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

ОТЧЕТ

по учебной практике

Тема: “VK Clique”

Студент гр. 8303 _____ Быков А. В.

Студент гр. 8303 _____ Деркач Н. В.

Студент гр. 8303 _____ Логинов Е.А.

Руководитель _____ Фирсов М. А.

Санкт-Петербург

2020

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Быков А. В. группы 8303
Студент Логинов Е.А. группы 8303
Студент Деркач Н. В. группы 8303
Тема практики: “VK Clique”

Задание на практику:

Требуется визуализировать граф общих друзей в социальной сети vk.com для заданного пользователем списка участников социальной сети с помощью ввода краткого имени пользователя. Вершины графа должны однозначно идентифицировать пользователя социальной сети. У рёбер видно свойство - количество общих друзей. Нажатие на кнопку визуализирует минимальный подграф, объединяющий всех пользователей по рёбрам с наибольшим количеством общих друзей. В спецификации требуется описать свойства полученной модели.

Сроки прохождения практики: 29.06.2020 – 12.07.2020
Дата сдачи отчета: 11.07.2020
Дата защиты отчета: 11.07.2020

Студент гр. 8303	_____	Быков А. В.
Студент гр. 8303	_____	Деркач Н. В.
Студент гр. 8303	_____	Логинов Е.А.
Руководитель	_____	Фирсов М. А.

АННОТАЦИЯ

Целью данной работы является приобретение навыков работы с объектно-ориентированной парадигмой программирования. В процессе практики необходимо реализовать проект” VK Clique”, который строит граф общих друзей пользователей социальной сети VKontakte, где вершины представлены в виде самих пользователей, а рёбра, соединяющие их - количеством общих друзей.

Возможность построить такой граф выполнена с помощью чтения списка людей и информации о них из файла или из консоли. Также при разработке выполняется тестирование программы, для проверки корректности основного алгоритма.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Программа “VK Clique” строит граф пользователей социальной сети VKontakte, где вершины представлены в виде самих пользователей, а рёбра, соединяющие их - количеством общих друзей.

Возможность построить такой граф выполнена с помощью чтения списка людей и информации о них из файла. О каждом пользователе хранится информация, связанная с его личными данными (фамилия, имя, возраст, количество друзей), которая отображается по клику на вершину с фотографией.

По нажатию кнопки “Build Graph” программа строит и визуализирует граф, отображающий все введённые связи между пользователями (строит рёбра между вершинами с весами, равными количеству общих друзей). Если построить граф с поставленной галочкой “Spanning Tree”, при визуализации графа программа с помощью алгоритма Прима построит минимальное остовое дерево этого графа по признаку наибольшего числа общих друзей между пользователями, и выделит красным цветом рёбра, которые входят в это минимальное остовое дерево.

При нажатии кнопки “Build Graph” с поставленной галочкой “Consider Non Friends”, граф визуализируется, построив рёбра между всеми пользователями, но рёбра между людьми, которые не являются непосредственными друзьями, будут обозначены пунктирной линией. Соответственно, минимальное остовое дерево такого графа будет отличаться от дерева, построенного без учета пользователей, не знакомых друг с другом, но при этом имеющих много общих друзей.

Изображение графа масштабируется в зависимости от его размера, присутствует возможность двигать его для более удобного изучения.

1. СПЕЦИФИКАЯ ПРОГРАММЫ

1.1 Исходные требования к программе

1.1.1 Требования к входным данным

- Краткое имя пользователя (ID).

1.1.2 Требования к выходным данным

- Программа визуализирует граф общих друзей.

1.1.3 Требования к визуализации

- Программа должна обладать простым и понятным интерфейсом. При нажатии на вершину с человек появляется окно с фотографией и данными. Прототип интерфейса представлен на рисунках 1 и 2.

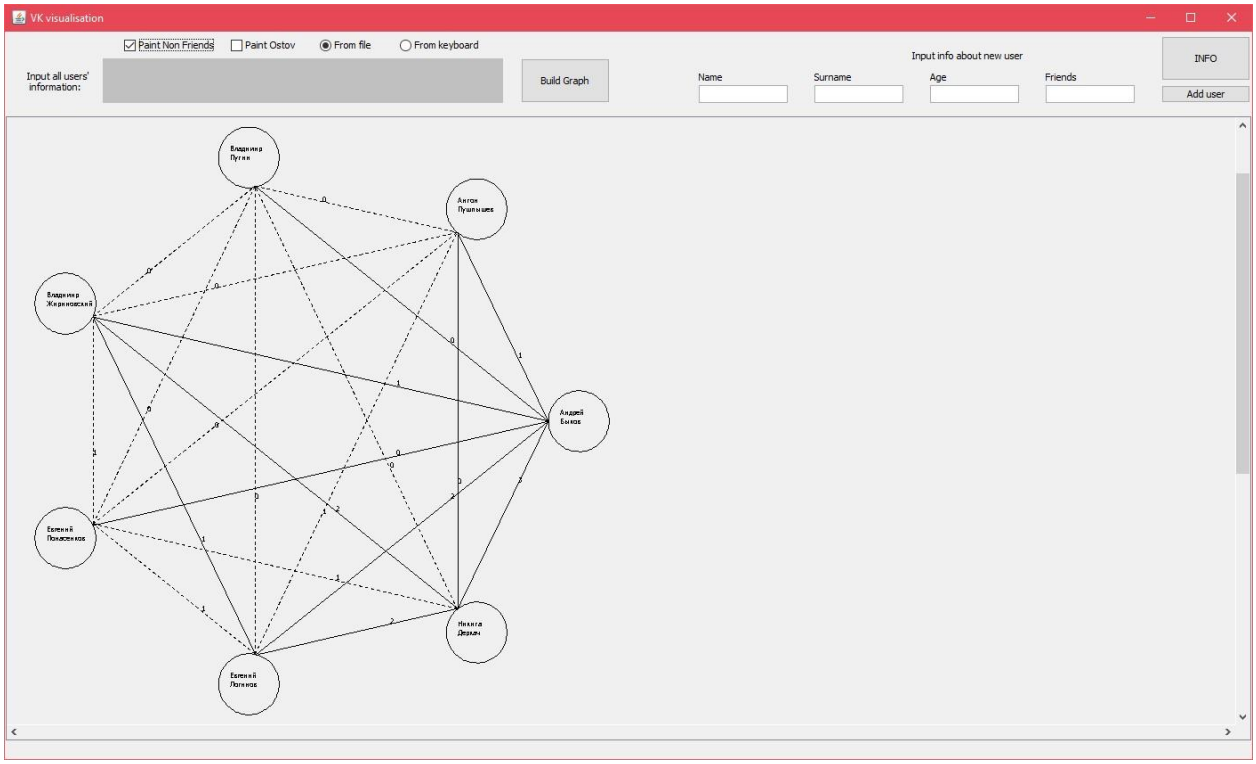


Рисунок 1 – прототип интерфейса.

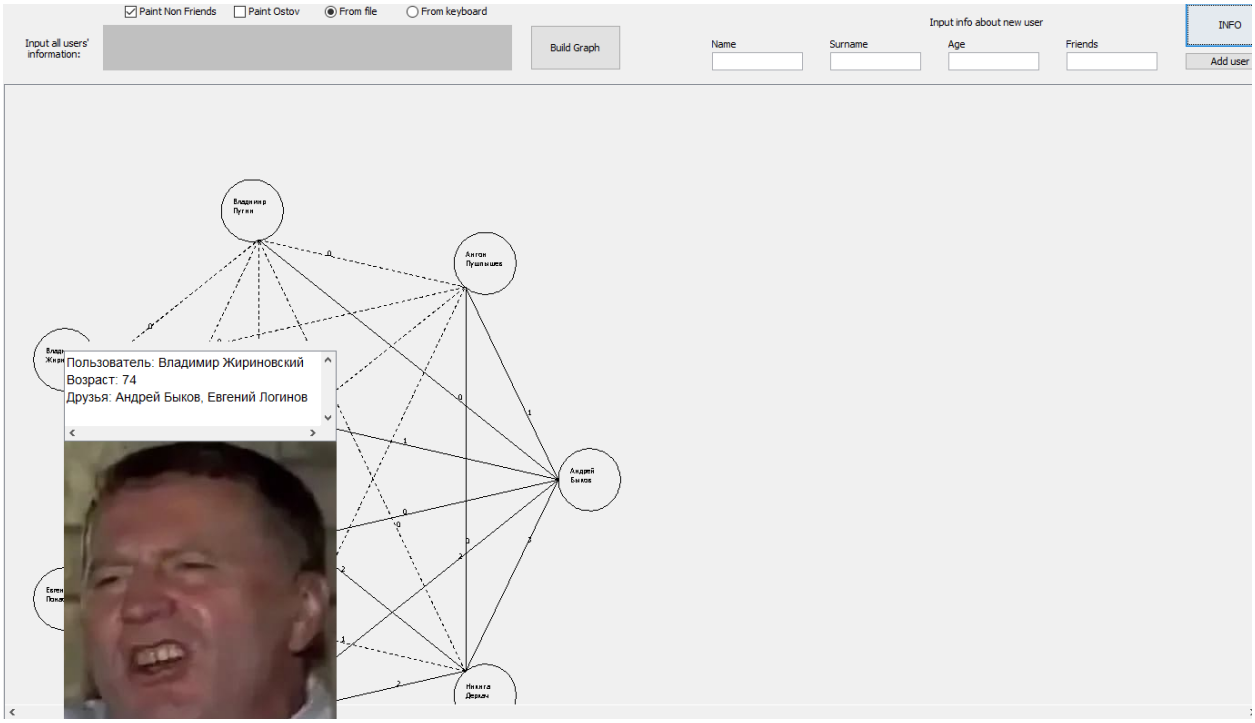


Рисунок 2 – окно с основной информацией пользователя при нажатии на вершину.

