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

ОТЧЕТ

по учебной практике

Тема: Визуализация алгоритма Флойда-Уоршелла

Студент гр. 7381 _____ Кортев Ю.В.

Студент гр. 7382 _____ Кравченко П.В.

Студент гр. 7383 _____ Васильев А.И.

Руководитель _____ Размочаева Н.В.

Санкт-Петербург

2019

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Кортев Ю.В. группы 7381
Студент Кравченко П.В. группы 7382
Студент Васильев А.И. группы 7383

Тема практики: визуализация алгоритма Флойда-Уоршелла

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.

Алгоритм: Флойда-Уоршелла поиска кратчайших путей между всеми вершинами графа

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета: 15.07.2019

Дата защиты отчета: 15.07.2019

Студент	_____	Кортев Ю.В.
Студент	_____	Кравченко П.В.
Студент	_____	Васильев А.И.
Руководитель	_____	Размочаева Н.В.

АННОТАЦИЯ

Целью практики являлось создание визуализатора алгоритма. На протяжении практики работа велась по итеративному плану. Каждый член команды имел свои обязанности. Был написан код визуализируемого алгоритма, так же был создан пользовательский интерфейс. Было проведено тестирование для проверки корректности составляющих программ и по результатам тестирования внесены изменения в программу с целью устранения найденных недостатков.

SUMMARY

The purpose of the practice was to create an algorithm visualizer. Throughout the practice, work was carried out according to an iterative plan. Each team member had their own responsibilities. The code of the visualized algorithm was written, the user interface was also created. Testing was conducted to verify the correctness of the constituent programs and, according to the results of testing, changes were made to the program in order to eliminate the deficiencies found.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью практики является освоение навыков разработки программ в команде.

Задачей практики является создание визуализатора алгоритма Флойда-Уоршелла поиска кратчайших путей между всеми вершинами графа на языке Java.

На каждом шаге алгоритм Флойда-Уоршелла генерирует матрицу, которая содержит длины кратчайших путей между всеми вершинами графа. Перед работой алгоритма матрица заполняется длинами ребер графа (или запредельно большим числом, если ребра нет) [1].

Алгоритм Флойда-Уоршелла является эффективным для расчета всех кратчайших путей в плотных графах, когда имеется большое количество пар ребер между парами вершин. В случае разреженных графов с ребрами неотрицательного веса лучшим выбором считается использование алгоритма Дейкстры для каждой возможной вершины [3].

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><вес ребра>.

Именем вершины могут быть строчные буквы латинского алфавита (от ’a’ до ‘z’).

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, в главном окне программы:

- Ввод графа из файла – при нажатии на кнопку пользователь выбирает путь до файла и с файла считываются входные данные;
- Ввод с клавиатуры – при нажатии на кнопку пользователь вручную с клавиатуры вводит данные;
- Ввести графически – при нажатии на кнопку открывается окно графического ввода в котором пользователь задает исходный граф;
- Запустить алгоритм — при нажатии на кнопку алгоритм начинает работу.

В окне, отображающем состояние программы:

- В начало – при нажатии на кнопку показывается граф, перед началом работы алгоритма;
- Прошлый шаг – при нажатии на кнопку алгоритм возвращается на предыдущую итерацию;
- Следующий шаг – при нажатии на кнопку алгоритм переходит на следующую итерацию;

- В конец — при нажатии на кнопку алгоритм проходит все итерации и завершает работы, при этом выводится матрица кратчайших путей;

В окне графического ввода:

- Назад – при нажатии на кнопку пользователь возвращается в главное окно и может запустить алгоритм на заданном графе;
- NewE — добавляет в граф новое ребро (при нажатии как этой кнопки, так и следующей, требуется выбрать две вершины, между которыми нужно добавить или удалить ребро);
- DelE — удаляет ребро из графа;
- NewV — добавляет вершину в граф;
- DelV — удаляет вершину из графа;

1.2. Уточнение требований после прототипа

Необходимо сделать Unit тесты (минимум 10 штук), реализовать пошаговое выполнение алгоритма.

1.3. Уточнение после первой итерации

Внизу графического окна сделать строку с указанием авторов.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

План разработки делится на несколько этапов.

Этап 1: Разработка спецификации. На этом этапе должны быть определены требования к программе. Срок реализации до 3 июля.

Этап 2: Прототип. На этом этапе производится демонстрация окна пользовательского интерфейса программы без функционала. Срок реализации до 5 июля.

Этап 3: Итерация 1. На этом этапе должны быть реализованы все окна программы. Все кнопки интерфейса должны будут реализовывать только открытие или закрытие соответствующих им окон, остальные кнопки отключены. Так же должен будет быть реализован пошаговый проход алгоритма в соответствующей области без возможности отмены действий. Срок реализации до 8 июля.

Этап 4: Итерация 2. Должно быть реализовано отображение графа в соответствующих полях на главном окне и на окне, отображающем текущее состояние. Должен быть добавлен ввод графа с помощью графического интерфейса. Срок реализации до 10 июля.

Этап 5: Финальная версия. В программу должна быть добавлена функция возврата к предыдущему шагу алгоритма. Так же должны быть проведены тесты и устранены недостатки программы, предоставлен отчет. Срок выполнения до 12 июля.

2.2. Распределение ролей в бригаде

Кортев Юрий — реализация алгоритма и визуализация графа, тестирование алгоритма;

Кравченко Павел — реализация связи между алгоритмом и графическим интерфейсом, написание отчета;

Васильев Артем — реализация графического интерфейса, тестирование графического интерфейса.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Использованные структуры данных и методы

Для описания графа используется матрица смежности. Был написан абстрактный класс Graph, а так же класс SimpleGraph, который был наследован от класса Graph и реализующий сам граф. Так же в классе SimpleGraph реализованы методы по построению графа по различным входным данным.

Алгоритм реализован в классе Algorithm. Методы по работе с графическим интерфейсом реализованы в остальных классах.

3.2. Работа графического интерфейса

При запуске программы появляется главное окно, предлагающее на выбор 3 варианта ввода исходных данных: «Ввод с файла», «Ввод с клавиатуры», «Ввести графически». При нажатии на кнопку «Ввод с файла» появляется окно, где можно выбрать любой текстовый файл. При нажатии на кнопку «Ввод с клавиатуры» пользователь во всплывшем окне задает количество вершин и расстояния между смежными вершинами. При нажатии на кнопку «Ввести графически» пользователь самостоятельно «рисует» граф в специально открывшемся для этого окне.

