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

ОТЧЕТ

по учебной практике

Тема: VKFriends

Студент гр. 7382 _____ Гаврилов А.В.

Студент гр. 7382 _____ Гиззатов А.С.

Студент гр. 7382 _____ Государкин Я.С.

Руководитель _____ Фирсов М.А.

Санкт-Петербург
2019

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Гаврилов А.В. группы 7382

Студент Гиззатов А.С. группы 7382

Студент Государкин Я.С. группы 7382

Тема практики: VKFriends

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Java с графическим интерфейсом.

Алгоритм: Поиск общих друзей двух пользователей во вконтакте.

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета:

Дата защиты отчета:

Студент _____ Гаврилов А.В.

Студент _____ Гиззатов А.С.

Студент _____ Государкин Я.С.

Руководитель _____ Фирсов М.А.

Аннотация

Темой данной учебной практики является командная итеративная разработка визуализатора алгоритма на языке программирования Java.

Цель работы научиться писать на языке программирования java, также обучиться налаживанию коммуникации между участниками практики, разобраться с декомпозицией обязанностей. Визуализировать любой алгоритм для работы с графами. В работе представлена визуализация поиска общих друзей в ВК.

Summary

The subject of summer practice is team iterative development of algorithm visualization on java programming language. The goal is learning how to program on java language, also learn how to set up communication between participants of the practice with duties decomposition. Visualize any working with graph algorithm. The visualization of searching friends is indicated in the work.

Содержание

ВВЕДЕНИЕ

Основная цель практики – реализация мини проекта, который является визуализацией алгоритма, написанного на языке Java. В данной работе алгоритмом является поиск общих друзей в ВК. Для выполнения этой цели были поставлены задачи – разработка GUI к проекту, отправка запроса к api ВК, обработка и использования полученных данных от серверов ВК. Проект использует 1 алгоритм – поиск общих друзей путем просматривания списков друзей 2 конкретных пользователей и поочередного сравнивания элементы этих списков.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

При запуске проекта будет создаваться отдельное окно. Данное окно будет иметь 3 поля:

1. Поле графического представления графа и визуализации алгоритма. Граф строится по данным, введенными пользователем с помощью графического интерфейса.

2. Поле с элементами управления:

- Добавление вершины в граф.
- Перестройка графа.
- Удаление вершины.
- Удаление графа.

3. Диалог создания графа:

- Ввод URL/ID пользователя.
- Ввод URL/ID двух или более пользователей для нахождения общих друзей.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

1. Обсудить задание, распределить роли, выбрать необходимые средства разработки и структуры данных. Данный пункт задания необходимо выполнить к 2 июля 2019 года.

2. Создать базовый gui. Данный пункт задания необходимо выполнить к 4 июля 2019 года.

3. Добавить возможность строить граф, где вершина - аккаунт, а ребро - кол-во общ. Друзей. Данный пункт необходимо выполнить к 5 июля 2019 года.

4. Добавить возможность добавлять/удалять новые вершины. Данный пункт необходимо выполнить к 6 июля.

5. Реализовать набор unit тестов. Данный пункт необходимо выполнить к 6 июля.

6. Создать интерфейс с кпопками добавления/удаления вершины, удаления всего графа, графическое поле для графа и поле/окно для вывода информации об аккаунте. Данный пункт необходимо выполнить к 8 июля.

7. Свести все наработки в один проект. Данный пункт необходимо выполнить к 10 июля.

8. Выводить некоторую информацию об аккаунте по нажатию на вершину. Данный пункт необходимо выполнить к 11 июля.

2.2. Распределение ролей в бригаде:

- Государкин Ярослав - ответственный за алгоритмы.
- Гаврилов Андрей - ответственный за графический интерфейс.
- Гиззатов Амир - ответственный за юнит тестирование и отчёт.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Используемые структуры данных.

Для реализации проекта потребовались следующие структуры данных:

- VKClient
- VKUser
- ApplicationMain
- MainWindow – наследовается от JFrame.
- Методы из скачанных или встроенных библиотек таких как:
  - Junit – отвечающие за тестирование программы.
  - Swing – отвечающие за построение графического интерфейса.

3.2. Описание классов.

Программа состоит из классов: ApplicationMain, VKClient, VKUser, MainWindow.

- Класс ApplicationMain отвечает за выполнение программы в целом. Он создает граф и вызывает методы для нахождения общих друзей.
- Класс VKClient отвечает за получение списка друзей, списка общих друзей, информации о пользователе, отправка запроса в вк api.

Поля класса VKClient предоставлены в таблице 1, а методы перечислены в таблице 2.

Таблица 1-Поля класса VKClient

Поле	Значение поля
public static final String[] basicArgs	Информация о пользователе

private static final String accessVkApiToken	Ключ для отправки запроса к вк api
private static final String versionVkApi	Версия api вк
private static final String beginVkApi	Начало запроса к api вк
private static final String endVkApi	Конец запроса к api вк

методы
public ArrayList<VKUser> parseFriendsJson(String str)
public ArrayList<Integer > getCommonFriends(int srcId, int[] targetIds)
public VKUser getUser(int userId, String[] args)
public String getUserFriends(int userId, String order, String[] args)
public ArrayList<VKUser> getFriends(int userId, String order, String[] args)
private String createGetRequest(String method, int id, String order, String[] args)
private String getRequest(String url)

• Класс VKUser отвечает за информацию о пользователе во вконтакте. Имеет методы для получения строки из полей класса (userdId, firstName, lastName), получение строки из списка друзей, поиска и удаления пользователя в списке. Поля класса представлены в таблице 3.

Таблица 3 – Поля класса VKUser.

Поле	Значение поля
public int userId	Уникальный номер пользователя.
public String firstName	Имя пользователя.
public String lastName	Фамилия пользователя.

Методы
public String toString()
public static String[] arrayToStrings(ArrayList<VKUser> list)
public static void findAndRemove(ArrayList<VKUser> list, int id)

• Класс MainWindow отвечает за графический интерфейс, строит граф друзей и граф общих друзей пользователя. Поля и методы класса представлены в таблицах 4 и 5.

Таблица 4 – Поля класса MainWindow.