Приложение имеет интерактивную область с графом. В верхней панели располагаются следующие элементы управления:

- Поля для ввода информации: считывание информации возможно с клавиатуры или из файла.
- Кнопка “Build Graph” - построение графа общих друзей.
- Кнопка “Spanning Tree” предназначена для построения для построения минимального остового дерева по признаку наибольшего числа общих друзей, ребра минимального остового дерева рисуются красным цветом.
- Кнопка “Consider Non Friends” рисует пунктирные ребра между людьми, которые не являются непосредственными друзьями.
- Кнопка “INFO” выводит информацию о данном приложении.

1.2 Уточнение требований после сдачи первого этапа

1.2.1 Требования к вводу исходных данных

- Требования к вводу исходных данных не изменились с прошлой итерации.

1.2.2 Требования к выходным данным

- Добавить список друзей пользователя (Рисунок 3).

Рисунок 3 – добавлен список друзей пользователей в окно основной информации (появляется при нажатии на вершину с пользователем).

1.2.3 Требования к визуализации

- Требования в визуализации не изменились.

1.3 Уточнение требований после сдачи второго этапа

1.3.1 Требования к вводу исходных данных

- Требования к вводу исходных данных не изменились с прошлой итерации.

1.3.2 Требования к выходным данным

- Добавить возможность сохранять снимок графа при помощи кнопки “Save”.

1.3.3 Требования к визуализации

- Увеличить толщину ребер графа (Рисунок 4).
- Добавить цвета к ребрам и вершинам графа (Рисунок 4).
- Добавить панель слева со списком пользователей, при нажатии на ID пользователя появляется диалоговое окно с основной информацией и списком друзей (Рисунок 4).
- Добавить возможность сохранять снимок графа при помощи кнопки “Save”, изображение сохраняется в формате .png (правый нижний угол, рисунок 4, 5).

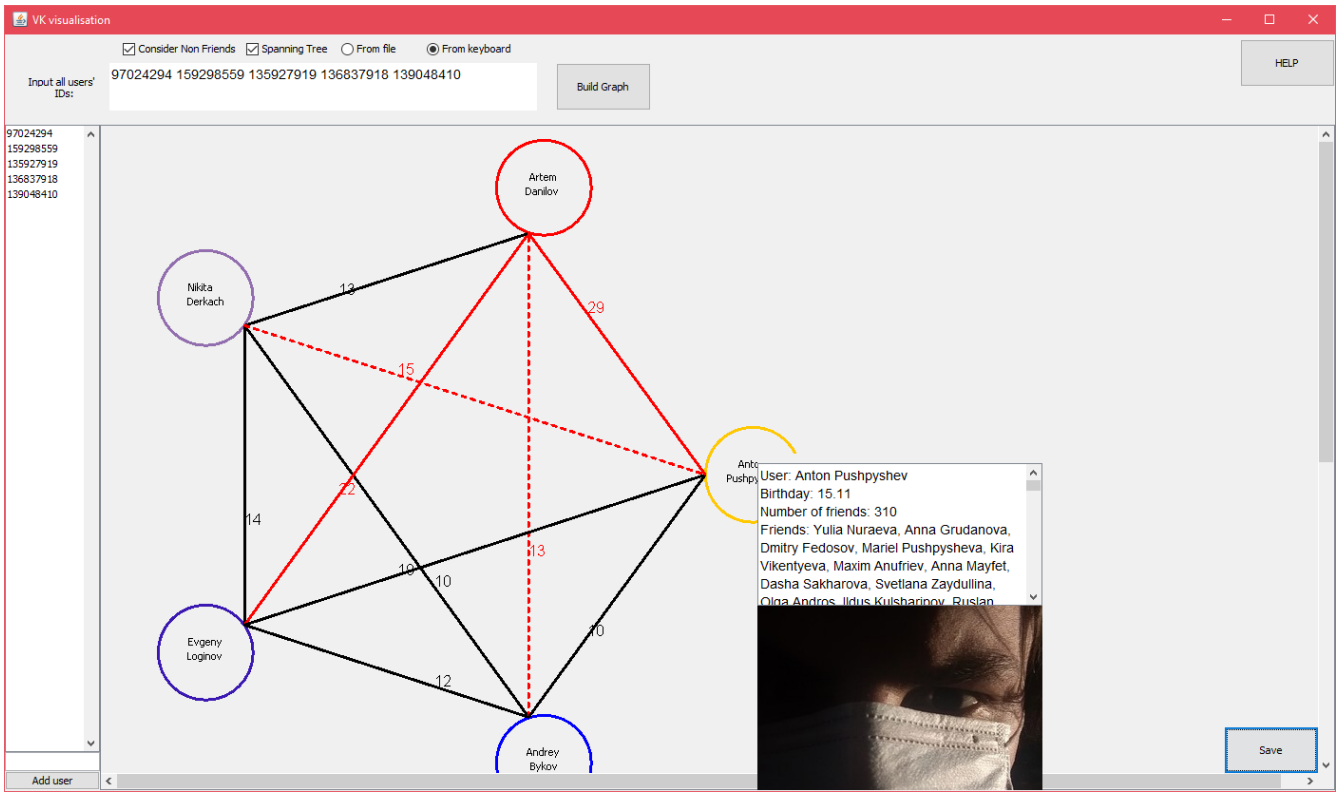


Рисунок 4 – изменения в интерфейсе после сдачи второй итерации.

В левой панели располагаются следующие элементы интерфейса:

- Поле для добавления нового пользователя, кнопка “Add user” и поле для ввода над ним (Рисунок 4, 5).
- Список id уже добавленных пользователей. При нажатии появляется диалоговое окно с основной информацией и количеством друзей (Рисунок 5).
- В правом нижнем расположена кнопка “Save”, позволяющая сделать скриншот графа общих друзей, изображение сохранится в формате .png.

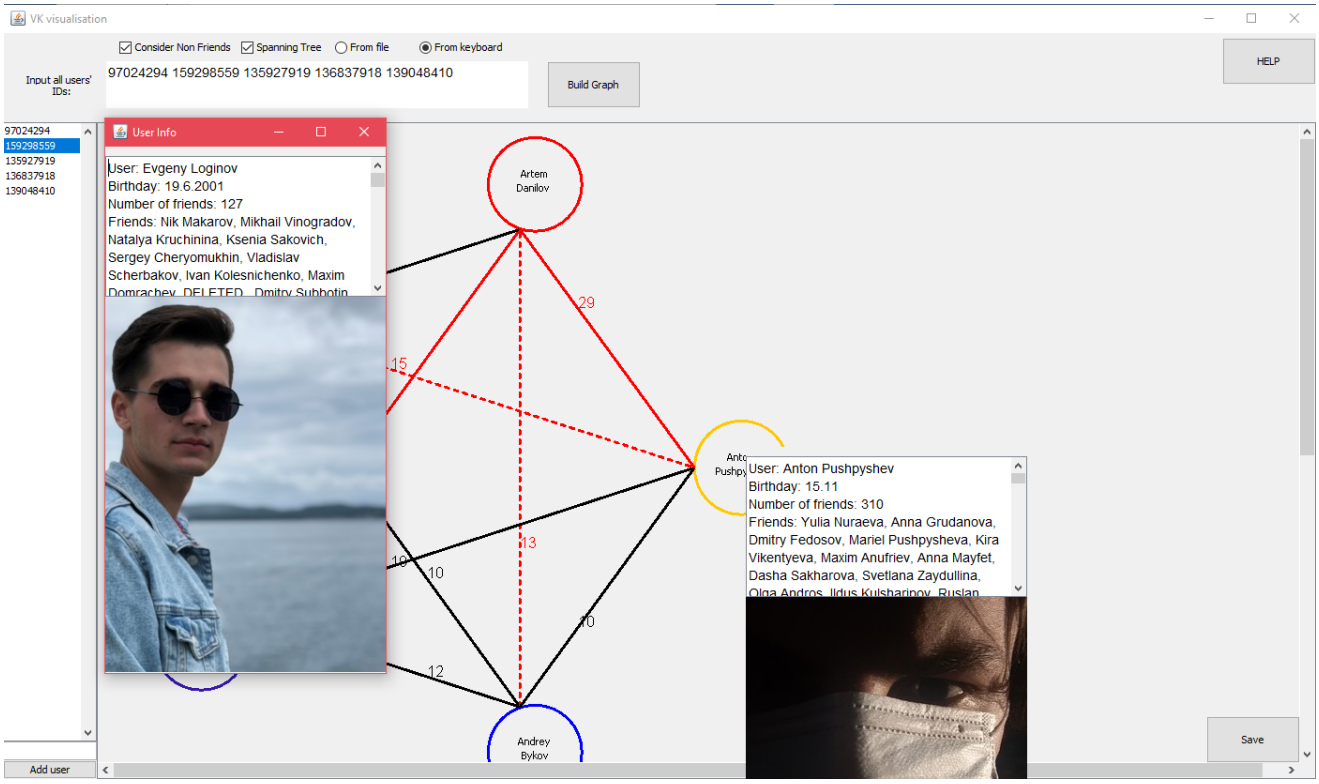


Рисунок 5 – возможность добавления нового пользователя.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1 План разработки

1) К 29 июня 2020 года выбрать тему проекта, обсудить задание, распределить роли между членами команды и создать примерный план разработки.
2) Ко 2 июля 2020 года выполнить первую итерацию, создав прототип интерфейса и отчет о проделанной работе.
3) К 3 июля, в случае необходимости, скорректировать интерфейс программы после сдачи первой итерации.
4) К 5 июля 2020 года реализовать структуры данных, необходимые для представления графа и для хранения информации о пользователе.
5) К 6 июля 2020 года реализовать работу приложения с социальной сетью “ВКотакте” при помощи api, используя для этого сервисный ключ доступа. Выполнить к 6 июля 2020 года.
6) 7 июля 2020 года произвести первичную сборку проекта рабочего прототипа программы и первичное тестирование функций. Написать отчет для второй итерации.
7) К 8 июля произвести разбор и устранение недочетов, выявленных после сдачи второй итерации.
8) К 10 июля 2020 года произвести полноценное тестирование программы и написать финальную версию отчета.