4. ТЕСТИРОВАНИЕ

Запуск и тестирование графического интерфейса программы происходило в среде разработки IntelliJ IDEA [4]. Все кнопки графического интерфейса данной программы работают корректно. В ходе тестирования все выявленные ошибки были устранены. В финальном тестировании ошибок выявлено не было.

ЗАКЛЮЧЕНИЕ

В ходе практической работы требовалось построить визуализатор алгоритма. В качестве результата практической работы был разработан и написан визуализатор алгоритма Флойда-Уоршелла поиска кратчайших путей между всеми вершинами на языке Java. Были освоены на практике навыки разработки программ в команде. Был написан код реализуемого алгоритма на языке Java. Был разработан и реализован пользовательский интерфейс для визуального представления работы алгоритма. Было проведено тестирование программы на предмет некорректной работы и исправлены ошибки по результатам тестирования. Так же бригада не смогла до конца уложиться в сроки плана, разработанного перед началом работы над проектом, причиной этому послужило отсутствие опыта разработки проектов в команде.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффод Штайн. Алгоритмы: построение и анализ. М.: Вильямс, 2006. 1296 с.

2. Белоусов А.И., Ткачев С.Б. Дискретная математика. М.: МГТУ, 2006. 744с.

3. Алгоритм Флойда-Уоршелла // Википедия URL: https://ru.wikipedia.org/wiki/Алгоритм_Флойда_-_Уоршелла (дата обращения: 04.07.2019).

4. Petar Tahchiev. «Junit in Action». «Manning Publications», 2010.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Main.java:

```
package Window;

import javax.swing.*;
import java.awt.*;

import static Window.Windows_par.*;

public class Main extends JFrame{

    public static void main(String[] args) {new Main(); }

    Main(){
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        setTitle("Алгоритм Форда-Уоршела");      //Имя окна
        setSize( WINDOW_SIZE );                            //Размер окна

        setResizable(false);

       add(new GraphPlane());


        setVisible(true);
    }
}
```

Algorithm.java:

```
package Window;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
import java.util.HashMap;

import static Window.Windows_par.*;

public class Algorithm {
```

```
    GraphPlane graph;
    int[][]adjMatr;
    int i=0;
    ArrayList<Alg_har> history=new ArrayList<>();

    Algorithm(GraphPlane graph,int[][]matr){
        this.graph=graph;
        this.adjMatr=matr;
        while(!graph.points.containsKey(i))i++;
        graph.points.get(i).color=CIRCLE_BORDERLINE_COLOR_IN_ALG;
        graph.repaint();
    }

    public boolean step(){
        if(i==adjMatr.length) return false;
        while(!graph.points.containsKey(i))i++;
        history.add(new Alg_har(this.graph.points,this.adjMatr));
        for (int v = 0; v < adjMatr.length; v++) {
            for (int u = 0; u < adjMatr.length; u++) {
                if(graph.points.containsKey(v) && graph.points.containsKey(u))
                    adjMatr[v][u] = Math.min(adjMatr[v][u], adjMatr[v][i] +
adjMatr[i][u]);
            }
        }
        graph.points.get(i).color=CIRCLE_BORDERLINE_COLOR_BASE;
        i++;
        while(!graph.points.containsKey(i)&&i!=adjMatr.length)i++;

if(graph.points.containsKey(i))graph.points.get(i).color=CIRCLE_BORDERLINE_COLOR
_IN_ALG;
        graph.repaint();
        return true;
    }

    public void step_back(){
        if(i==0){
            JOptionPane.showMessageDialog(graph,
                    "Ошибка: начало алгоритма",
                    "Ошибка алгоритма",
                    JOptionPane.ERROR_MESSAGE);
        }
```

```
        else{
            Alg_har back=history.get(history.size()-1);
            history.remove(history.size()-1);
            this.adjMatr=back.adjMatr;
            for(int i=0;i<graph.countV;i++)

if(graph.points.containsKey(i))graph.points.get(i).color=back.points.get(i);
            graph.repaint();
            i--;
        }
    }

    public void go_to_start(){//откал в начало
        if(history.isEmpty()){
            JOptionPane.showMessageDialog(graph,
                    "Ошибка: уже начало алгоритма",
                    "Ошибка алгоритма",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        Alg_har back=history.get(0);
        for(int i=0;i<graph.countV;i++)
if(graph.points.containsKey(i))graph.points.get(i).color=back.points.get(i);
        adjMatr=back.adjMatr;
        history.clear();

    }

    public int[][] result(){
        while(step());
        return adjMatr;
    }

    public class Alg_har{
        public HashMap<Integer, Color> points;
        public int[][]adjMatr;
        Alg_har(HashMap<Integer,VertexViz>points,int[][]adjMatr){
            this.points=new HashMap<>();
            for(int i=0;i<graph.countV;i++)
                if(points.containsKey(i))this.points.put(i,points.get(i).color);
            this.adjMatr=adjMatr.clone();
```

```
        }
    }

}
```

Filer.java:

```
package Window;

import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import javax.swing.*;
import javax.swing.text.html.parser.Parser;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.FileNotFoundException;

public class Filer {
    public Filer(GraphPlane graph) {
        JFileChooser fileopen = new JFileChooser("D:\\OneDrive\\JetBrainPro\\
MainWindow_1\\MainWindow");
        int ret = fileopen.showDialog(null, "Открыть файл");
        if (ret == JFileChooser.APPROVE_OPTION) {
            File file = fileopen.getSelectedFile();
            try {
                FileReader reader = new FileReader(file.getName());
                JSONParser jsonParser = new JSONParser();
                JSONObject jsonObject = (JSONObject) jsonParser.parse(reader);
                graph.fromFile((JSONObject) jsonObject);
            } catch (FileNotFoundException ex) {
                ex.printStackTrace();
            } catch (IOException ex) {
                ex.printStackTrace();
            } catch (ParseException ex) {
                ex.printStackTrace();
            } catch (NullPointerException ex) {
                ex.printStackTrace();
            }
        }
```

```
    }
}
```