Поле	Значение поля
private final int leftPanelWidth	Высота левой панели
private final int colNum	Количество столбцов в textfild
private JPanel toolBar	Панель инструментов
private JPanel buttonBar	Панель кнопок
private JPanel LeftPanel	Левая панель

private final Color colorForTools	Цвет инструментов
private final Color colorForOther	Цвет графа и текстового поля
private JList<String> mList	Динамический список пользователей
private JTextField InputField	Поле ввода id или URL пользователя
private mxGraph graph	Граф
private VKClient mClient	Информация о пользователе
private ArrayList<VKUser> mUsers	Список пользователей типа VKUser

Методы
protected mxGraphComponent makeGraph()
protected void makeToolbar()
public void actionPerformed(ActionEvent actionEvent)
protected void makeLeftPane(mxGraphComponent graphComponent)
private void setUpVKClient()

3.3 UML – диаграмма классов.

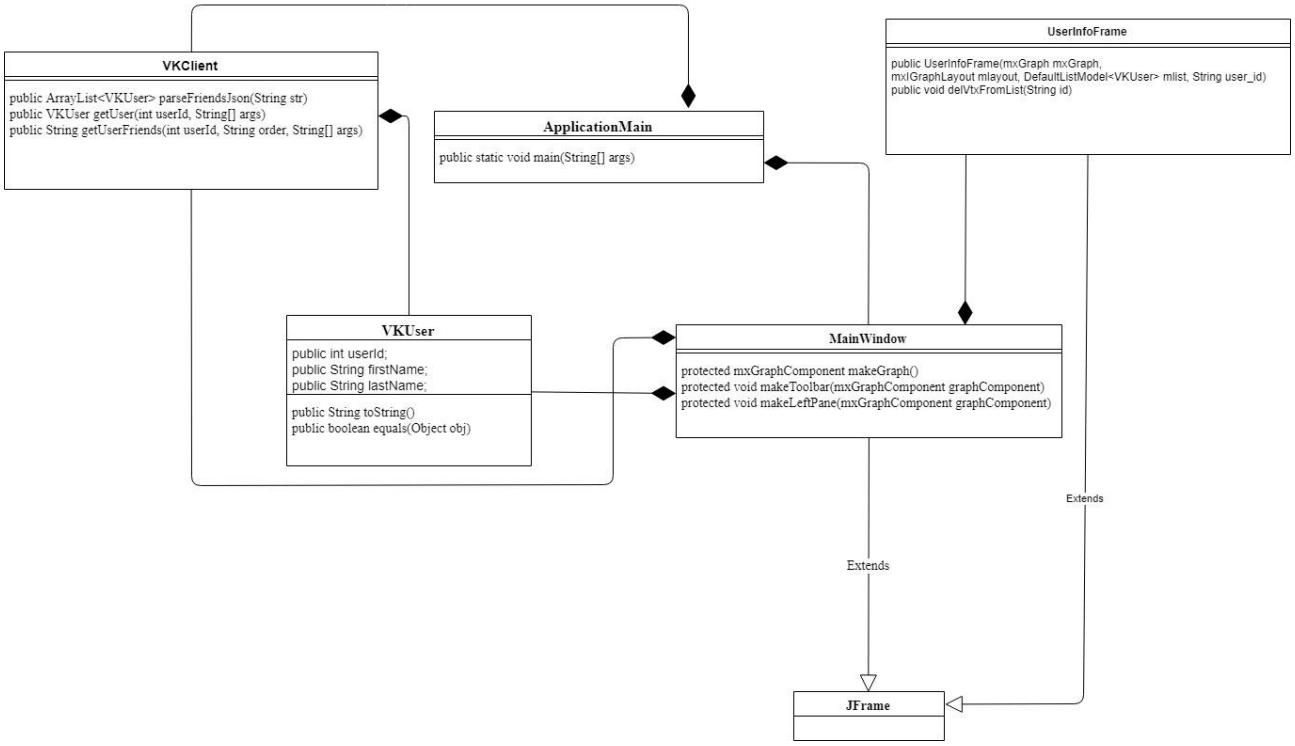

Рис.3 - диаграмма классов.

4. ТЕСТИРОВАНИЕ.

4.1. Примеры работы графического интерфейса.

1) Сперва был создана вершина с 1 пользователем. В списке слева добавилась имя, фамилия и id пользователя. Данная операция представлена на 1 рисунке.

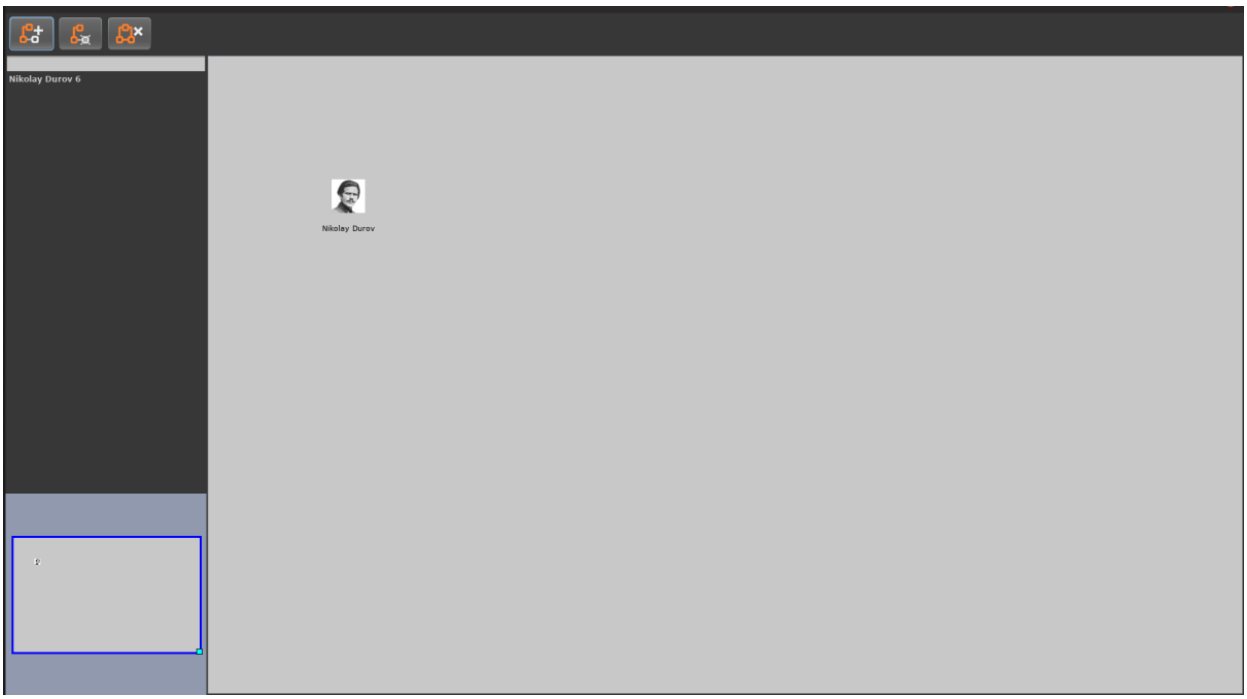

Рис.1.