2.2 Распределение ролей в бригаде

- Быков Андрей – лидер, алгоритмист, фтронтенд.
- Деркач Никита – алгоритмист, тестировщик.
- Логинов Евгений – фротенд, документация.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1 Структуры данных и основные методы

Для реализации проекта потребовалось разработать следующие структуры данных:

Класс Graph отвечает за построение графа и считывание информации из поля ввода. Конструктор этого класса Graph(String text, boolean nonFriends, boolean spanningTree) принимает на вход строку с исходными данными и два флага, отвечающие за дополнительную отрисовку графа.

- Поля:
    - public UsersContainer users – доступ к полям класса контейнера пользователя.
    - public EdgesContainer edges – доступ к ребрам графа.
    - public final boolean nonFriends – флаг для отрисовки пунктирных ребер между людьми, которые не являются друзьями.
    - public final boolean spanningTree – флаг для построения минимального остового дерева.
- Методы:
    - public void addUser(String VkID) – добавление нового пользователя непосредственно по id ВК.
    - public User getUser(int x, int y) – получение координат каждого пользователя для представления на графе.
    - private void buildGraph() – построение графа.
    - private void addEdges() – добавление ребер.
    - private void processText(String text) – считывание строки с ID пользователя и проверка на корректность введенных данных.

- private void buildSpanningTree() – построение минимального остового дерева по признаку наибольшего числа общих друзей, если конечно нажата соответствующая кнопка.

Класс Edge предназначен для хранения конкретного ребра графа. Конструктор класса public Edge(User u1, User u2) принимает на вход двух пользователей и выполняет начальную инициализацию переменных.

- Поля:
  - public boolean isFriends – флаг, показывающий являются ли пользователи друзами.
  - public boolean isInSpanningTree - построение минимального остового дерева по признаку наибольшего числа общих друзей, если нажата соответствующая кнопка. Работает на основе алгоритма Прима.
  - public User user1, user2 – ссылка на двух соседних пользователей.
  - public int weight – хранение веса ребра.
  - public Cords cords – объект класса с координатами.
- Методы:
  - public boolean equals(Edge edge) – проверка двух ребер на эквивалентность.
  - public void drawEdge(Graphics2D g2) – рисование ребра графа.
  - public User connectWith(User user) – метод принимает на вход первого пользователя и возвращает второго пользователя, который соединён с первым.

Класс EdgesContainer предназначен для хранения всех ребер графа. Конструктор класса public void addEdge(User u1, User u2) принимает на ход двух пользователей и выполняет начальную инициализацию переменных.

- Поля:
  - public Edge [] edges – переменная для доступа к полям класса Edge.
  - private int length – длина контейнера, кол-во ребер в контейнере.
- Методы:
  - public void addEdge(User u1, User u2) – добавление ребра между двумя друзами.
  - public boolean find(Edge edge) – проверяет, входит ли ребро в контейнер.
  - public int buildEdges(int vertexNum) – определение координат ребра для их последующей отрисовки.
  - public void buildWeights() – определение веса ребра.
  - public Edge findMaxEdge(User user, UsersContainer spanningTree, boolean nonFriends) – поиск максимального по весу ребра.

Класс User предназначен для получения информации о пользователе непосредственно из ВК при помощи сервисного ключа доступа и api ВК, который записан в файл .idea/secret.txt. Конструктор класса public User(String VkID, int ID) принимает на вход получает на вход ID пользователя из ВК и ID вершины графа, после этого выполняется инициализация переменных.

- Поля:
  - private final String name – имя пользователя.
  - private final String surname – фамилия пользователя.
  - private final int ID – id в графе, номер каждой вершины.

- public final String VkID – id пользователя из ВК.
- private final String age – возврат пользователя.
- private final String photo – фото пользователя.
- public int friendsNumber – количество друзей пользователя.
- public String [] friends – список друзей.
- public User.Cords cords – координаты расположения вершин с пользователями.

- Методы:
  - public User(String VkID, int ID) throws MyExceptions – присваивание переменным информации о пользователе.
  - static public String [] getUserFriends(String VkID) – осуществляет запрос к ВКонтакте при помощи сервисного ключа доступа и возвращает список ID друзей.
  - static public String [] getUserInfo(String VkID) – получение информации о пользователе при помощи сервисного ключа доступа.
  - public String getName() – получение имени.
  - public String getSurname() – получение фамилии.
  - public String getAge() – получение возраста.
  - public String getPhoto() - получение фото.
  - public int getID() – получение ID.
  - public boolean equals(User u) – сравнение двух пользователей по фамилии.
  - public boolean areFriends(User u) – проверка, являются ли пользователи друзьями.
  - public void drawUser(Graphics2D g2) – добавление пользователя в виде вершины графа.

Класс UserContainer представляет собой контейнер для хранения информации о пользователях.

- Поля:
  - public User [] users – массив пользователей.
  - private int length – длина контейнера, количество пользователей.
- Методы:
  - public UsersContainer() – конструктор класса, начальная инициализация переменных.
  - public void addUser(String VkID) – добавление нового пользователя в контейнер.
  - public User getUser(String name, String surname) – получение информации о пользователе.
  - public int getLength() – получение информации о количестве пользователей в контейнере.
  - public int getLength() – проверка на то, входит ли данный пользователь в контейнер.
  - public void buildUsers() – определение координат вершин для отрисовки пользователей в вершинах графа.

3.2 Классы для GUI

VKGui – класс графического интерфейса, реализует главное окно программы, поля для ввода, кнопки и их реализацию. Класс наследуется от JFrame, благодаря чему получает доступ ко множеству методов для настройки главного окна. Вложенные классы, реализующие действие кнопки, наследуются от класса ActionListener и переопределяют его методы.

WarningDioalog – класс диалоговых окон, наследуется от JDialog. Используется при вводе некорректных данных.

Класс InfoPanel – наследуется от JPanel и реализует окно с информацией при нажатии на вершину с пользователем.

HelpPanel – наследуется от JFrame, данный класс реализует окно с информацией, которое появляется при нажатии на кнопку “INFO”.

Класс GraphPanel – реализация панели со scroll барами для визуализации графа.

Класс ColorGenerate позволяет изменять цвета вершин графа при изменении масштаба (Рисунок 6 и 7).

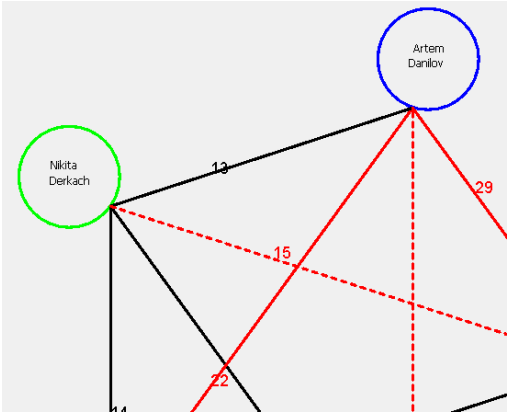


Рисунок 6.

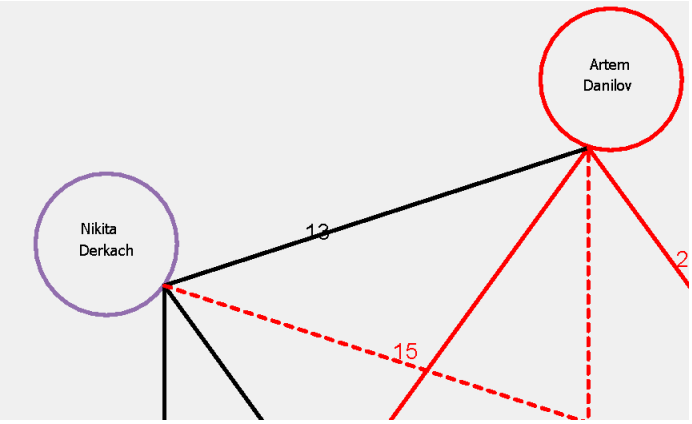


Рисунок 7.

4. ТЕСТИРОВАНИЕ

4.1 Ручное тестирование программы

Ручное тестирование программы проводилось для выявления слабых мест. Проверялось корректность построение графа общих друзей, проверка на корректность введенных данных, тестировался графический интерфейс и все его составляющие. Пример ручного тестирования представлен на рисунке 8.

