```
    public void add(Edge edge) {

        log.fine(String.format("Adding edge: %c %c", edge.source, edge.destination));

        if(vertices.indexOf(edge.source) == -1 || vertices.indexOf(edge.destination) ==
-1) {

            throw new IllegalArgumentException("Cannot add edge: one of the vertices
isn't in the graph");

        }

        m.set(vertices.indexOf(edge.source), vertices.indexOf(edge.destination), true);

    }


    public int verticesCount() {

        return vertices.length();

    }


    public char getVertex(int Index) {

        return vertices.charAt(Index);

    }


    public Edge[] getEdges() {

        Vector<Edge> edges = new Vector<Edge>();

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

            for(int j=0; j<vertices.length(); ++j)

                if(m.get(i, j))

                    edges.add(new Edge(vertices.charAt(i), vertices.charAt(j)));

        return (Edge[])edges.toArray();

    }


    public BoolMatrix getMatrix() {

        return new BoolMatrix(m);
```

```
}

public Memento save() {
    log.fine("Saving memento");
    return new Memento(vertices, m);
}

public void restore(Memento state) {
    log.fine("Restoring graph");
    m = state.getMatrix();
    vertices = state.getVertices();
}

public String toString() {
    return m.toString();
}

public void transitiveClosure() {
    log.fine("Doing all algorithm");
    BoolMatrix prev;
    do {
        prev = m;
        m = m.multiply(m);
        m.add(prev);
    } while(!m.equals(prev));
}

public boolean stepTransitiveClosure() {
    log.fine("Doing one step");
    BoolMatrix prev;
```

```
        prev = m;
        m = m.multiply(m);
        m.add(prev);
        return prev.equals(m);
    }
}
```

BoolMatrix.java:

```
public class BoolMatrix{
    private int rowCount;
    private int columnCount;
    private boolean[] matr;

    public BoolMatrix(int rowCount, int columnCount) {
        if(rowCount < 1 || columnCount < 1)
            throw new IllegalArgumentException("Matrix scales must be positive");
        this.rowCount = rowCount;
        this.columnCount = columnCount;
        matr = new boolean[rowCount*columnCount];
    }

    public BoolMatrix(BoolMatrix m) {
        rowCount = m.rowCount;
        columnCount = m.columnCount;
        matr = new boolean[rowCount*columnCount];
        for(int i=0; i<matr.length; ++i)
            matr[i] = m.matr[i];
    }
```

```
    public void set(int rowIndex, int columnIndex, boolean value) {
        matr[rowIndex*columnCount + columnIndex] = value;
    }

    public boolean get(int rowIndex, int columnIndex) {
        return matr[rowIndex*columnCount + columnIndex];
    }

    public int getRowCount() {
        return rowCount;
    }

    public int getColumnCount() {
        return columnCount;
    }

    public void add(BoolMatrix m) {
        for(int i=0; i<matr.length; ++i)
            matr[i] = matr[i] || m.matr[i];
    }

    public void sub(BoolMatrix m) {
        for(int i=0; i<matr.length; ++i)
            matr[i] ^= m.matr[i];
    }

    public BoolMatrix multiply(BoolMatrix m) {
        if(columnCount != m.rowCount)
            throw new IllegalArgumentException("Column count of first matrix must be
equal to row count of second matrix");
```

```
    BoolMatrix res = new BoolMatrix(rowCount, m.columnCount);
    for(int i=0; i<rowCount; ++i){
        for(int j=0; j<res.columnCount; ++j){
            for(int k=0; k<columnCount; ++k){
                if(matr[i*columnCount + k] && m.matr[k*m.columnCount + j]){
                    res.matr[i*res.columnCount + j] = true;
                    break;
                }
            }
        }
    }
    return res;
}

public String toString() {
    StringBuilder str = new StringBuilder();
    for(int i=0; i<rowCount; ++i){
        for(int j=0; j<columnCount; ++j){
            str.append(matr[i*columnCount + j] ? '1' : '0');
            str.append(' ');
        }
        str.append(System.lineSeparator());
    }
    return str.toString();
}

public boolean equals(Object o) {
    if(!(o instanceof BoolMatrix))
```

```
             return false;

        BoolMatrix m = (BoolMatrix) o;

        if(rowCount != m.rowCount || columnCount != m.columnCount)

            return false;

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

            if(matr[i] != m.matr[i])

                return false;

        return true;

    }

}
```

WarshallTest.java:

```
import com.company.WarshallAlgorithm;

import org.junit.*;

import static org.junit.Assert.*;

import java.io.*;


public class WarshallTest {

    private String path = "C:/Users/nadez/IdeaProjects/GUIlast/test/file.txt"; //файл с