2) После был добавлен пользователь не имеющий общих друзей с уже добавленным пользователем пользователем. Данная операция представлена на 2 рисунке.

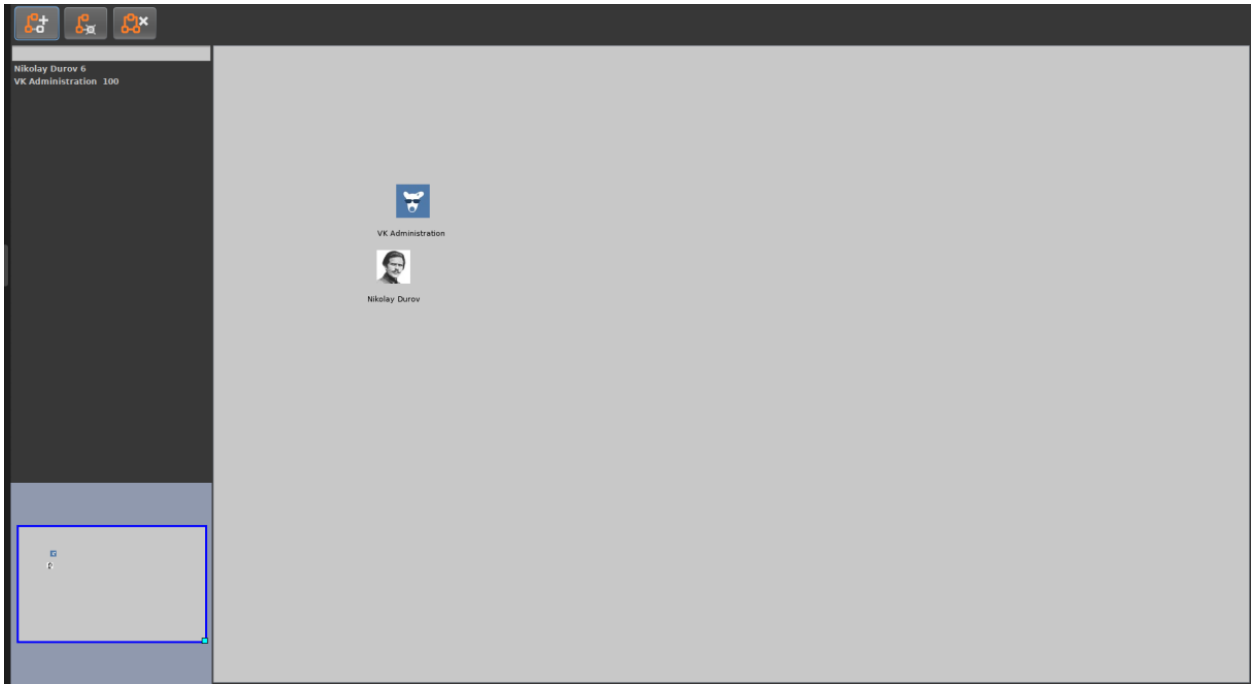

Рис.2.

3) Следующим был добавлены несколько пользователей имеющие общих друзей хоть с 1 из уже добавленных. Между этими пользователями

построились ребра, весами которых являются количество их общих друзей. Данная операция представлена на 3 рисунке.

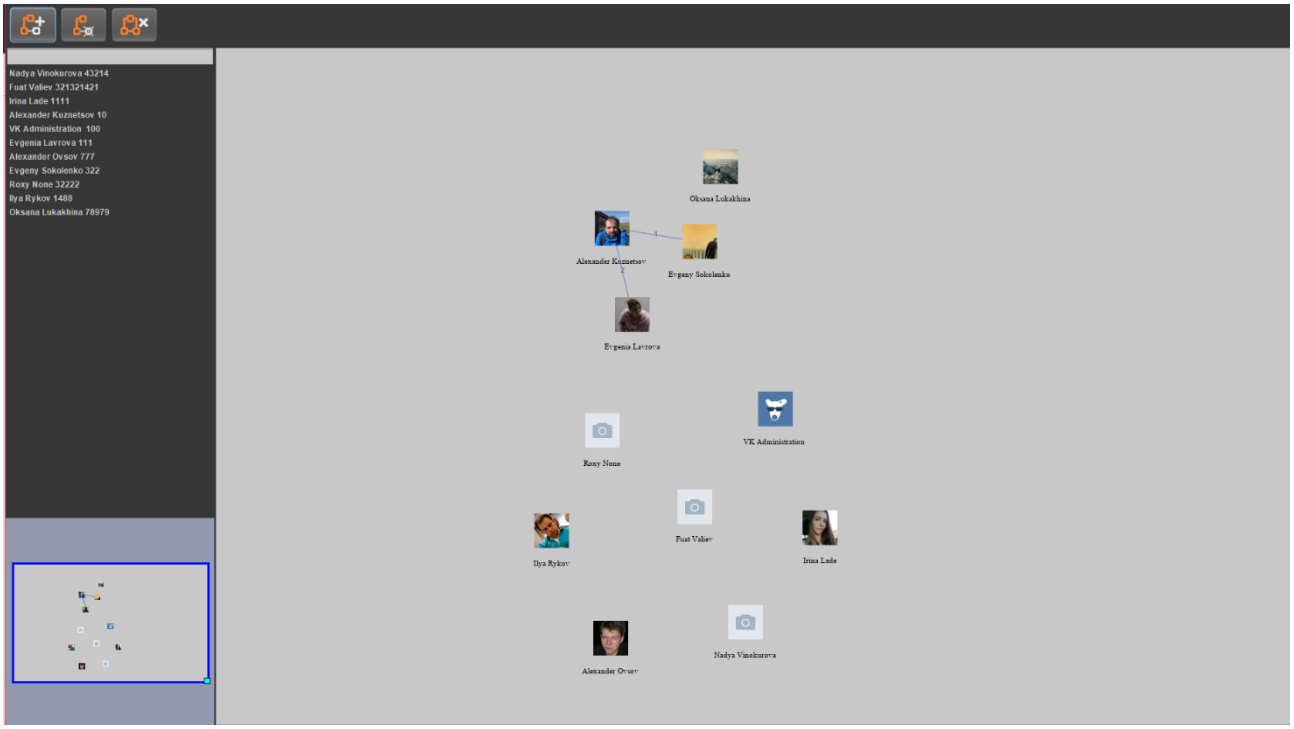

Рис.3.

4) После был удалена вершина с пользователем, который имел хоть 1 общего друга с любым другим пользователем. Данная операция представлена на рисунке 4.