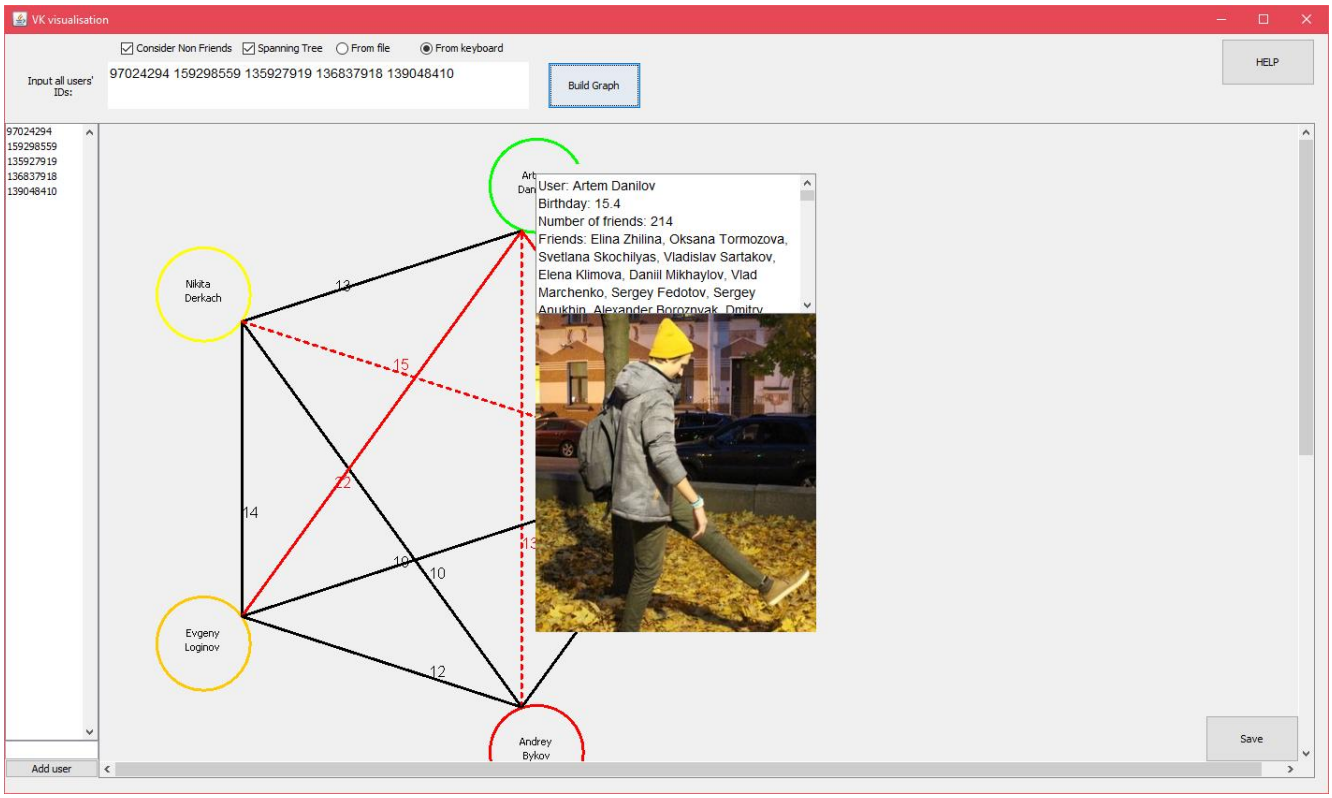


Рисунок 8 – пример ручного тестирования.

4.2 Unit-тестирование программы

Проведено автоматическое тестирование основных функций с использованием библиотеки JUnit и Unit-тестов. Классы со строкой “Test” в названии вызывают и передают в тестируемые функции входные данные, после этого функция Assert() проверяет, совпадает ли ожидаемый результат с полученным.

Для проверки тестирования класса Edge был написан класс EdgeTest, проверяющий методы сравнения двух ребер Equals() и метод соединения двух пользователей (ребро между вершинамит) connectWith().

Для тестирования функций класса EdgeContainer был написан класс EdgeContainerTest, проверяющий на корректность функции добавления ребра addEdge(), вычисления веса ребра buildWeights() и поиска максимального по весу ребра findMaxEdge().

Для тестирования функций класса Graph был написан класс GraphTest, проверяющий на корректность методы добавления нового пользователя addUser() и считывания исходных данных processText().

Для тестирования функций класса User был написан класс UserTest, проверяющий на корректность методы запроса списка друзей пользователя из ВК getUserFriends() и запроса основной информации пользователя getUserInfo().

Для тестирования функций класса UserContainer был написан класс UserContainerTest, проверяющий на корректность метод добавление нового пользователя в контейнер addUser().

После всех перечисленных тестов, можно сказать, что программа работает корректно. UNIT-тесты представлены в приложении В.

ЗАКЛЮЧЕНИЕ

Разработка поставленной задачи была выполнена соответствии с планом и требованиями технического задания. Было спроектировано и реализовано приложение “VK Clique” для визуализации общих друзей в виде графа. Также был разработан удобный и понятный графический интерфейс пользователя, позволяющий ознакомиться с графом общих друзей ВК.

Кроме того, в приложении были добавлены дополнительные функции, позволяющие просмотреть основную информацию о данном аккаунте при нажатии на вершину с пользователем, добавить нового пользователя в граф, сохранить снимок графа в виде файла формата .png.

Работоспособность приложения основана на алгоритме Прима, позволяющем построить минимальное остовное дерево.

Подводя итог, можно сказать, что поставленные задачи были выполнены успешно, о чем говорит корректное прохождение тестирования программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления

2. Официальная документация к Java: https://docs.oracle.com/en/java/javase/

3. Java. Эффективное программирование. Блох Джошуа 2014 год

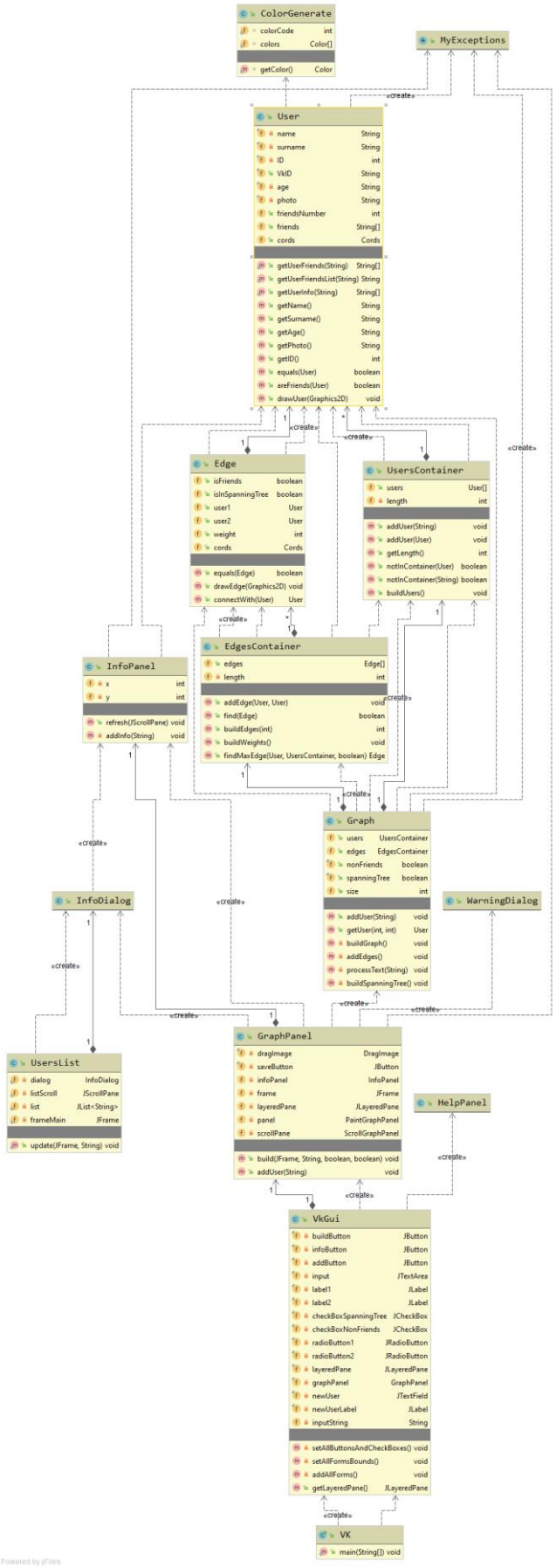
4. Учебный курс по основам Java на Stepik: https://stepik.org/course/187/

5. Википедия: https://ru.wikipedia.org

6. https://ru.stackoverflow.com/

7. https://habr.com/ru/

ПРИЛОЖЕНИЕ А. UML-ДИАГРАММА



ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл VK.java

```
import javax.swing.*;

public class VK {

   public static void main(String[] args) throws ClassNotFoundException,
InstantiationException, IllegalAccessException, ClassCastException,
UnsupportedLookAndFeelException{

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
       VkGui myGui = new VkGui();
       myGui.setVisible(true);
    }
}
```