GraphPlane.java:

```
package Window;

import Graph.*;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import javax.swing.*;
import java.awt.*;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.math.*;
import java.util.Iterator;

import static Window.Windows_par.*;

public class GraphPlane extends JPanel {
    final static int INF=10_000_000;
    int countV=0;
    int countE=0;
    HashMap<Integer,VertexViz> points=new HashMap<>();
    HashMap<Integer,Graph.Edge> edges=new HashMap<>();
    Graph graph=new SimpleGraph();
    Algorithm floyd_warshell=null;

    public GraphPlane() {
        setLayout(null);
        setPreferredSize(SIZE_OF_GRAPH_FIELD);    //Размер рамки
        setBackground(BACKGROUND);

    }

    private void print_matr(int[][] matr){
```

```
        for(int i=0;i<matr.length;i++){
            for(int j=0;j<matr.length;j++){
                if(matr[i][j]==INF)
                    System.out.print("INF  ");
                else
                    System.out.print(matr[i][j]+"    ");
            }
            System.out.print('\n');
        }
        System.out.println();
    }

    private int[][] list_in_matrix(boolean is_alg){
        int[][] matr=new int[countV][countV];
        for(int i=0;i<countV;i++){
            for(int j=0;j<countV;j++){
                if(i!=j) matr[i][j]=is_alg ? INF : 0;
                else {
                    if(points.containsKey(i)) matr[i][j]=0;
                    else matr[i][j]=INF;
                }
            }
        }

        for(int i=0;i<countE;i++){
            if(edges.containsKey(i))matr[edges.get(i).v1]
[edges.get(i).v2]=edges.get(i).weight;
        }
        return matr;
    }

    public int[][] get_matr(){  //возвращает массив
        return list_in_matrix(false);
    }

    public void start_alg(){    //начать алгоритм(кнопка)
        this.floyd_warshell=new Algorithm(this,list_in_matrix(true));
    }

    public void go_to_start(){//кнопка в начало алг
        if (floyd_warshell == null) {
```

```
            JOptionPane.showMessageDialog(this,
                    "Ошибка: алгоритм еще не запущен",
                    "Ошибка алгоритма",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        floyd_warshell.go_to_start();
    }

    public void step() { //кнопка шага вперед
        if (floyd_warshell == null) {
            JOptionPane.showMessageDialog(this,
                    "Ошибка: алгоритм еще не запущен",
                    "Ошибка алгоритма",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (!floyd_warshell.step()) {
            JOptionPane.showMessageDialog(this,
                    "Ошибка: алгоритм уже закончил работу",
                    "Ошибка алгоритма",
                    JOptionPane.ERROR_MESSAGE);
        }
    }       //шаг вперед(кнопка)

    public void step_back(){
        if (floyd_warshell == null) {
            JOptionPane.showMessageDialog(this,
                    "Ошибка: алгоритм еще не запущен",
                    "Ошибка алгоритма",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        floyd_warshell.step_back();
    }   //шаг назад

    public int[][] result(){    //алгоритм сразу
        if(floyd_warshell==null){
            JOptionPane.showMessageDialog(this,
                    "Ошибка: алгоритм еще не запущен",
```

```
                    "Ошибка алгоритма",
                    JOptionPane.ERROR_MESSAGE);
            return null;
        }
        int[][] res=floyd_warshell.result();
        floyd_warshell=null;
        for(int i=0;i<countV;i++)

if(points.containsKey(i))points.get(i).color=CIRCLE_BORDERLINE_COLOR_BASE;
        return res;
    }   //алгоритм сразу(кнопка)

    public void addFromKlav(int [][] adjMatr){
        graph=new SimpleGraph();
        points=new HashMap<>();
        edges=new HashMap<>();
        countV=0;
        countE=0;
        for(int i=0;i<adjMatr.length;i++) addV();

        for(int i=0;i<adjMatr.length;i++){
            for(int j=0;j<adjMatr.length;j++){
                if(i==j)continue;
                if(adjMatr[i][j]!=0)
                    addE(new Graph.Edge(i,j,adjMatr[i][j]));
            }

        }
    }//сюда матрицу(кнопка ввести с клавиатуры)

    @Override
    public void paint(Graphics g) {

        g.setColor( GRAPH_FIELD_BACKGROUND );
        g.fillRect(0,0,600,500);

        drawGraph(g);

        g.setColor( GRAPH_FIELD_BORDER );                // Цвет рамки
        ((Graphics2D)g).setStroke(new BasicStroke(4));  // Толщина рамки
        g.drawRect( 0, 0,
```

```
            SIZE_OF_GRAPH_FIELD.width,
            SIZE_OF_GRAPH_FIELD.height);        // Нарисовать рамку



}

public void addV(){
    if (floyd_warshell != null) {
        JOptionPane.showMessageDialog(this,
                "Ошибка: алгоритм запущен",
                "Ошибка добавления вершины",
                JOptionPane.ERROR_MESSAGE);
        return;
    }
    int i=0;
    for(;i<=countV;i++)
        if(!points.containsKey(i))break;
    points.put(i, new VertexViz(this, i));
    add(points.get(i));
    graph.addV(i);