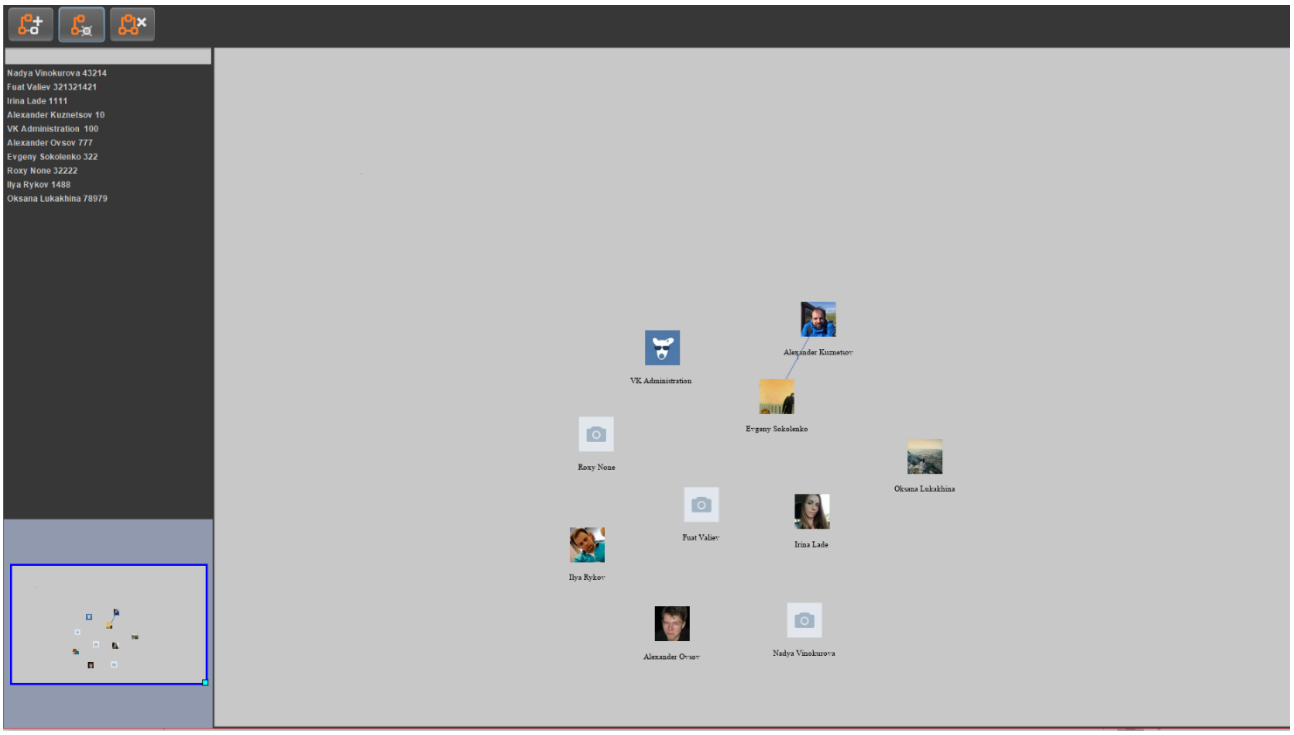

Рис.4.

5) После был очищен весь граф. Данная операция представлена на рисунке 5.

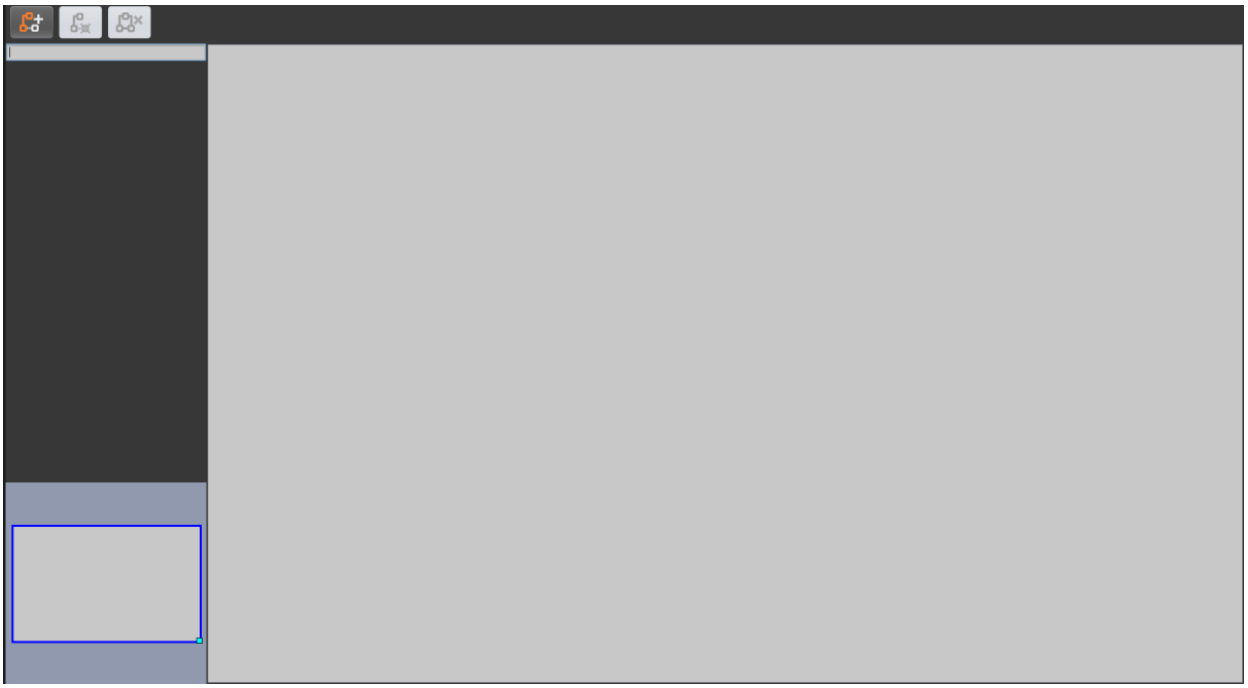

Рис.5.

4.2. Описание методов в юнит тестировании.

Всего было протестировано 3 метода:

- public ArrayList<VKUser> parseFriendsJson(String str)
- public ArrayList<Integer> getCommonFriends(int srcId, int[] targetIds)
- public VKUser getUser(int userId, String[] args)
- Остальные или являются приватными или вызывают эти методы и возвращают их типы.

Метод parseFriendsJson был протестирован с помощью сравнения возвращающегося списка и списка, который берется из файла и является правильным. В приложении В представлен код теста.