Файл VkGui.java

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;

public class VkGui extends JFrame {
   private final JButton buildButton = new JButton("Build Graph");
   private final JButton infoButton = new JButton("HELP");
   private final JButton addButton = new JButton("Add user");
   private final JTextArea input = new JTextArea("97024294 159298559
135927919 136837918 139048410");
   private final JLabel label1 = new JLabel("Input all users'");
   private final JLabel label2 = new JLabel(" IDs:");
   private final JCheckBox checkBoxSpanningTree = new JCheckBox("Spanning
Tree",false);
   private final JCheckBox checkBoxNonFriends = new JCheckBox("Consider
Non Friends",false);
   private final JRadioButton radioButton1 = new JRadioButton("From file");
```

```
    private final JRadioButton radioButton2 = new JRadioButton("From
keyboard");
    private final JLayeredPane layeredPane = new JLayeredPane();
    private final GraphPanel graphPanel = new GraphPanel();
    private final JTextField newUser = new JTextField();
    private final JLabel newUserLabel = new JLabel("Input User ID");

    private String  inputString;


    class BuildButtonActionListener implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) {
            inputString = radioButton1.isSelected() ? readFile() : input.getText();
            graphPanel.build(VkGui.this, inputString,
checkBoxNonFriends.isSelected(), checkBoxSpanningTree.isSelected());
            UsersList.update(VkGui.this, inputString);
        }
        private String readFile(){
            StringBuilder text = new StringBuilder();
            try(FileReader reader = new FileReader("tests.txt"))
            {
                int c;
                while((c = reader.read()) != -1){
                    text.append ((char) c);
                }
            }
            catch(IOException ex){

                System.out.println(ex.getMessage());
            }
            return  text.toString();
        }
    }
    class AddButtonActionListener implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) {
            String id = newUser.getText();
            if (id.length() == 0){
                new WarningDialog(VkGui.this, "Ошибка", true, "Введите ID");
                return;
            }
            graphPanel.addUser(id);
            inputString += " " + id;
```

```
            UsersList.update(VkGui.this, inputString);

            //SwingUtilities.updateComponentTreeUI(VkGui.this);
            //graphPanel.build(VkGui.this, inputString,
checkBoxNonFriends.isSelected(), checkBoxSpanningTree.isSelected());
        }
    }

    public VkGui(){
        super("VK visualisation");
        this.setBounds(100, 100, 1430, 850);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setAllFormsBounds();
        setAllButtonsAndCheckBoxes();
        addAllForms();
        new HelpPanel();
    }
    private void setAllButtonsAndCheckBoxes(){
        buildButton.addActionListener(new BuildButtonActionListener());
        addButton.addActionListener(new AddButtonActionListener());
        infoButton.addActionListener(e -> new HelpPanel());
        checkBoxSpanningTree.addActionListener(e -> graphPanel.build(this,
inputString, checkBoxNonFriends.isSelected(),
checkBoxSpanningTree.isSelected()));
        checkBoxNonFriends.addActionListener(e -> graphPanel.build(this,
inputString, checkBoxNonFriends.isSelected(),
checkBoxSpanningTree.isSelected()));
        radioButton1.addActionListener(e -> {
            input.setEnabled(false);
            input.setBackground(Color.LIGHT_GRAY);
        });
        radioButton2.addActionListener(e -> {
            input.setEnabled(true);
            input.setBackground(Color.WHITE);
        });

        ButtonGroup group = new ButtonGroup();
        group.add(radioButton1);
        group.add(radioButton2);

        radioButton2.setSelected(true);
        checkBoxSpanningTree.setSelected(true);
        checkBoxNonFriends.setSelected(true);
    }
```

```
    private void setAllFormsBounds(){
        infoButton.setBounds(1300, 5, 100, 50);
        label1.setBounds(25,45,80,10);
        label2.setBounds(50,57,80,10);
        input.setBounds(110,30,450,50);
        input.setFont(new Font("Dialog", Font.PLAIN, 14));
        buildButton.setBounds(580,30,100,50);
        checkBoxNonFriends.setBounds(120,5,130,20);
        checkBoxSpanningTree.setBounds(250,5,100,20);
        radioButton1.setBounds(350,5,80,20);
        radioButton2.setBounds(440,5,100,20);
        newUser.setBounds(0, 755, 100, 20);
        addButton.setBounds(0, 775, 100, 20);
        layeredPane.setBounds(100,95, 1400, 700);
    }
    private void addAllForms(){
        Container container = this.getContentPane();
        container.setLayout(null);
        container.add(radioButton1);
        container.add(radioButton2);
        container.add(label1);
        container.add(label2);
        container.add(input);
        container.add(buildButton);
        container.add(checkBoxSpanningTree);
        container.add(checkBoxNonFriends);
        container.add(newUser);
        container.add(newUserLabel);
        container.add(addButton);
        container.add(infoButton);
        container.add(layeredPane);
    }
    @Override
    public JLayeredPane getLayeredPane(){
        return layeredPane;
    }
}
```

Файл WarningDialog.java

```
import javax.swing.*;

public class WarningDialog extends JDialog {
    public WarningDialog(java.awt.Frame owner, String title, boolean modal,
```

```
Exception e){
        super(owner, title, modal);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setSize(600,120);
        setLocationRelativeTo(null);
        add(new JLabel(e.getMessage()));
        setVisible(true);
    }
    public WarningDialog(java.awt.Frame owner, String title, boolean modal, String
message){
        super(owner, title, modal);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setSize(600,120);
        setLocationRelativeTo(null);
        add(new JLabel(message));
        setVisible(true);
    }
}
```

Файл ColorGenerate.java

```
import java.awt.*;

public class ColorGenerate {
    static int colorCode;
    static Color [] colors = {Color.RED, Color.ORANGE, Color.YELLOW,
Color.GREEN, Color.BLUE,
            new Color(58,24,177), new Color(146,110,174)};
    static Color getColor(){
        if(colorCode == 6){
            colorCode = 0;
        }
        else{
            colorCode++;
        }
        return colors[colorCode];
    }
}
```

Файл Edge.java

```
import java.awt.*;

public class Edge {
   public boolean isFriends;
   public boolean isInSpanningTree;
   public User user1;
   public User user2;
   public int weight;
   public Cords cords;
   public static class Cords{
      public int x1;
      public int y1;
      public int x2;
      public int y2;
      public int xText;
      public int yText;
   }
   public Edge(User u1, User u2){
      this.isInSpanningTree = false;
      this.isFriends = u1.areFriends(u2);
      user1 = u1;
      user2 = u2;
      weight = 0;
      cords = new Cords();
   }
   public Edge(){
      user1 = new User();
      user2 = new User();
      this.isInSpanningTree = false;
      this.isFriends = false;
      weight = -1;
      cords = new Cords();
   }
   public boolean equals(Edge edge){
      return ((this.user1.equals(edge.user1)  && this.user2.equals(edge.user2))
            || (this.user2.equals(edge.user1) && this.user1.equals(edge.user2)));
   }
   public void drawEdge(Graphics2D g2){
      if(!isFriends) {
         float[] dashl = {5,5};
         BasicStroke pen = new
BasicStroke(3,BasicStroke.CAP_ROUND,BasicStroke.JOIN_BEVEL,10, dashl,0);
         g2.setStroke(pen);
      }
```

```
        else {
            BasicStroke pen = new
BasicStroke(3,BasicStroke.CAP_ROUND,BasicStroke.JOIN_BEVEL,0);
            g2.setStroke(pen);
        }
        if(isInSpanningTree)
            g2.setColor(Color.RED);
        else
            g2.setColor(Color.BLACK);
        g2.drawLine(cords.x1, cords.y1, cords.x2, cords.y2);
        g2.setFont(new Font("Dialog", Font.PLAIN, 16));
        g2.drawString(Integer.toString(weight), cords.xText, cords.yText);
    }
    public User connectWith(User user){
        if(user.equals(user1))
            return user2;
        if(user.equals(user2))
            return user1;
        System.out.println(user1.getSurname() + " " + user2.getSurname() + " " +
user.getSurname());
        return null;
    }
}
```

Файл EdgesContainer.java

```
public class EdgesContainer {
    public Edge [] edges;
    private int length;
    public EdgesContainer(){
        length = 0;
        edges = new Edge[0];
    }
    public void addEdge(User u1, User u2){
        Edge newEdge = new Edge(u1, u2);
        if(find(newEdge))
            return;
        length++;
        Edge [] newArr = new Edge [length];
        System.arraycopy(edges, 0, newArr, 0, edges.length);
```

```
        newArr[length - 1] = newEdge;
        edges = newArr;
    }
    public boolean find(Edge edge){
        for (Edge value : edges) {
            if (value.equals(edge))
                return true;
        }
        return false;
    }
    public int buildEdges(int vertexNum){
        int R = (int)(vertexNum * 200 / 2 / Math.PI) * 2 - 50;
        for(Edge edge : edges){
            edge.cords.x1 = (int)(100 + R + R * Math.cos(2 * Math.PI *
edge.user1.getID() / vertexNum));
            edge.cords.y1 = (int)(100 + R + R * Math.sin(2 * Math.PI *
edge.user1.getID() / vertexNum));
            edge.cords.x2 = (int)(100 + R + R * Math.cos(2 * Math.PI *
edge.user2.getID() / vertexNum));
            edge.cords.y2 = (int)(100 + R + R * Math.sin(2 * Math.PI *
edge.user2.getID() / vertexNum));
            edge.cords.xText = edge.cords.x1 + (edge.cords.x2- edge.cords.x1)/3;
            edge.cords.yText = edge.cords.y1 + (edge.cords.y2- edge.cords.y1)/3;
        }
        return R;
    }
    public void buildWeights(){
        for(Edge edge : edges){
            edge.weight = 0;
            for(int i = 0; i < edge.user1.friendsNumber; i++){
                for(int j = 0; j < edge.user2.friendsNumber; j++){
                    if(edge.user1.friends[i].equals(edge.user2.friends[j])){
                        edge.weight++;
                    }
                }
            }
        }
    }
    public Edge findMaxEdge(User user, UsersContainer spanningTree, boolean
nonFriends){
        Edge maxEdge = new Edge();
        int max = -1;
        for(Edge edge : edges){
            if(!nonFriends && !edge.isFriends) continue;
```