    if(i==countV) countV++;
    repaint();
}//кнопка добавить вершину

public void addE(Graph.Edge edge){
    if (floyd_warshell != null) {
        JOptionPane.showMessageDialog(this,
                "Ошибка: алгоритм запущен",
                "Ошибка добавления ребра",
                JOptionPane.ERROR_MESSAGE);
        return;
    }
    try{
        graph.addE(edge);
    }
    catch (RuntimeException exception) {
        JOptionPane.showMessageDialog(this,
                "Ошибка: " + exception.getLocalizedMessage(),
                "Ошибка добавления ребра",
                JOptionPane.ERROR_MESSAGE);
```

```
            return;
        }
        edges.put(countE,edge);
        countE++;
        repaint();
    }//добавить ребро

    public void remV(int v) {
        if (floyd_warshell != null) {
            JOptionPane.showMessageDialog(this,
                    "Ошибка: алгоритм запущен",
                    "Ошибка удаления вершины",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        if(v==countV-1)countV--;
        graph.removeV(new Graph.Vertex(v));
        for (int i = 0; i < countE; i++){
            if(edges.containsKey(i)) if (edges.get(i).v1 == v || edges.get(i).v2
== v) edges.remove(i);
        }
        remove(points.get(v));
        points.remove(v);
        repaint();
    }//удалить вершину

    public void remE(Graph.Edge edge){
        if (floyd_warshell != null) {
            JOptionPane.showMessageDialog(this,
                    "Ошибка: алгоритм запущен",
                    "Ошибка удаления ребра",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        graph.removeE(edge);
        for(int i=0;i<countE;i++) if(edges.containsKey(i) &&
edges.get(i).v1==edge.v1 && edges.get(i).v2==edge.v2)edges.remove(i);
        repaint();
    }//удалить ребро

    public void fromFile(JSONObject file){
```

```
        points=new HashMap<>();
        edges=new HashMap<>();
        graph=new SimpleGraph();
        countV=0;
        countE=0;
        int vertxs=(int)(long)file.get("count_vert");
        for(int i=0;i<vertxs;i++)addV();

        JSONArray edges=(JSONArray)file.get("edges");
        Iterator iter= edges.iterator();
        while(iter.hasNext()){
            JSONObject jnode=(JSONObject)iter.next();
            addE(new Graph.Edge((int)(long)jnode.get("v1"),(int)
(long)jnode.get("v2"),(int)(long)jnode.get("weight")));
        }
    }

    private void drawVertex(Graphics g, int v,Color color) {
        drawCircle(g, points.get(v).point.x,  points.get(v).point.y, VERTEX_R,
color);
        drawInt(g, points.get(v).point.x, points.get(v).point.y, v);
    }

    // Пишет text в точку (x,y)
    private void drawInt(Graphics g, int x, int y, int text) {
        g.setColor(TEXT_COLOR);
        Font font = new Font("Default", Font.PLAIN, TEXT_SIZE);  //Шрифт

        g.setFont(font);

        FontMetrics fm = g.getFontMetrics(font);

        g.drawString(Integer.toString(text),
                x-fm.stringWidth(Integer.toString(text))/2,
                y+fm.getAscent()/2);
    }

    private void drawCircle(Graphics g, int cX, int cY, int rad,Color color) {
        g.fillOval(cX-rad, cY-rad, rad*2, rad*2);

        ((Graphics2D)g).setStroke(new BasicStroke(2));
```

```
        g.setColor( color );
        g.drawOval(cX-rad, cY-rad, rad*2, rad*2);
    }

    private void drawEdge(Graphics g, Graph.Edge edge, Color color){

        Point v1 = new Point(points.get(edge.v1).point.x,
points.get(edge.v1).point.y);
        Point v2 = new Point(points.get(edge.v2).point.x,
points.get(edge.v2).point.y);

        ((Graphics2D)g).setStroke( EDGE_LINE_THIKNESS );  // Устанавливаем
толщину ребра

        g.setColor( EDGE_LINE_COLOR );
        g.drawLine(v1.x, v1.y, v2.x, v2.y);


        int x = (v1.x+v2.x)/2;
        int y = (v1.y+v2.y)/2;

        if(!graph.childrenV(edge.v2).way.containsKey(edge.v1))
            drawArrow(g,new Point((v1.x+v2.x)/2,(v1.y+v2.y)/2),v1);
        g.setColor(color);
        g.fillOval(x-14, y-14, EDJE_CIRKLE_R, EDJE_CIRKLE_R);

        ((Graphics2D)g).setStroke( EDGE_CIRKLE_LINE_THKNESS);
        g.setColor( EDGE_CIRKLE_LINE_COLOR );
        g.drawOval(x-14, y-14,  EDJE_CIRKLE_R, EDJE_CIRKLE_R);


        drawInt(g, x, y, edge.weight);
    }

    private void drawArrow(Graphics g,Point v1,Point v2){
        g.setColor(new Color(83, 83, 117));
        double dx=v1.x-v2.x;
        double dy=v1.y-v2.y;

        double t=Math.atan2(dy,dx);
        double p=Math.toRadians(40);
```

```
        int b=30;

        double x,y,r=t+p;
        for(int i=0;i<2;i++){
            x=v1.x-b*Math.cos(r);
            y=v1.y-b*Math.sin(r);
            g.drawLine(v1.x,v1.y,(int)x,(int)y);
            r=t-p;
        }
    }

    private void drawGraph(Graphics g){
        for(int i=0;i<countE;i++){
            if(edges.containsKey(i))
drawEdge(g,edges.get(i),edges.get(i).color);
        }
        for(int i=0;i<countV;i++){
            g.setColor(BASE_VERTEX_COLOR);
            if(points.containsKey(i)) drawVertex(g,i,points.get(i).color);
        }



    }
}
```