Метод getUser был протестирован на корректность взятия информации о пользователе по его id и проверен на списке id друзей одного из участников бригады. После сверен с уже проверенным методом parseFriendsJson.

Метод getCommonFriends был протестирован с помощью сравнения уже известного количества общих друзей и количеством которое возвращает метод. В метод подается id первого пользователя и списка пользователей с которыми надо сравнить первого пользователя.

ЗАКЛЮЧЕНИЕ

Был реализован алгоритм поиска общих друзей во вконтакте с визуализацией. Был разработан интерфейс удобный и понятный для пользователя. Так же в ходе учебной практики были получены знания по языку программирования Java и налажена работа в команде. К тому же были получены навыки создания запросов для получения информации с сайта ВК с помощью ВК api, получен опыт работы с методами класса JFrame для создания графического интерфейса, изучен способ отладки и тестирования кода с помощью библиотеки Junit.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) VK API [Электронный ресурс] URL: https://vk.com/dev/manuals
2) HABR[Электронный ресурс] URL: https://habr.com/
3) ORACLE JFRAME[Электронный ресурс] URL: https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html
4) JUNIT ON JAVARUSH[Электронный ресурс] URL: https://javarush.ru/groups/posts/605-junit

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Класс MainWindow

```
package GUI;

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

import com.mxgraph.layout.*;
import com.mxgraph.swing.*;
import com.mxgraph.view.*;
import com.mxgraph.model.*;

import VKClient.VKClient;
import VKClient.VKUser;

public class MainWindow extends JFrame{
    private final int leftPanelWidth = 300;
    private final int colNum = 20;
    private JPanel toolBar;
    private JPanel buttonBar;
    private JPanel LeftPanel;
    private final Color colorForTools = new Color(55,55,55);
    private final Color colorForOther = new Color(200,200,200);
    private JList<String> mList;
    private DefaultListModel<VKUser> listModel;
    private JTextField InputField;

    private mxGraph graph;
    private mxIGraphLayout layout;

    private VKClient mClient;
    private Integer count = 2;


    protected mxGraphComponent makeGraph(){
        graph = new mxGraph();
        graph.setCellsResizable(false);
```

```
        layout = new mxOrganicLayout(graph);
        mxGraphComponent graphComponent = new mxGraphComponent(graph);
        graphComponent.getGraphControl().addMouseListener(new MouseListener() {
            @Override
            public void mouseClicked(MouseEvent mouseEvent) {
                mxCell vertex = (mxCell)
graphComponent.getCellAt(mouseEvent.getX(),mouseEvent.getY());
                if(vertex==null)
                    return;
                if(!vertex.isVertex())
                    return;
                new UserInfoFrame(graph,layout,listModel, vertex.getId());
            }

            @Override
            public void mousePressed(MouseEvent mouseEvent) {

            }

            @Override
            public void mouseReleased(MouseEvent mouseEvent) {

            }

            @Override
            public void mouseEntered(MouseEvent mouseEvent) {

            }

            @Override
            public void mouseExited(MouseEvent mouseEvent) {

            }
        });
        return graphComponent;
    }

    protected void makeToolbar(){
        toolBar = new JPanel();
        getContentPane().add(LeftPanel,BorderLayout.WEST);
        toolBar.setLayout( new BorderLayout());
```

```
        toolBar.setBackground(colorForTools);

        buttonBar = new JPanel();
        buttonBar.setLayout(new FlowLayout());
        buttonBar.setBackground(colorForTools);

        ImageIcon delImg = new
ImageIcon(System.getProperty("user.dir")+"/assets/Icons/erase.png");
        JButton DelButton = new JButton("",delImg);

        ImageIcon resImg = new
ImageIcon(System.getProperty("user.dir")+"/assets/Icons/graphreset.png");
        JButton DeleteAll = new JButton( "",resImg);

        DelButton.setBackground(colorForTools);
        DelButton.setEnabled(false);
        DelButton.setToolTipText("Erase user from graph");
        DelButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                String userId = InputField.getText();
                InputField.setText("");
                delVertex(userId);
                count--;
                if(count == 2)
                {
                    DelButton.setEnabled(false);
                    DeleteAll.setEnabled(false);
                }
            }
        });

        DeleteAll.setBackground(colorForTools);
        DeleteAll.setEnabled(false);
        DeleteAll.setToolTipText("Delete all graph");
        DeleteAll.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                graph.getModel().beginUpdate();
                Object[] arr=graph.getChildVertices(graph.getDefaultParent());
```

```
                for (Object c: arr)
                {
                    mxCell vertex=(mxCell)c;
                    graph.removeCells(graph.getEdges(vertex));
                    graph.getModel().remove(vertex);
                }
                graph.getModel().endUpdate();
                listModel.removeAllElements();
                DeleteAll.setEnabled(false);
                DelButton.setEnabled(false);
                count = 2;
            }
        });

        ImageIcon addImg = new
ImageIcon(System.getProperty("user.dir")+"/assets/Icons/add.png");
        JButton AddButton = new JButton("", addImg);
        AddButton.setBackground(colorForTools);
        AddButton.setToolTipText("Add new user in graph");
        AddButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                String userId = InputField.getText();
                InputField.setText("");
                VKUser user = null;

                if(userId.equals("1"))
                {
                    getWarningMassage("Magic Durov account." +
((Integer)listModel.size()).toString());
                    return;
                }
                try {
                    user = mClient.getUser(userId, VKClient.basicArgs);
                }catch (Exception e){
                    getWarningMassage(e.getMessage() +
((Integer)listModel.size()).toString());
                    return;
                }
                if(user == null) {
                    getWarningMassage( ((Integer)listModel.size()).toString());
```

```
                    return;
                }
                int[] arrId= new int[listModel.size()];
                for(int i=0;i<listModel.size();i++){
                    if(user.equals(listModel.get(i))) {
                        getWarningMassage("User already exist's");
                        return;
                    }
                    arrId[i]=listModel.get(i).userId;

                }
                ArrayList<Integer> listEdges =
mClient.getCommonFriends(user.userId,arrId);
                graph.getModel().beginUpdate();
                try {
                    Object[] arr=graph.getChildVertices(graph.getDefaultParent());
                    Object vert1 =
graph.insertVertex(graph.getDefaultParent(),((Integer)(user.userId)).toString(),
                            user.firstName+" "+user.lastName,
                            110,100,50,50,

"verticalLabelPosition=bottom;fontColor=black;shape=image;" +
                                    "fontFamily=Times New
Roman;image="+user.photoUrl);
                    for(int i=0;i<listEdges.size();i++){
                        if(listEdges.get(i)==0)
                            continue;
                        for (Object c: arr) {
                            mxCell vert2=(mxCell)c;

if(Integer.parseInt(vert2.getId())==listModel.get(i).userId) {

graph.insertEdge(graph.getDefaultParent(),null,listEdges.get(i).toString(),
                                        vert1,vert2,"endArrow=none");
                            }
                        }
                    }
                }
                finally {
                    DeleteAll.setEnabled(true);
                    DelButton.setEnabled(true);
```

```
                    listModel.addElement(user);
                    layout.execute(graph.getDefaultParent());

                    graph.getModel().endUpdate();
                }
                count++;
            }
        });


        buttonBar.add(AddButton);
        buttonBar.add(DelButton);
        buttonBar.add(DeleteAll);
        buttonBar.setSize(Toolkit.getDefaultToolkit().getScreenSize().width,400);

        toolBar.add( buttonBar, BorderLayout.WEST);
        getContentPane().add( toolBar, BorderLayout.NORTH);
    }

    protected void makeLeftPane(mxGraphComponent graphComponent){
        LeftPanel = new JPanel();
        LeftPanel.setBackground(colorForTools);
        LeftPanel.setLayout(new BorderLayout());

        listModel=new DefaultListModel<>();
        mList = new JList(listModel);
        mList.addMouseListener(new MouseListener() {
            @Override
            public void mouseClicked(MouseEvent e) {
                int ind=mList.getSelectedIndex();
                new UserInfoFrame(graph,layout,listModel,
((Integer)listModel.get(ind).userId).toString());
            }

            @Override
            public void mousePressed(MouseEvent e) {

            }

            @Override
            public void mouseReleased(MouseEvent e) {
```

```
            }

            @Override
            public void mouseEntered(MouseEvent e) {

            }

            @Override
            public void mouseExited(MouseEvent e) {

            }
        });

        mList.setFixedCellWidth(leftPanelWidth);
        mList.setBackground(colorForTools);
        mList.setForeground(colorForOther);
        mList.setFont(new Font("New Roman",Font.BOLD,12));
        mxGraphOutline graphOutline = new mxGraphOutline(graphComponent);
        graphOutline.setPreferredSize(new
Dimension(leftPanelWidth,leftPanelWidth));
        InputField = new JTextField();
        InputField.setColumns(colNum);
        InputField.setBackground(colorForOther);

        JPanel panelForList = new JPanel();
        panelForList.setLayout(new BoxLayout(panelForList,BoxLayout.Y_AXIS));
        panelForList.setBackground(colorForTools);
        panelForList.add(InputField);
        panelForList.add(mList);

        LeftPanel.add( graphOutline, BorderLayout.PAGE_END);
        LeftPanel.add(panelForList, BorderLayout.PAGE_START);
        getContentPane().add(LeftPanel,BorderLayout.WEST);
    }

    private void setUpVKClient()
    {
        mClient = new VKClient();
    }
    private void getWarningMassage(String error){
```

```
    JOptionPane.showMessageDialog(null,error);
}

private void delVertex(String id){
    graph.getModel().beginUpdate();
    try {
        Object[] arr=graph.getChildVertices(graph.getDefaultParent());
        for (Object c: arr) {
            mxCell vertex=(mxCell)c;
            if(id.equals(vertex.getId())) {
                graph.removeCells(graph.getEdges(vertex));
                graph.getModel().remove(vertex);
            }
        }

    }
    finally {
        layout.execute(graph.getDefaultParent());
        delVtxFromList(id);
        graph.getModel().endUpdate();
    }
}

public void delVtxFromList(String id)
{
    for(int i=0; i<listModel.size(); i++)
        if(listModel.get(i).userId == Integer.parseInt(id))
            listModel.remove(i);
}

public MainWindow() throws Exception{
    super("VKFriends");
    try {
        UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");
    }
    catch (Exception e){
        throw e;
    }
    setSize(Toolkit.getDefaultToolkit().getScreenSize());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setUpVKClient();
```

```
        mxGraphComponent graphComponent= makeGraph();
        graphComponent.getViewport().setOpaque(true);
        graphComponent.getViewport().setBackground(colorForOther);
        getContentPane().setBackground(colorForTools);
        getContentPane().add(graphComponent, BorderLayout.CENTER);
        makeLeftPane(graphComponent);
        makeToolbar();

        setVisible(true);
    }
}
```