```
            if((edge.user1.equals(user) && edge.weight > max &&
spanningTree.notInContainer(edge.user2))
            || (edge.user2.equals(user) && edge.weight > max &&
spanningTree.notInContainer(edge.user1))){
                max = edge.weight;
                maxEdge = edge;
            }
        }
        return  maxEdge;
    }
}
```

Файл Graph.java

```
public class Graph {
    public UsersContainer users;
    public EdgesContainer edges;
    public final boolean nonFriends;
    public final boolean spanningTree;

    public int size;

    public Graph(){
        users = new UsersContainer();
        edges = new EdgesContainer();
        nonFriends = true;
        spanningTree = true;
    }
    public Graph(String text, boolean nonFriends, boolean spanningTree) throws
MyExceptions {
        users = new UsersContainer();
        edges = new EdgesContainer();
        this.nonFriends = nonFriends;
        this.spanningTree = spanningTree;
        processText(text);
        buildGraph();
    }
    public void addUser(String VkID) throws MyExceptions {
        try{
            Double.parseDouble(VkID);
        }
        catch (NumberFormatException e) {
            throw new MyExceptions("Incorrect ID format, only integer value");
        }
```

```
        users.addUser(VkID);
        buildGraph();
    }

    public User getUser(int x, int y){
        for(User user : users.users){
            if((Math.pow((user.cords.x + 50 - x), 2) + Math.pow((user.cords.y + 50 -
y), 2)) < 50 * 50)
                return user;
        }
        return null;
    }

    private void buildGraph(){
        addEdges();
        size = edges.buildEdges(users.getLength());
        edges.buildWeights();
        users.buildUsers();
        if(spanningTree)
            buildSpanningTree();
    }
    private void addEdges(){
        for(int i = 0; i < users.getLength() - 1; i++){
            for(int j = i + 1; j < users.getLength(); j++){
                edges.addEdge(users.users[i], users.users[j]);
            }
        }
    }


    private void processText(String text) throws MyExceptions {
        if(text == null) return;
        String [] strings = text.split(" ");
        for(String str : strings){
            if(str.equals("")) continue;
            try{
                Double.parseDouble(str);
            }
            catch (NumberFormatException e) {
                throw new MyExceptions("Incorrect ID format, only integer value");
            }
            users.addUser(str);
        }
    }
```

```
    private void buildSpanningTree(){
        UsersContainer ostovUsers = new UsersContainer();
        if(users.getLength() < 2) return;
        ostovUsers.addUser(users.users[0]);
        while(ostovUsers.getLength() < users.getLength()){
            int max = -1;
            Edge maxEdge = edges.edges[0];
            User maxUser = ostovUsers.users[0];
            for(User curr : ostovUsers.users){
                Edge currEdge = edges.findMaxEdge(curr, ostovUsers, nonFriends);
                if(currEdge.weight > max){
                    max = currEdge.weight;
                    maxEdge = currEdge;
                    maxUser = currEdge.connectWith(curr);
                }
            }
            if(max == -1) break;
            maxEdge.isInSpanningTree = true;
            ostovUsers.addUser(maxUser);
        }
    }
}
```

Файл GraphPanel.java

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.AffineTransform;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

public class GraphPanel{
    private final DragImage dragImage = new DragImage();
    private final JButton saveButton = new JButton("Save");
    private InfoPanel infoPanel = new InfoPanel();
    private JFrame frame = new JFrame();
    private JLayeredPane layeredPane = new JLayeredPane();
    private PaintGraphPanel panel = new PaintGraphPanel(new Graph());
    private ScrollGraphPanel scrollPane = new ScrollGraphPanel(panel);

    static class DragImage{
```

```
        public int x;
        public int y;
    }
    class PaintGraphPanel extends JPanel{
        private double zoomFactor = 1;
        private boolean zoomed = true;
        private AffineTransform at;
        private final Graph graph;

        class mouseClickListener implements MouseListener {
            public void mouseClicked(MouseEvent e) {
                layeredPane.remove(infoPanel);
                layeredPane.repaint();
                User user =
graph.getUser((int)(e.getX()/zoomFactor),(int)((e.getY())/zoomFactor) );
                if(user == null) return;
                infoPanel = new InfoPanel(user.VkID, e.getX(), e.getY(), scrollPane);
                layeredPane.add(infoPanel, new Integer(10));
            }
            public void mousePressed(MouseEvent e){
                dragImage.x =  e.getLocationOnScreen().x +
scrollPane.getHorizontalScrollBar().getValue();
                dragImage.y =  e.getLocationOnScreen().y +
scrollPane.getVerticalScrollBar().getValue();
            }
            public void mouseReleased(MouseEvent e){

            }
            public void mouseExited (MouseEvent e) {

            }
            public void mouseEntered (MouseEvent e){

            }
        }
        class mouseMotionListener implements MouseMotionListener {
            public void mouseDragged(MouseEvent e){
                int dx = dragImage.x - e.getLocationOnScreen().x;
                int dy = dragImage.y - e.getLocationOnScreen().y;
                scrollPane.getHorizontalScrollBar().setValue(dx);
                scrollPane.getVerticalScrollBar().setValue(dy);

            }
            public void mouseMoved(MouseEvent e){
```

```
    }
}
class mouseWheelListener implements MouseWheelListener{
    @Override
    public void mouseWheelMoved(MouseWheelEvent e) {
        if (e.getWheelRotation() < 0) {
            setZoomFactor(1.1 * getZoomFactor());
            revalidate();
            repaint();
            //SwingUtilities.updateComponentTreeUI(frame);
        }
        if (e.getWheelRotation() > 0) {
            setZoomFactor(getZoomFactor() / 1.1);
            revalidate();
            repaint();
            //SwingUtilities.updateComponentTreeUI(frame);
        }
        infoPanel.setVisible(false);
    }
}

public PaintGraphPanel(Graph graph){
    this.graph = graph;
    addMouseMotionListener(new mouseMotionListener());
    addMouseListener(new mouseClickListener());
    addMouseWheelListener(new mouseWheelListener());
}
protected void paintComponent(Graphics g){
    super.paintComponent(g);
    setPreferredSize(new Dimension((int)(graph.size * 3 * zoomFactor) + 500,
            (int)(graph.size * 3 * zoomFactor) + 500));
    Graphics2D g2=(Graphics2D)g;
    if (zoomed) {
        scrollPane.getVerticalScrollBar().setValue(0);
        scrollPane.getHorizontalScrollBar().setValue(0);
        at = g2.getTransform();
        at.scale(zoomFactor, zoomFactor);
        zoomed = false;
    }
    g2.transform(at);
    g2.setColor(Color.BLACK);
    for(User user : graph.users.users){
        user.drawUser(g2);
```

```
            }
            for(Edge edge : graph.edges.edges){
                if(graph.nonFriends || edge.isFriends)
                    edge.drawEdge(g2);
            }
        }
        public void setZoomFactor(double factor){
            if(factor<this.zoomFactor && factor > 0.3){
                this.zoomFactor=this.zoomFactor/1.1;
            }
            else if(factor>this.zoomFactor && zoomFactor < 3){
                this.zoomFactor=factor;
            }
            this.zoomed =true;
        }
        public double getZoomFactor() {
            return zoomFactor;
        }
    }
    class ScrollGraphPanel extends JScrollPane{
        public ScrollGraphPanel(JPanel panel){
            super(panel);
            infoPanel.setVisible(false);

setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS
);

setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
            setBounds(0,0, 1300, 700);
            getVerticalScrollBar().addAdjustmentListener(e ->
infoPanel.refresh(scrollPane));
            getHorizontalScrollBar().addAdjustmentListener(e ->
infoPanel.refresh(scrollPane));

        }

    }
    public void build(JFrame frame, String inputString, boolean nonFriends,
boolean spanningTree){
        try {
            this.frame = frame;
            this.layeredPane = frame.getLayeredPane();
            layeredPane.remove(scrollPane);
            panel = new PaintGraphPanel(new Graph(inputString, nonFriends,
```

```
spanningTree));
        scrollPane = new ScrollGraphPanel(panel);
        layeredPane.add(scrollPane, new Integer(5));
        saveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                BufferedImage bImg = new BufferedImage(panel.getWidth() - 400,
panel.getHeight() - 400, BufferedImage.TYPE_INT_RGB);
                Graphics2D cg = bImg.createGraphics();
                panel.paintAll(cg);
                try {
                    if (ImageIO.write(bImg, "png", new File("./output_image.png")))
                    {
                        new InfoDialog("Message", "Graph was saved", 700, 300, 200,
100);
                    }
                } catch (IOException exception) {
                    exception.printStackTrace();
                }
            }
        });
        saveButton.setBounds(1183,633, 100, 50);
        layeredPane.add(saveButton, new Integer(100));
      }
      catch (MyExceptions myEx){
        new WarningDialog(frame, "Ошибка", true, myEx);
      }
   }
   public void addUser(String str){
      try {
        panel.graph.addUser(str);
        panel.revalidate();
        panel.repaint();
      } catch (MyExceptions myEx) {
        new WarningDialog(frame,"Ошибка", true, myEx);
      }

   }
}
```