KeyIn.java:

```
package Window;

import javax.swing.*;

import static Window.Windows_par.*;



public class KeyIn extends JFrame {
    private static KeyInAlgo in;
    private static int[][] ar;
   // public void KeyIn(int ar[][]) {new KeyIn(); }
    public KeyIn(){
        //setDefaultCloseOperation(EXIT_ON_CLOSE);
        setTitle("Алгоритм");      //Имя окна
```

```
        setSize( WINDOW_SIZE );                                  //Размер окна
        in=new KeyInAlgo( this);
        setResizable(true);
        add(in);
        setVisible(true);
       int[][] ar_c = in.getArr();
        int i,j;
        ar= new int[ar_c.length][];
        for(i=0;i<ar_c.length;i++)
            ar[i] = new int[ar_c[i].length];
        for(i=0;i<ar_c.length;i++)
            for (j = 0; j < ar_c[i].length; j++)
                ar[i][j]=ar_c[i][j];
        System.out.println('o');
        for(i=0;i<ar.length;i++) {
            for (j = 0; j < ar[i].length; j++)
                System.out.print(ar[i][j]);
            System.out.println();
        }

    }
    public static int[][] getArr() {
        return ar;
    }

}
```

KeyInAlgo.java:

```
package Window;

import Graph.*;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import Window.*;
import Graph.*;
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.image.ImageObserver;
import java.text.AttributedCharacterIterator;
import java.util.HashMap;
import Window.KeyIn;
import static Window.Windows_par.*;
import static Window.Windows_par.BACKGROUND;
import static Window.Windows_par.SIZE_OF_GRAPH_FIELD;

import static Window.Windows_par.*;

public class KeyInAlgo extends JPanel {
    private JButton B1 = new JButton("OK");
    private JButton B2 = new JButton("OK");
    private JComboBox number = new JComboBox();
    private JLabel Dia[];
    JTextField in[][];
    static int[][] arr;
    private int n=2,i;
    JFrame Parent;


    public KeyInAlgo(JFrame Parent) {
        setLayout(null);
        this.setBounds(200, 110, 600, 500);
        B1.setBounds(100, 10, 200, 100);
        B1.addActionListener(new Check_number());
        add(B1);
        number.setBounds(300,10,100,100);
        add(number);
        for(i=3;i<10;i++) {
            number.addItem(i);
        }
        this.Parent=Parent;
        //arr=ar;
    }
    class Check_number implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            int i,j;
            setLayout(null);
```

```
        //System.out.println("sdada");
        n= Integer.parseInt(number.getSelectedItem().toString());
       /* System.out.println("\n\n");
        System.out.println(n);
        System.out.println("\n\n");*/
        removeAll();
        in=new JTextField[n][];
        arr= new int[n][];
        Dia=new JLabel[n];
        for(i=0;i<n;i++) {
            arr[i] = new int[n];
            in[i] = new JTextField[n];
            Dia[i]=new JLabel("0");
        }
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
            {
                arr[i][j]=100*i+j;
               // System.out.println(arr[i][j]+"Lol"+i+" "+j);
                in[i][j] = new JTextField("0",5);
                if(i!=j) {
                    in[i][j].setBounds(i * 40, j * 40, 40, 40);
                    in[i][j].setVisible(true);
                    add(in[i][j]);
                }
                else
                {
                    Dia[i].setBounds(i * 40, j * 40, 40, 40);
                    Dia[i].setVisible(true);
                    add(Dia[i]);
                }
            }
        repaint();
        B2.setBounds(n*40, 0, 100, 500);
        B2.addActionListener(new Out_matr());
        add(B2);

    }
}
class Out_matr implements ActionListener {
    public void actionPerformed(ActionEvent e) {
```

```
            int i,j;
            /*for(i=0;i<n;i++) {
                for (j = 0; j < n; j++)
                    System.out.print(arr[i][j]);
                System.out.println();
            }*/
            for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                {
                    arr[i][j]=Integer.parseInt(in[i][j].getText());
                }
            /*for(i=0;i<n;i++) {
                for (j = 0; j < n; j++)
                    System.out.print(arr[i][j]+" ");
                System.out.println();
            }*/
            Parent.setVisible(false);
            //Parent.dispose();
        }
        }

    public static int[][] getArr() {
        return arr;
    }
}

Out_mat.java:
package Window;

import javax.swing.*;

import static Window.Windows_par.*;



public class Out_mat extends JFrame {
    private static Out_matAlgo in;
    public Out_mat(int[][]ar){
        //setDefaultCloseOperation(EXIT_ON_CLOSE);
        setTitle("Результат");      //Имя окна
        setSize( WINDOW_SIZE );
```

```
        in= new Out_matAlgo( this,ar);//Размер окна
        setResizable(true);
        add(in);
        setVisible(true);

    }

}
```

Out_matAlgo.java:

```
package Window;

import Graph.*;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import Window.*;
import Graph.*;

import javax.swing.*;
import java.awt.*;
import java.awt.image.ImageObserver;
import java.text.AttributedCharacterIterator;
import java.util.HashMap;
import Window.KeyIn;
import static Window.Windows_par.*;
import static Window.Windows_par.BACKGROUND;
import static Window.Windows_par.SIZE_OF_GRAPH_FIELD;

import static Window.Windows_par.*;

public class Out_matAlgo extends JPanel {
    private JButton B1 = new JButton("OK");
    private JTextField number = new JTextField("", 5);
    private JLabel Dia[];
    JLabel out[][];
    static int[][] arr;
```

```
    JFrame Parent;


    public Out_matAlgo(JFrame Parent,int[][] ar) {
        setLayout(null);
        this.setBounds(200, 110, 600, 500);
        this.Parent = Parent;
        int i,j;
        out=new JLabel[ar.length][];
        for(i=0;i< ar.length;i++)
            out[i] = new JLabel[ar[i].length];
        for(i=0;i<ar.length;i++)
            for(j=0;j< ar[i].length;j++)
            {
                if(ar[i][j]>=10_000_000)
                    out[i][j] = new JLabel("INF");
                else
                out[i][j] = new JLabel(String.valueOf(ar[i][j]));
                    out[i][j].setBounds(i * 40, j * 40, 40, 40);
                    out[i][j].setVisible(true);
                    add(out[i][j]);

            }
        B1.setBounds(ar.length*40, 0, 100, 500);
        B1.addActionListener(new Out_mat_ev());
        add(B1);
    }

    class Out_mat_ev implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Parent.dispose();
        }
    }

}
```