Класс UserInfoFrame:

```
package GUI;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import VKClient.VKUser;
import com.mxgraph.model.mxCell;
import com.mxgraph.view.mxGraph;
import com.mxgraph.layout.*;

public class UserInfoFrame extends JFrame{
    private mxGraph graph;
    private DefaultListModel<VKUser> list;
    private String id;
    private final Color color = new Color(55,55,55);
    private mxIGraphLayout layout;

    public UserInfoFrame(mxGraph mxGraph, mxIGraphLayout mlayout,
DefaultListModel<VKUser> mlist, String user_id) {
        graph = mxGraph;
        list = mlist;
        id = user_id;
        layout=mlayout;
        setTitle("User Info");
        setAlwaysOnTop(true);
        setResizable(false);
        setSize(400, 300);
```

```
        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        setLocation(dim.width/2-this.getSize().width/2, dim.height/2-
this.getSize().height/2);
        getContentPane().setLayout(new BorderLayout());
        JPanel northPanel = new JPanel();
        northPanel.setBackground(color);
        add(northPanel,BorderLayout.NORTH);

        ImageIcon delImg = new
ImageIcon(System.getProperty("user.dir")+"/assets/Icons/erase.png");
        JButton delUser=new JButton("",delImg);
        delUser.setBackground(color);
        delUser.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                delVertex(id);
                delVtxFromList(id);
                dispose();
            }
        });

        northPanel.setLayout(new BorderLayout());
        northPanel.add(delUser,BorderLayout.EAST);

        JPanel centrePanel = new JPanel();
        centrePanel.setBackground(color);
        add(centrePanel,BorderLayout.CENTER);
        getContentPane().setBackground(color);
        VKUser user=new VKUser(0,null,null,null);
        for(int i=0;i<list.size();i++){
            if(list.get(i).userId==Integer.parseInt(user_id)) {
                user = list.get(i);
                break;
            }
        }

        JTextArea textField=new JTextArea(user.getInfo());
        textField.setEnabled(false);
        textField.setPreferredSize(new Dimension(400,200));
        textField.setFont(new Font("Times New Roman",Font.BOLD,12));
        centrePanel.add(textField,BorderLayout.CENTER);
```

```
        setVisible(true);
    }

    private void delVertex(String id){
        graph.getModel().beginUpdate();
        try {
            Object[] arr=graph.getChildVertices(graph.getDefaultParent());
            for (Object c: arr) {
                mxCell vertex=(mxCell)c;
                if(id.equals(vertex.getId())) {
                    graph.removeCells(graph.getEdges(vertex));
                    graph.getModel().remove(vertex);
                }
            }

        }
        finally {
            layout.execute(graph.getDefaultParent());
            delVtxFromList(id);
            graph.getModel().endUpdate();
        }
    }

    public void delVtxFromList(String id)
    {
        for(int i=0; i<list.size(); i++)
            if(list.get(i).userId == Integer.parseInt(id))
                list.remove(i);
    }
}
```