Файл HelpPanel.java

```
import javax.imageio.ImageIO;
import javax.swing.*;
```

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

public class HelpPanel extends JFrame {
   HelpPanel(){
      super("HELP");
      this.setBounds(700,200, 500, 700);
      this.setVisible(true);
      this.setLayout(null);
      JTextArea info = new JTextArea();
      JLabel picture = new JLabel();
      info.setText("\tHello, this is high level program made by Luxury Genius
Black Team " +
            "subdivision of Black Lives Matter movement.\n\n" +
            "It is our statement and direct message to everyone who is against our
rights being heard.\n\n" +
            "Main purpose of this IT product is to show all naive people how Social
Networks really affect their lives. " +
            "Simple graph visualisation can demonstrate your whole being.\n" +
            "If you are ready to touch your fragile little inner world, read the
instruction below.\n\n" +
            "1)Choose input method(from file or from keyboard)\n" +
            "2)Input data about Social Network users in such form" +
            "\tID1 ID2 ID3 ...\n" +
            "3)Press button BUILD GRAPH\n" +
            "4)If you forgot somebody, you can add him or her by using field in right
upper corner\n" +
            "5)Now you can see nearly everything but if you want to go further press
PAINT SPANNING TREE\n" +
            "6)Your life has changed forever");
      info.setEditable(false);
      info.setLineWrap(true);
      info.setWrapStyleWord(true);
      info.setFont(new Font("Dialog", Font.PLAIN, 14));
      JScrollPane scrollPane = new JScrollPane(info,
JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCR
OLLBAR_NEVER);
      scrollPane.setBounds(0, 0, 485, 300);

      try {
         BufferedImage myPicture = ImageIO.read(new File("Power.png"));
         Image image = myPicture.getScaledInstance(400,400, 300);
```

```
            picture = new JLabel(new ImageIcon(image));
            picture.setBounds(45,300, 400, 400);
        } catch (IOException e) {
            e.printStackTrace();
        }
        this.getLayeredPane().add(picture, JLayeredPane.POPUP_LAYER);
        this.getLayeredPane().add(scrollPane, JLayeredPane.POPUP_LAYER);
        this.setAlwaysOnTop(true);
    }
}
```

Файл InfoDialog.java

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

public class InfoDialog extends JFrame {
    public InfoDialog(String title, String message, int x, int y,  int width, int height){
        super(title);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        JTextArea infoText = new JTextArea(message);
        infoText.setLineWrap(true);
        infoText.setWrapStyleWord(true);
        infoText.setFont(new Font("Dialog", Font.PLAIN, 14));
        infoText.setEditable(false);
        JScrollPane scrollInfoText = new JScrollPane(infoText,
                JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
                JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        scrollInfoText.setBounds(0,0,width-15,height-40);
        add(scrollInfoText);
        setVisible(true);
        setAlwaysOnTop(true);
        this.setBounds(x, y, width, height);
    }
    public InfoDialog(String VkID, int x, int y,  int width, int height){
        super("User Info");
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        add(new InfoPanel(VkID, x, y, new JScrollPane()));
        setVisible(true);
        setAlwaysOnTop(true);
        this.setBounds(x, y, width, height);
    }
    public InfoDialog(){
        super();
```

```
    }
}
```

Файл InfoPanel.java

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.WindowEvent;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.URL;

public class InfoPanel extends JPanel {
    private int x;
    private int y;
    public InfoPanel(String VkID, int x, int y, JScrollPane scrollPane){
        setVisible(true);
        setLayout(null);
        addInfo(VkID);
        this.x = x;
        this.y = y;
        refresh(scrollPane);
    }
    public InfoPanel(){
        super();
    }
    public void refresh(JScrollPane scrollPane){
        if(y < 500)
            setBounds(x - scrollPane.getHorizontalScrollBar().getValue(),
                    y - scrollPane.getVerticalScrollBar().getValue(), 300, 500);
        else
            setBounds(x - scrollPane.getHorizontalScrollBar().getValue(),
                    y - scrollPane.getVerticalScrollBar().getValue() - 500, 300, 500);
    }
    private void addInfo(String VkID){
        try {
            String [] userInfo = User.getUserInfo(VkID);
            URL url;
            try {
                url = new URL(userInfo[3].replaceAll("\\\\", ""));
                BufferedImage c = ImageIO.read(url);
```

```
            JLabel pictureLabel = new JLabel(new ImageIcon(c));
            pictureLabel.setBounds(0,160, 300, 400);
            add(pictureLabel);
        } catch (IOException e) {
            e.printStackTrace();
        }

        String str = "User: " + userInfo[0]
                + " " + userInfo[1] + "\nBirthday: " + userInfo[2]
                + "\nNumber of friends: " ;
        String friends = User.getUserFriendsList(VkID);
        str += friends.split(",").length + "\nFriends: " + friends;
        JTextArea infoText = new JTextArea(str);
        infoText.setLineWrap(true);
        infoText.setWrapStyleWord(true);
        infoText.setBounds(0,10,300,200);
        infoText.setFont(new Font("Dialog", Font.PLAIN, 14));
        JScrollPane scrollInfoText = new JScrollPane(infoText,
                JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
                JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        scrollInfoText.setBounds(0,10,300,150);
        add(scrollInfoText);
      } catch (MyExceptions myExceptions) {
        myExceptions.printStackTrace();
      }


   }
}
```

Файл MyExceptions.java

```
public class MyExceptions extends Exception{
   MyExceptions(String message){
      super(message);
   }
}
```

Файл User.java

```
import java.awt.*;
import java.io.BufferedReader;
import java.io.FileReader;
```

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.StringJoiner;

public class User {
    private final String name;
    private final String surname;
    private final int ID;
    public final String VkID;
    private final String age;
    private final String photo;
    public int friendsNumber;
    public String [] friends;
    public User.Cords cords = new Cords();
    public static class Cords{
        public int x;
        public int y;
    }

    public User(){
        this.name = "";
        this.surname = "";
        this.age = "";
        this.ID = -1;
        this.VkID = "1";
        this.photo = "https://vk.com/images/camera_400.png";
    }
    public User(String VkID, int ID) throws MyExceptions {
        String [] info = getUserInfo(VkID);
        String [] friendsInfo = getUserFriends(VkID);
        this.name = info[0];
        this.surname = info[1];
        this.age = info[2];
        this.photo = info[3].replaceAll("\\\\", "");
        this.ID = ID;
        this.VkID = VkID;
        this.friendsNumber = friendsInfo.length;
        this.friends = friendsInfo;
    }
    static public String [] getUserFriends(String VkID){
        StringBuilder text = new StringBuilder();
        try(FileReader reader = new FileReader(".idea/secret.txt"))
        {
```

```
            int c;
            while((c = reader.read()) != -1){
                text.append ((char) c);
            }
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
        String TOKEN = text.toString();
        String url = "https://api.vk.com/method/" +
                "friends.get" +
                "?user_id=" + VkID +
                "&access_token=" + TOKEN +
                "&v=5.52";
        String line = "";
        try {
            URL url2 = new URL(url);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(url2.openStream()));
            line = reader.readLine();
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return line.split("\\[")[1].split("]")[0].split(",");
    }
    static public String getUserFriendsList(String VkID){
        StringBuilder text = new StringBuilder();
        try(FileReader reader = new FileReader(".idea/secret.txt"))
        {
            int c;
            while((c = reader.read()) != -1){
                text.append ((char) c);
            }
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
        String TOKEN = text.toString();
        String url = "https://api.vk.com/method/" +
                "friends.get" +
                "?user_id=" + VkID +
                "&access_token=" + TOKEN +
                "&fields=first_name,last_name"+
```

```
                "&v=5.52";
        String line = "";
        try {
            URL url2 = new URL(url);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(url2.openStream()));
            line = reader.readLine();
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        line = line.replaceAll("\\d","").replaceAll("\\[]", "");
        StringJoiner join = new StringJoiner(", ");
        try {
            String[] friendsInfo = line.split("\\[")[1].split("]")[0].split("},\\{");
            for (String str : friendsInfo) {
                str = str.split("\"")[5] + " " + str.split("\"")[9];
                join.add(str);
            }
        }
        catch (ArrayIndexOutOfBoundsException e){
            return "Closed friends";
        }
        return join.toString();
    }
    static public String [] getUserInfo(String VkID) throws MyExceptions {
        StringBuilder text = new StringBuilder();
        try(FileReader reader = new FileReader(".idea/secret.txt"))
        {
            int c;
            while((c = reader.read()) != -1){
                text.append ((char) c);
            }
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
        String TOKEN = text.toString();
        String[] info = new String [4];
        String url = "https://api.vk.com/method/" +
                "users.get" +
                "?user_id=" + VkID +
                "&access_token=" + TOKEN +
                "&fields=photo_400_orig,bdate"+
```

```
                "&v=5.52";
        String line = "";
        try {
            URL url2 = new URL(url);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(url2.openStream()));
            line = reader.readLine();
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try{
            info[0] = line.split("\"")[7];
            info[1] = line.split("\"")[11];
            info[2] = line.split("\"")[15];
            info[3] = line.split("\"")[19];
        }
        catch (ArrayIndexOutOfBoundsException e){
            if(e.getMessage().equals("19")){
                info[3] = info[2];
                info[2] = "Not specified";
            }
            else
                throw new MyExceptions("Error has occurred with ID " + VkID);
        }
        return info;
    }
    public String getName(){
        return name;
    }
    public String getSurname(){
        return surname;
    }
    public String getAge(){return age;}
    public String getPhoto(){return photo;}
    public int getID(){
        return ID;
    }
    public boolean equals(User u){
        return (u.getName().equals(this.name) &&
u.getSurname().equals(this.surname));
    }
    public boolean areFriends(User u){
        for (String id : u.friends)
```

```
            if(VkID.equals(id))
                return true;
        return false;
    }
    public void drawUser(Graphics2D g2){
        BasicStroke pen = new
BasicStroke(3,BasicStroke.CAP_ROUND,BasicStroke.JOIN_BEVEL,0);
        g2.setColor(ColorGenerate.getColor());
        g2.setStroke(pen);
        g2.drawOval(cords.x,cords.y, 100,100);
        g2.setColor(Color.BLACK);
        g2.drawString(name, cords.x + 50 - (float)name.length()/2 * 6,cords.y + 43);
        g2.drawString(surname, cords.x + 50 - (float)surname.length()/2 *
5.5f,cords.y + 58);
    }
}
```