Quation_for_delete.java:

```
package Window;

import javax.swing.*;
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import static Window.Windows_par.*;

import Graph.*;



public class Quation_for_delete extends JFrame {
    private JButton B1 = new JButton("OK");
    private JButton B2 = new JButton("OK");
    JComboBox numb = new JComboBox();
    private JTextField wei= new JTextField("0",5);
    private JComboBox number = new JComboBox();
    int in[][] = {
            {0, 0, 1, 0, 0},
            {2, 0, 0, 0, 0},
            {1, 2, 0, 0, 0},
            {0, 0, 3, 0, 0},
            {1, 0, 5, 6, 0},
    };
    private int a, b, i, j, counter, flag, flag2,c;
    GraphPlane graph;

    public Quation_for_delete(GraphPlane graph, int flag, int flag2) {
        //setDefaultCloseOperation(EXIT_ON_CLOSE);
        setTitle("Кого?");      //Имя окна
        setSize(WINDOW_SIZE);
        this.graph = graph;
        this.flag = flag;
        this.flag2 = flag2;
        setLayout(null);
        setVisible(true);
        char[] a = {'0', '1', '2', '3'};
        // JComboBox number = new JComboBox();
        //number.setEditable(true);
        //System.out.println("lol" + graph.points.keySet());
                            //тип тут копирую.
```

```
    in=graph.get_matr();
    ArrayList<Integer> vert =graph.graph.getVertexes();
    if(flag2==3)
        Doner();
    if(flag2==4)
        for (i=0;i< vert.size();i++) {
            number.addItem(vert.get(i));
            System.out.println(vert.get(i));
        }
    else
    for (i = 0; i < in.length; i++) {
        counter = 0;
        for (j = 0; j < in[i].length; j++)
            if (in[i][j] > 0||flag2==1)
                counter++;
        if (counter > 0)
            number.addItem(i);
    }
    if (flag2==1)
        add(wei);
    wei.setBounds(200, 0, 100, 100);
    this.setBounds(100, 110, 600, 500);
    B1.setBounds(0, 100, 100, 40);
    B1.addActionListener(new Check_number());
    add(B1);
    if (flag==2){
        add (numb);
        numb.setBounds(100, 0, 100, 100);
    }
    add(number);
    number.setBounds(0, 0, 100, 100);
    number.addActionListener(actionListener);
    //add(number);
    //if(flag==2)

    // number.actionPerformed(new Check_number1());



}
class Check_number implements ActionListener {
```

```
    public void actionPerformed(ActionEvent e) {
        a = Integer.parseInt(number.getSelectedItem().toString());
        if(flag==2)
            b= Integer.parseInt(numb.getSelectedItem().toString());
        if(flag2==1)
            c=Integer.parseInt(wei.getText());
        Doner();
    }
}

ActionListener actionListener = new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int selectedIndex = number.getSelectedIndex();
        System.out.println(selectedIndex);
        numb.removeAllItems();
        //JComboBox number2=new JComboBox();
        //numb.setEditable(true);
        numb.setVisible(true);
        getContentPane().add(numb);
        numb.setLayout(null);
        // numb.addItem(1);
        a = Integer.parseInt(number.getSelectedItem().toString());
        for (i = 0; i < in[a].length; i++)
            if(selectedIndex!=i)
                if ((in[selectedIndex][i] > 0)!=(flag2!=2)) {
                    numb.addItem(i);
                    System.out.println(i);
            }
        numb.setBounds(100, 0, 100, 100);
        numb.setVisible(true);
    }
};
void Doner(){
    if (flag2==1)
        graph.addE(new Graph.Edge(a,b,1));
    if (flag2==2)
        graph.remE(new Graph.Edge(a,b,1));
    if (flag2==3)
        graph.addV();
    if (flag2==4)
        graph.remV(a);
```

```
        dispose();
    }
}
```

VertexViz.java:

```
package Window;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.util.Random;

import static Window.Windows_par.*;

public class VertexViz extends JPanel implements MouseListener,
MouseMotionListener {

    JPanel parent;
    Point point;
    final int v;
    Color color_base=new Color(81, 201, 66);
    Color color_alg_i=new Color(201, 19, 25);
    Color color_alg_v=new Color(14, 19, 201);
    Color color=new Color(81, 201, 66);

    private Point mouse = new Point();

    private boolean flagCanMove = false;

    VertexViz(JPanel parent, int v ) {
        this.parent = parent;
        this.v = v;

        Random random = new Random();
        point = new Point(random.nextInt(600- VERTEX_D)+ VERTEX_R,
                random.nextInt(500- VERTEX_D)+ VERTEX_R);

        setSize(new Dimension( VERTEX_D, VERTEX_D));
        setLocation(point.x- VERTEX_R, point.y- VERTEX_R);
```

```
        addMouseMotionListener(this);
        addMouseListener(this);
    }
    // Движение вершины
    @Override
    public void mousePressed(MouseEvent e) {
        flagCanMove = true;
        mouse.x = e.getX();
        mouse.y = e.getY();
    }
    @Override
    public void mouseReleased(MouseEvent e) {
        flagCanMove = false;
    }
    @Override
    public void mouseDragged(MouseEvent e) {
        if (flagCanMove) {
            int dx = e.getX() - mouse.x;
            int dy = e.getY() - mouse.y;

            int x = point.x + dx;
            if (x- VERTEX_R < 0) point.x = VERTEX_R;
            else if (x+ VERTEX_R>600) point.x = 600- VERTEX_R;
            else point.x = x;

            int y = point.y + dy;
            if (y- VERTEX_R < 0) point.y = VERTEX_R;
            else if (y+ VERTEX_R>500) point.y = 500- VERTEX_R;
            else point.y = y;

            setLocation(point.x- VERTEX_R, point.y- VERTEX_R);
            parent.repaint();
        }
    }
}
```

Windows_par.java:

```
package Window;

import java.awt.*;
```

```
public class Windows_par {
    // Размеры полей
    final static Dimension WINDOW_SIZE = new Dimension(941, 550);
    final static Dimension CREATE_GRAPH_PANEL_SIZE = new Dimension(300,200);
    final static Dimension SIZE_OF_INPUT_FIELD = new Dimension(25,19);
    public final static Dimension SIZE_OF_GRAPH_FIELD = new Dimension(600,500);
    // Цвета
    final static Color TEXT_COLOR = Color.BLACK;
    final static Color BUTTENS_BORDER = new Color(59, 157, 165);
    public final static Color BACKGROUND = new Color(225, 219, 180);
    final static Color BASE_EDGE_COLOR = new Color(193, 241, 236);
    final static Color EDGE_LINE_COLOR = new Color(15, 15, 21);
    final static Color EDGE_CIRKLE_LINE_COLOR = new Color(0, 0, 0);
    final static Color CREATE_BUTTONS_BG = new Color(127, 186, 189);
    final static Color RESULT_EDGE_COLOR = new Color(203, 138, 82);
    final static Color BASE_VERTEX_COLOR = new Color(81, 201, 66);
    final static Color CIRCLE_BORDERLINE_COLOR_BASE = new Color(81, 201, 66);
    final static Color CIRCLE_BORDERLINE_COLOR_IN_ALG = new Color(201, 19, 25);
    final static Color GRAPH_FIELD_BORDER = new Color(93, 91, 71);
    final static Color RESULT_VERTEX_COLOR = new Color(163, 109, 53);
    public final static Color GRAPH_FIELD_BACKGROUND = new Color(251, 255, 253);
    // Целочисленные константы
    final static int VERTEX_R = 18;
    final static int TEXT_SIZE = 14;
    final static int EDJE_CIRKLE_R = 28;
    final static int VERTEX_D = VERTEX_R*2;

    // Размеры линий
    final static BasicStroke EDGE_LINE_THIKNESS = new BasicStroke(1);
    final static BasicStroke EDGE_CIRKLE_LINE_THKNESS = new BasicStroke(2);
}
```