Класс VKClient:

```
package VKClient;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.security.InvalidParameterException;
import java.util.ArrayList;
import java.net.URL;
import java.net.URLConnection;

import com.google.gson.*;
```

```
public class VKClient {

    public static final String[] basicArgs = {"id", "first_name", "last_name",
"deactivated", "is_closed", "photo_50"
                                                , "bdate", "sex", "education",
"relation"};

    private static final String accessVkApiToken =
"0c91fdfc0c91fdfc0c91fdfc3e0cfa8e5100c910c91fdfc518a61b257703bf5b0589264";
    private static final String versionVkApi = "5.101";
    private static final String beginVkApi = "https://api.vk.com/method/";
    private static final String endVkApi = "&access_token=" + accessVkApiToken +
"&v=" + versionVkApi;

    public ArrayList<VKUser> parseFriendsJson(String str) {
        try
        {
            JsonParser jsonParser = new JsonParser();
            JsonObject jo = (JsonObject)jsonParser.parse(str);
            if (jo.get("error") != null)
            {
                System.out.println("Error code: " +
jo.get("error").getAsJsonObject().get("error_code").getAsInt()
                        + "\ndescription: " +

jo.get("error").getAsJsonObject().get("error_msg").getAsString());
                return null;
            }

            JsonArray jsonArr =
jo.get("response").getAsJsonObject().getAsJsonArray("items");
            ArrayList<VKUser> list = new ArrayList<>();

            for (JsonElement obj : jsonArr)
            {
                JsonObject tmp = obj.getAsJsonObject();
                list.add( new VKUser( tmp.get("id").getAsInt(),
                        tmp.get("first_name").getAsString(),
                        tmp.get("last_name").getAsString(),
                        tmp.get("photo_50").getAsString()));
```

```
            }

            return list;
        }
        catch (Exception e)
        {
            System.out.println("Failed: parseFriendsJson");
            e.printStackTrace();
        }
        return null;
    }

    public ArrayList<Integer> getCommonFriends(int srcId, int[] targetIds)
    {
        ArrayList<VKUser> srcUser = getFriends(srcId, basicArgs);
        ArrayList<Integer> cmnFriends = new ArrayList<Integer>();
        for(int it : targetIds)
        {
            ArrayList<VKUser> list = getFriends(it,  basicArgs);
            int count = 0;
            for(VKUser i : list)
                if(srcUser.contains(i))
                    count++;
            int id = count;
            cmnFriends.add(id);
        }
        return cmnFriends;
    }

    public VKUser getUser(String userId, String[] args) throws
InvalidParameterException
    {
        if(userId == null || userId.equals(""))
        {
            throw new InvalidParameterException("Invalid input, null or empty
input.");
        }
        int id = -1;
        if(userId.startsWith("https://vk.com/"))
        {
            try {
```

```
                String str = userId.substring(15);
                id = Integer.parseInt(str);
            } catch (Exception e)
            {
                id = getUserId(userId);
            }
        }
        else
            id = Integer.parseInt(userId);


        String sb = createGetRequest("users.get?user_ids=", id, args);
        String request = getRequest(sb);

        JsonParser jsonParser = new JsonParser();
        JsonObject jo = (JsonObject)jsonParser.parse(request);

        if (jo.get("error") != null)
        {
            System.out.println("Error code: " +
jo.get("error").getAsJsonObject().get("error_code").getAsInt()
                    + "\ndescription: " +

jo.get("error").getAsJsonObject().get("error_msg").getAsString());
            return null;
        }
        //System.out.println(request);
        JsonObject response =
jo.get("response").getAsJsonArray().get(0).getAsJsonObject();

        if(response.get("is_closed").getAsString().equals("true"))
            throw new InvalidParameterException("Invalid input, private/deleted
profile.");

        VKUser user = new VKUser(response.get("id").getAsInt(),
                response.get("first_name").getAsString(),
                response.get("last_name").getAsString(),
                response.get("photo_50").getAsString());

        try { user.sex = response.get("sex").getAsInt(); } catch (Exception e) {
//System.out.println(e.getMessage());
```

```
        }
        try { user.relation = response.get("relation").getAsInt(); } catch
(Exception e) {//System.out.println(e.getMessage());
        }
        try { user.education = response.get("university_name").getAsString(); }
catch (Exception e) {//System.out.println(e.getMessage());
        }
        try { user.bdate = response.get("bdate").getAsString(); } catch (Exception
e) {//System.out.println(e.getMessage());
        }

        return user;
    }

    public String getUserFriends(int userId, String[] args)
    {
        return getRequest(createGetRequest("friends.get?user_id=", userId, args));
    }

    public ArrayList<VKUser> getFriends( int userId, String[] args )
    {
        String res = getUserFriends(userId, args);
        return parseFriendsJson(res);
    }

    private String createGetRequest(String method, int id, String[] args)
    {
        StringBuilder sb = new StringBuilder();
        sb.append(beginVkApi + method + id);
        if(args != null) {
            sb.append("&fields=");
            for (String field : args)
                sb.append(field + ",");
            sb.deleteCharAt(sb.length()-1);
        }
        sb.append(endVkApi);

        return sb.toString();
    }

    public int getUserId(String url)
```

```
    {
        String account = url.substring(15);
        String response = getRequest(beginVkApi +
"utils.resolveScreenName?screen_name=" + account + endVkApi);
        JsonParser jsonParser = new JsonParser();
        JsonObject obj = (JsonObject)jsonParser.parse(response);
        return obj.get("response").getAsJsonObject().get("object_id").getAsInt();
    }

    private String getRequest(String url)
    {
        StringBuilder sb = new StringBuilder();
        try {
            URL requestUrl=new URL(url);
            URLConnection con = requestUrl.openConnection();
            BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()));
            int cp;
            while ((cp = in.read()) != -1) {
                sb.append((char) cp);
            }
        }
        catch(Exception e) {
            System.out.println("Failed: getRequest");
        }
        return sb.toString();
    }
}
```