Файл UsersContainer.java

```
public class UsersContainer {
    public User [] users;
    private int length;
    public UsersContainer(){
        length = 0;
        users = new User[0];
    }
    public void addUser(String VkID) throws MyExceptions {
        if(notInContainer(VkID)){
            length++;
            User [] newArr = new User [length];
            System.arraycopy(users, 0, newArr, 0, users.length);
            newArr[length - 1] = new User(VkID, length);
            users = newArr;
        }
    }
    public void addUser(User user){
        if(notInContainer(user)){
            length++;
            User [] newArr = new User [length];
            System.arraycopy(users, 0, newArr, 0, users.length);
            newArr[length - 1] = user;
            users = newArr;
```

```
        }
    }
    public int getLength(){
        return length;
    }
    public boolean notInContainer(User user){
        for(User curr : users){
            if(user.equals(curr))
                return false;
        }
        return true;
    }
    public boolean notInContainer(String VkID){
        for(User curr : users){
            if(curr.VkID.equals(VkID))
                return false;
        }
        return true;
    }
    public void buildUsers(){
        int R = (int)(length * 200 / 2 / Math.PI) * 2;
        for(User user : users){
            user.cords.x = (int)(R + R * Math.cos(2 * Math.PI * user.getID()/length));
            user.cords.y = (int)(R + R * Math.sin(2 * Math.PI * user.getID()/length));
        }
    }
}
```

Файл UserList.java

```
import javax.swing.*;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import java.awt.event.WindowEvent;

public class UsersList {
    static private InfoDialog dialog = new InfoDialog();
    static private JScrollPane listScroll = new JScrollPane();
    static private JList<String> list = new JList();
    static private JFrame frameMain;
    static class SelectionListener implements ListSelectionListener {
        @Override
        public void valueChanged(ListSelectionEvent e) {
            if(!e.getValueIsAdjusting()) return;
```

```
        dialog.dispatchEvent(new WindowEvent(dialog,
WindowEvent.WINDOW_CLOSING));
        dialog = new InfoDialog(list.getSelectedValue(), frameMain.getX() + 100,
frameMain.getY() + 127,315, 600);
      }
   }
   static public void update(JFrame frame, String inputString){
         frameMain = frame;
         inputString = inputString.replaceAll("[ ]+", " ");
         frame.getContentPane().remove(listScroll);
         list = new JList(inputString.split(" "));
         list.setBounds(0,95,100,660);
         list.addListSelectionListener(new SelectionListener());
         listScroll = new JScrollPane(list,
JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
         listScroll.setBounds(0,95,100,660);
         frame.getContentPane().add(listScroll);
         listScroll.revalidate();
         listScroll.repaint();
    }
}
```

ПРИЛОЖЕНИЕ В. UNIT-ТЕСТИРОВАНИЕ

Файл EdgeTest.java

```
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
class EdgeTest {
    @Test
    void testEquals() {
        Edge edge = new Edge();
        if (edge.equals(edge)){
            assertTrue(true);
        }
    }
    @Test
    void connectWith() {
        try {
            User user1 = new User("237044688", 1);
            User user2 = new User("135927919", 2);
            Edge edge = new Edge(user1, user2);
            if (edge.connectWith(user1) == user2){
                assertTrue(true);
            }
        }
        catch (MyExceptions ex){
            System.out.println("Not ok");
```

```
        }
    }
}
```

Файл EdgeContainerTest.java

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
class EdgesContainerTest {
    @Test
    void addEdge() {
        try{
            User u1 = new User("135927919", 2);
            User u2 = new User("237044688", 1);
            u1.friends[0] = "237044688";
            u2.friends[0] = "135927919";
            EdgesContainer edges = new EdgesContainer();
            edges.addEdge(u1, u2);
            Edge edge = new Edge(u1, u2);
            System.out.println("OK");
            assertTrue(edges.find(edge));
        }
        catch(MyExceptions ex){
            System.out.println("Not OK, data was correct");
        }
    }
    @Test
    void buildWeights() {
```

```
    try{
        User u1 = new User("135927919", 2);
        User u2 = new User("237044688", 1);
        User u3 = new User("633549608", 3);
        u1.friends[0] = "237044688";
        u1.friends[1] = "633549608";
        u2.friends[0] = "135927919";
        u2.friends[1] = "633549608";
        EdgesContainer edges = new EdgesContainer();
        edges.addEdge(u1, u2);
        edges.buildWeights();
        if (edges.edges[0].weight == 1){
            System.out.println("OK");
            assertTrue(true);
        }
        else {
            System.out.println("Not OK");
        }

    }
    catch(MyExceptions ex){
        System.out.println("Not OK, data was correct");
    }
}
@Test
void findMaxEdge() {
```

```
try{
    User u1 = new User("135927919", 2);
    User u2 = new User("237044688", 1);
    User u3 = new User("633549608", 3);
    User u4 = new User("123456789", 4);
    User u5 = new User("428104856", 5);
    UsersContainer users = new UsersContainer();
    u1.friends[0] = "237044688";
    u1.friends[1] = "663549618";
    u1.friends[2] = "732987469";
    u1.friendsNumber = 3;
    u2.friends[0] = "135927919";
    u2.friends[1] = "663549618";
    u2.friends[2] = "732987469";
    u2.friendsNumber = 3;
    u3.friends[0] = "135927919";
    u3.friends[1] = "237044688";
    u3.friends[2] = "927014388";
    u3.friendsNumber = 3;
    users.addUser(u1);
    users.addUser(u3);
    EdgesContainer edges = new EdgesContainer();
    edges.addEdge(u1, u2);
    edges.addEdge(u1, u3);
    edges.addEdge(u2, u3);
    edges.buildWeights();
```

```
            if (edges.findMaxEdge(u1, users, true).weight == 2){
                System.out.println("OK");
                assertTrue(true);
            }
            else {
                System.out.println("Not OK");
            }
        }
        catch(MyExceptions ex){
            System.out.println("Not OK, data was correct");
        }
    }
}
```

Файл GraphTest.java

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

class GraphTest {
    @org.junit.jupiter.api.Test
    void addUser() {
        try {
            Graph graph = new Graph();
            graph.addUser("blabla");
        } catch (MyExceptions myExceptions) {
            System.out.println("OK");
        }
```

```
    }
    @org.junit.jupiter.api.Test
    void processText(){
        try {
            Graph graph = new Graph();
            graph.processText("234fwe133");
        } catch (MyExceptions myExceptions) {
            System.out.println("OK");
        }
    }
}
```

Файл UserTest.java

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class UserTest {

    @Test
    void getUserFriends() {
        String ID = "135927919";
        User user = new User();
        String[] line = User.getUserFriends(ID);
        for (String str : line){
            try{
                Double.parseDouble(str);
            }
```

```
        catch (NumberFormatException e) {
            System.out.println("Incorrect ID");
            assertFalse(true);
        }
    }
}

@Test
void getUserInfo() {
        try{
            User user = new User("fwer423ew", 2);
            String[] line = user.getUserInfo(user.VkID);
        }
        catch (MyExceptions e) {
            System.out.println("Incorrect ID");
            assertFalse(true);
        }
}

}
```

Файл UsersContainerTest.java

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
class UsersContainerTest {

    @Test
```

```
    void addUser() {
        try{
            User user = new User("4234", 3);
            UsersContainer users = new UsersContainer();
            users.addUser(user.VkID);
        }
        catch (MyExceptions ex){
            System.out.println("Ok");
        }
    }

    @Test
    void testAddUser() {
        try{
            User user = new User("135927919", 1);
            UsersContainer users = new UsersContainer();
            users.addUser(user);
            User user2 = user;
            assertEquals(user2, users.users[0]);
        }
        catch (MyExceptions ex){
            System.out.println("Not ok, data was correct");
        }
    }
}
```