GUI.java:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import Window.*;
import Graph.*;

import javax.swing.*;
```

```
import java.awt.*;
import java.awt.image.ImageObserver;
import java.text.AttributedCharacterIterator;
import java.util.HashMap;
import Window.KeyIn;
import static Window.Windows_par.*;
import static Window.Windows_par.BACKGROUND;
import static Window.Windows_par.SIZE_OF_GRAPH_FIELD;


public class GUI extends JFrame {
    private JButton in_from_file = new JButton("Ввод с файла");
    private JButton Matr_out = new JButton("Вон матрицу");
    private JButton to_start = new JButton("В начало");
    private JButton prev_step = new JButton("Прошлый шаг");
    private JButton next_step = new JButton("Следующий шаг");
    private JButton to_finish = new JButton("В конец");
    private JButton in_from_key = new JButton("Ввод с клавиатуры");
    private JButton in_from_graphic = new JButton("Ввод графически");
    private JButton start_algorithm = new JButton("Запустить алгоритм");
    private JButton NewE= new JButton("NewE");
    private JButton DelE = new JButton("DelE");
    private JButton NewV = new JButton("NewV");
    private JButton DelV = new JButton("DelV");
    private JButton BB = new JButton("Назад");
    private JButton B1 = new JButton("Назад");
    private GraphPlane graph = new GraphPlane();
    JPanel down_panel = new JPanel();
    JPanel up_panel = new JPanel();
    JPanel create_panel = new JPanel();
    private JLabel important_to_add=new JLabel("Developed By KKB");
    private int ar[][];

    public GUI() {
        super("MainWindow");
        this.setBounds(250, 150, 1000, 800); //само окно
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //закрытие окна

        setPreferredSize(SIZE_OF_GRAPH_FIELD);    //Размер рамки
        setBackground(BACKGROUND);
        up_panel.setLayout(null);
```

```
        down_panel.setLayout(null);
        create_panel.setLayout(null);
        important_to_add.setBounds(300,610,200,15);
        add(important_to_add);

        in_from_file.addActionListener(new InFromFileEvent());//вызов чего то
при нажатии
        up_panel.add(in_from_file);
        in_from_file.setBounds(100, 10, 200, 100);

        Matr_out.addActionListener(new Matr_out_event());//вызов чего то при
нажатии
        add(Matr_out);
        Matr_out.setBounds(100, 610, 200, 100);

        in_from_key.addActionListener(new InFromKeyEvent());
        in_from_key.setBounds(300, 10, 200, 100);
        up_panel.add(in_from_key);

        in_from_graphic.addActionListener(new InFromGraphicEvent());
        in_from_graphic.setBounds(500, 10, 200, 100);
        up_panel.add(in_from_graphic);

        start_algorithm.addActionListener(new Algorithm());
        start_algorithm.setBounds(700, 10, 200, 100);
        up_panel.add(start_algorithm);

        graph.setBackground(GRAPH_FIELD_BACKGROUND);
        graph.setBounds(200,110,600,500);
        getContentPane().add(graph);
        repaint();

        to_start.addActionListener(new Step_to_start());
        to_start.setBounds(100, 10, 200, 100);
        down_panel.add(to_start);

        prev_step.addActionListener(new Step_back());
        prev_step.setBounds(300, 10, 200, 100);
        down_panel.add(prev_step);

        next_step.addActionListener(new Step_forward());
```

```
        next_step.setBounds(500, 10, 200, 100);
        down_panel.add(next_step);

        to_finish.addActionListener(new Step_to_finish());
        to_finish.setBounds(700, 10, 200, 100);
        down_panel.add(to_finish);

        NewE.addActionListener(new NewE_ev());
        NewE.setBounds(100, 10, 200, 100);
        create_panel.add(NewE);

        DelE.addActionListener(new DelE_ev());
        DelE.setBounds(300, 10, 200, 100);
        create_panel.add(DelE);

        NewV.addActionListener(new NewV_ev());
        NewV.setBounds(500, 10, 200, 100);
        create_panel.add(NewV);

        DelV.addActionListener(new DelV_ev());
        DelV.setBounds(700, 10, 200, 100);
        create_panel.add(DelV);

        BB.addActionListener(new Back_to_main());
        BB.setBounds(700, 610, 200, 100);
        create_panel.add(BB);
        B1.addActionListener(new Back_to_main());
        B1.setBounds(700, 610, 200, 100);
        down_panel.add(B1);

        getContentPane().add(down_panel);
        getContentPane().add(create_panel);
        down_panel.setVisible(false);
        create_panel.setVisible(false);
        getContentPane().add(up_panel); //форма
        //pack();

    }

    class InFromFileEvent implements ActionListener {
        public void actionPerformed(ActionEvent e) {
```

```
        Filer in =new Filer(graph);
    }
}

class Matr_out_event implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        int i,j;
        System.out.println('K');
        for(i=0;i<ar.length;i++) {
            for (j = 0; j < ar[i].length; j++)
                System.out.print(ar[i][j]);
            System.out.println();
        }
        Out_mat ee=new Out_mat(ar);
        add(ee);
    }
}

class InFromKeyEvent implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        KeyIn hm=new KeyIn();
        int i,j;
        graph.addFromKlav(hm.getArr());
        int ar_c[][] = hm.getArr();
        System.out.println("SUKA");
        for(i=0;i<ar_c.length;i++) {
            for (j = 0; j < ar_c[i].length; j++)
                System.out.print(ar_c[i][j]);
            System.out.println();
        }
        ar= new int[ar_c.length][];
        for(i=0;i<ar_c.length;i++)
            ar[i] = new int[ar_c[i].length];
        for(i=0;i<ar_c.length;i++)
            for (j = 0; j < ar_c[i].length; j++)
                ar[i][j]=ar_c[i][j];
        System.out.println('L');
        for(i=0;i<ar.length;i++) {
            for (j = 0; j < ar[i].length; j++)
                System.out.print(ar[i][j]);
            System.out.println();
```

```
        }
        add(hm);
        repaint();

    }
}

class InFromGraphicEvent implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        up_panel.setVisible(false);
        create_panel.setVisible(true);
        getContentPane().add(create_panel);
        repaint();
    }
}

class Algorithm implements ActionListener {
    public void actionPerformed(ActionEvent e) {
       graph.start_alg();
        up_panel.setVisible(false);
        down_panel.setVisible(true);
        getContentPane().add(down_panel);
        repaint();
    }
}

class Step_to_start implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        graph.go_to_start();
    }
}

class Step_back implements ActionListener {
    public void actionPerformed(ActionEvent e) {
       graph.step_back();
    }
}

class Step_forward implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        graph.step();
```