Класс VKUser:

```
package VKClient;

public class VKUser {
    public int userId;
    public String firstName;
    public String lastName;
    public String photoUrl;
    public int sex;
    public String bdate;
    public int relation;
    public String education;
```

```
public VKUser(
        int userId,
        String firstName,
        String lastName,
        String url)
{
    this.userId = userId;
    this.firstName = firstName;
    this.lastName = lastName;
    this.photoUrl = url;
    sex = 0;
    bdate = "-";
    relation = 0;
    education = "-";
}

@Override
public String toString() {
    return firstName + " " + lastName + " " + userId;
}

@Override
public boolean equals(Object obj) {
    VKUser tmp = (VKUser)obj;
    if (tmp.userId == this.userId)
        return true;
    return false;
}

public String getSex() {
    if(sex == 2)
        return "man";
    else if(sex == 1)
        return "woman";
    else
        return "transhomosek";
}

public String getRelation()
{
    switch (relation)
```

```
        {
            case 1:
                return "not married";
            case 2:
                return "boyfriend/girlfriend";
            case 3:
                return "engaged";
            case 4:
                return "married";
            case 5:
                return "complicated";
            case 6:
                return "in binary search";
            case 7:
                return "still loving you";
            default:
                return "superunknown";
        }
    }

    public String getInfo()
    {
        StringBuilder sb = new StringBuilder();
        sb.append("ID: " + String.valueOf(userId) + "\n");
        sb.append("First Name: " + firstName + "\n");
        sb.append("Last Name: " + lastName + "\n");
        sb.append("Birth Date: " + bdate + "\n");
        sb.append("Relations: " + getRelation() + "\n");
        sb.append("Sex: " + getSex() + "\n");
        sb.append("University: " + education + "\n");
        return sb.toString();
    }
}
```

Класс ApplicationMain:

```
import GUI.*;

import javax.swing.*;

public class ApplicationMain
{
    public static void main(String[] args)
```

```
    {
        try {
            new MainWindow();
        }
        catch (Exception e){
            JOptionPane.showMessageDialog(null,"Unknown error!");
        }

    }
}
```

Класс VKClientTest:

```
package VKClient;

import org.junit.Test;

import java.io.File;
import java.util.ArrayList;
import java.io.BufferedReader;
import java.io.FileReader;
public class VKClientTest {
    private static final String[] requestArgs = {"photo50", "education"};
    public static final String[] basicArgs = {"id", "first_name", "last_name",
"deactivated", "is_closed", "photo_50"};
    private int userID = 206043986;
    @Test
        public void parseFriendsJson() {
        VKClient user = new VKClient();
        String response = user.getUserFriends(userID, basicArgs);
        ArrayList<VKUser> list = new ArrayList<>();
        list = user.parseFriendsJson(response);
        try {
            File test = new File(new File("").getAbsolutePath()  +
"\\tests\\VKClient\\DataSets\\test1.txt");
            BufferedReader br;
            FileReader fileReader = new FileReader(test);
            br = new BufferedReader(fileReader);
            if (!test.exists()) {
                System.out.println("File doesn't exists");
                return;
            }
            String line;
```

```
            int j = 0;
            while((line = br.readLine()) != null){
                System.out.print(line);
                System.out.print(" - ");
                System.out.print(list.get(j).toString());
                System.out.print(" - ");
                if(list.get(j).toString().equals(line)){
                    System.out.println("equals");
                }
                else{
                    System.out.println("not equals");
                    return;
                }
                ++j;
            }
        }catch(Throwable e){
            System.out.println("ERROR" + e);
        }
    }

    @Test
    public void getUser() {
        VKClient user = new VKClient();
        userID = 206043986;
        String response = user.getUserFriends(userID, basicArgs);
        ArrayList<VKUser> list;
        list = user.parseFriendsJson(response);
        for(int i = 0; i< list.size();i++){
            try {
                VKUser tmp = list.get(i);
                String[] photo = {"photo_50"};
                System.out.print(user.getUser(Integer.toString(tmp.userId),
photo).toString() + " ");
            }catch (Throwable e){
                System.out.print("ERROR" + e);
            }
            System.out.println();

        }
    }
```

```
    @Test
    public void getCommonFriends() {
        try {
            userID = 206043986;
            VKClient user = new VKClient();
            File test = new File(new File("").getAbsolutePath()  +
"\\tests\\VKClient\\DataSets\\test2.txt");
            BufferedReader br;
            FileReader fileReader = new FileReader(test);
            br = new BufferedReader(fileReader);
            if (!test.exists()) {
                System.out.println("File doesn't exists");
                return;
            }
            ArrayList<String> id = new ArrayList<>();
            String str;
            while( (str = br.readLine()) != null){
                id.add(str);
            }
            int[] target = new int[id.size()];
            for(int i =0;i < id.size();i++) {
                target[i] = Integer.parseInt(id.get(i));
            }
            ArrayList<Integer> common = user.getCommonFriends(userID, target);
            String[] photo = {"photo_50"};
            for(int i = 0; i < common.size();i++){
                System.out.println(user.getUser(Integer.toString(target[i]),photo)
+ " - " + user.getUser(Integer.toString(userID),photo) + " in common: " +
common.get(i));
            }
        }
        catch(Throwable e) {
            System.out.println("ERROR: " + e);
        }
    }
}
```

ПРИЛОЖЕНИЕ В

КОД ОДНОГО ЮНИТ ТЕСТА ПРОГРАММЫ

```
      @Test
        public void parseFriendsJson() {
        VKClient user = new VKClient();
        String response = user.getUserFriends(userID, orderFriends, requestArgs);
        ArrayList<VKUser> list = new ArrayList<>();
        list = user.parseFriendsJson(response);
        try {
            File test = new File(new File("").getAbsolutePath()  +
"\\tests\\VKClient\\DataSets\\test1.txt");
            BufferedReader br;
            FileReader fileReader = new FileReader(test);
            br = new BufferedReader(fileReader);
            if (!test.exists()) {
                System.out.println("File doesn't exists");
                return;
            }
            String line;
            int j = 0;
            while((line = br.readLine()) != null){
                System.out.print(line);
                System.out.print(" - ");
                System.out.print(list.get(j).toString());
                System.out.print(" - ");
                if(list.get(j).toString().equals(line)){
                    System.out.println("equals");
                }
                else{
                    System.out.println("not equals");
                    return;
                }
                ++j;
            }
        }catch(Throwable e){
            System.out.println("ERROR" + e);
        }
```