тестом

    private String str1 = "a b\nb c\n c j\nj a";

    private String str2 = "a b\nb c\n";

    private String res1 = "1 1 1 1 \r\n" + "1 1 1 1 \r\n" + "1 1 1 1 \r\n" + "1 1 1 1
\r\n";

    private String res2 = "0 1 1 \r\n0 0 1 \r\n0 0 0 \r\n";


    @Test(expected = IllegalArgumentException.class)

    public void testsetGraphData() { //проверка ввода данных и построение матрицы смеж-
ности
```

```
        WarshallAlgorithm graph3 = new WarshallAlgorithm("0 0\na b\n");

        WarshallAlgorithm graph2 = new WarshallAlgorithm("A B\n"); //введены  названия
вершин русским алфавитом

        WarshallAlgorithm graph1 = new WarshallAlgorithm(str1);

        WarshallAlgorithm graph4 = new WarshallAlgorithm("a bh\nb c\n co j\nj a");

        WarshallAlgorithm graph5 = new WarshallAlgorithm("a b h\nb c\n co j\nj a");


        assertEquals("Неверно строится матрица смежности", graph1.toString(), "0 1 0 0
\r\n" +

                "0 0 1 0 \r\n" + "0 0 0 1 \r\n" + "1 0 0 0 \r\n");


        graph1 = new WarshallAlgorithm("a v \n v g\n");


        assertEquals("Неверно строится матрица смежности", graph1.toString(), "0 1 0
\r\n" +

                "0 0 1 \r\n0 0 0 \r\n");

    }

    @Test

    public void test_transitiveClosure() { //проверка алгоритма

        WarshallAlgorithm graph1 = new WarshallAlgorithm(str1);

        WarshallAlgorithm graph2 = new WarshallAlgorithm(str2);

        graph1.transitiveClosure();

        graph2.transitiveClosure();

        assertEquals("Неверная работа Алгоритма", graph1.toString(), res1);

        assertEquals("Неверная работа Алгоритма", graph2.toString(), res2);

    }

    @Test

    public void test_stepUp() { //проверка перехода на шаг вперед

        String str3 = "a g\n g h\n h g\n";

        WarshallAlgorithm graph1 = new WarshallAlgorithm(str1);
```

```
        WarshallAlgorithm graph2 = new WarshallAlgorithm(str3);

        graph1.stepUp(); //переход на шаг вперед

        graph2.stepUp();

        assertEquals("Неверно посчитан следующий шаг", graph1.toString(), "0 1 1 0
\r\n" +

                "0 0 1 1 \r\n" + "1 0 0 1 \r\n" + "1 1 0 0 \r\n");

        assertEquals("Неверно посчитан следующий ша", graph2.toString(), "0 1 1 \r\n0 1
1 \r\n0 1 1 \r\n");

    }

    @Test

    public void test_stepDown() { //проверка перехода на шаг назад

        WarshallAlgorithm graph1 = new WarshallAlgorithm(str1);

        WarshallAlgorithm graph2 = new WarshallAlgorithm(str1);

        graph1.stepUp(); //переход на шаг вперед

        graph1.stepDown(); //переход на шаг назад

        assertEquals("Неверно посчитан предыдущий шаг", graph1.toString(),
graph2.toString());

    }

    @Test

    public void test_toFinalResult() { //Проверка перехода к конечному состоянию

        WarshallAlgorithm graph1 = new WarshallAlgorithm(str2);

        WarshallAlgorithm graph2 = new WarshallAlgorithm(str1);

        graph1.stepUp(); //переход на шаг вперед

        graph1.stepUp();

        graph1.toFinalResult(); //переход к конечному результату не с начального состо-
яния

        assertEquals("Неверный конечный результат", graph1.toString(), res2);


        graph2.stepUp(); //переход на шаг вперед

        graph2.stepUp();
```

```
        graph2.toFinalResult(); //переход к конечному результату не с начального состояния

        assertEquals("Неверный конечный результат", graph2.toString(), res1);

    }

    @Test

    public void test_toStart() {

        WarshallAlgorithm graph1 = new WarshallAlgorithm(str2);

        WarshallAlgorithm graph2 = new WarshallAlgorithm(str2);

        graph1.transitiveClosure();

        graph1.toStart(); //переход к начальному состоянию

        assertEquals("Неверный возврат к начальным значениям", graph1.toString(), graph2.toString());

    }

    @Test

    public void test_file() {

        String inputData = "";

        try {

            FileInputStream fstream = new FileInputStream(path);

            BufferedReader reader = new BufferedReader(new InputStreamReader(fstream));

            StringBuilder builder = new StringBuilder();

            String currentLine = reader.readLine();

            while (currentLine != null) {

                builder.append(currentLine);

                builder.append("\n");

                currentLine = reader.readLine();

            }

            reader.close();

            inputData = builder.toString();
```

```
            try {
                WarshallAlgorithm graph = new WarshallAlgorithm(inputData);
            } catch (IllegalArgumentException ex) {
                System.out.println("Wrong input data.");
            }
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```