```
    }
}

class Step_to_finish implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        ar=graph.result();
       add(new Out_mat(ar));
    }
}
class Back_to_main implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        create_panel.setVisible(false);
        down_panel.setVisible(false);
        up_panel.setVisible(true);
        getContentPane().add(up_panel);
        repaint();
    }
}
class NewE_ev implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        Integer first=0,second=1;
        Quation_for_delete a =new Quation_for_delete( graph,2,1);
        //add(a);
        System.out.print(first);
        graph.addE(new Graph.Edge(first,second,1));
    }
}
class DelE_ev implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        Integer first=0,second=0;
        Quation_for_delete a =new Quation_for_delete( graph,2,2);
        System.out.print(first);
        //graph.remE(new Graph.Edge(first,second));
        add(a);
    }
}
class NewV_ev implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        graph.addV();
    }
}
```

```
    class DelV_ev implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Quation_for_delete a =new Quation_for_delete( graph,1,4);
        //repaint();
        }
    }
}
```

Graph.java:

```
package Graph;

import java.awt.*;
import java.util.ArrayList;
import java.util.HashMap;

/**
 * Абстрактный граф
 */
public abstract class Graph {

    int kolV = 0;
    int kolE = 0;

    abstract public boolean addV(int v);

    abstract public boolean addE(Edge e);

    abstract public boolean removeV(Vertex v);

    abstract public boolean removeE(Edge e);

    abstract public Vertex childrenV(int v);

    abstract public Vertex checkV(int v);

    abstract public Edge checkE(int v1, int v2);

    abstract public int countChildren(int v);

    abstract public boolean connectivity();
```

```
    public int getKolE() {
        return kolE;
    }
    public int getKolV() {
        return kolV;
    }




    abstract public void clear();

    abstract public ArrayList<Integer> getVertexes();


    public static class Edge {
        public Edge(int v1, int v2, int weight) {
            this.v1 = v1;
            this.v2 = v2;
            this.weight = weight;
        }

        public int v1;
        public int v2;
        public int weight;
        public Color color=new Color(193, 241, 236);
    }
    public static class Vertex {
        public int v;
        public HashMap<Integer,Integer> way = new HashMap<Integer,Integer>();

        public Vertex(int v) {
            this.v = v;
        }
    }

}
```

SimpleGraph.java:

```
package Graph;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class SimpleGraph extends Graph{
    HashMap<Integer,Vertex> adjacenyList=new HashMap<>();



    @Override
    public boolean addV(int v) {
        if (adjacenyList.containsKey(v)) throw new RuntimeException("Такая
вершина уже есть");

        adjacenyList.put(v, new Vertex(v));
        if(v==kolV)kolV++;
        return true;
    }

    @Override
    public boolean addE(Edge e) {

        if ( !adjacenyList.containsKey(e.v1) ) throw new
RuntimeException("Вершина "+e.v1+" не существует");
        if ( !adjacenyList.containsKey(e.v2) ) throw new
RuntimeException("Вершина "+e.v2+" не существует");
        if ( adjacenyList.get(e.v1).way.containsKey(e.v2)) throw new
RuntimeException("Данное ребро уже существует");

        adjacenyList.get(e.v1).way.put(e.v2, e.weight);
      //  adjacenyList.get(e.v2).way.put(e.v1, e.weight);

        kolE++;
        return true;
    }

    @Override
    public boolean removeV(Vertex v) {
        if (!adjacenyList.containsKey(v.v)) throw new RuntimeException("Такой
вершины нет");
        for(int i=0;i<kolV;i++){
            if(i==v.v)continue;
```

```
            if(adjacenyList.containsKey(i)&&
adjacenyList.get(i).way.containsKey(v.v)) adjacenyList.get(i).way.remove(v.v);
        }
        adjacenyList.remove(v.v);
        if(v.v==kolV-1)kolV--;
        return true;
    }

    @Override
    public boolean removeE(Edge e) {
        if ( !adjacenyList.containsKey(e.v1) ) throw new
RuntimeException("Вершина "+e.v1+" не существует");
        if ( !adjacenyList.containsKey(e.v2) ) throw new
RuntimeException("Вершина "+e.v2+" не существует");
        if ( !adjacenyList.get(e.v1).way.containsKey(e.v2)) throw new
RuntimeException("Данное ребро не существует");

        adjacenyList.get(e.v1).way.remove(e.v2);
        //adjacenyList.get(e.v2).way.remove(e.v1);

        kolE--;
        return true;
    }

    @Override
    public Vertex childrenV(int v) {
        return adjacenyList.get(v);
    }

    @Override
    public Edge checkE(int v1, int v2) {
        if (childrenV(v1)!=null && childrenV(v2)!=null) {
            Integer i = adjacenyList.get(v1).way.get(v2);

            return i==null ? null : new Edge(v1,v2,i.intValue());
        }
        return null;
    }

    @Override
```

```
    public Vertex checkV(int v) {
        return adjacenyList.get(v);
    }

    @Override
    public int countChildren(int v) {
        if (!adjacenyList.containsKey(v)) return -1;
        return adjacenyList.get(v).way.size();
    }

    @Override
    public boolean connectivity() {
        ArrayList<Integer> stek = new ArrayList<Integer>();
        ArrayList<Integer> baseV = this.getVertexes();
        stek.add(baseV.get(0));
        int n = 0;
        int k = 1;

        while (n < k){
            Graph.Vertex v_i = this.checkV(stek.get(n));
            for(Map.Entry<Integer,Integer> j: v_i.way.entrySet()) {
                if (!stek.contains(j.getKey().intValue())) {                // И
путь меньше уже найденного
                    stek.add(j.getKey().intValue());
                    k++;
                }
            }
            n++;
        }
        return k==adjacenyList.size();
    }

    @Override
    public void clear() {
        adjacenyList = new HashMap<Integer, Vertex>();
        kolE = 0;
        kolV = 0;
    }

    @Override
    public ArrayList<Integer> getVertexes() {
```

```
        ArrayList<Integer> ret = new ArrayList<Integer>();

        for(Map.Entry<Integer, Vertex> v: adjacenyList.entrySet()) {
            ret.add(v.getKey());
        }
        return ret;
    }
}
```