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

ОТЧЕТ

по учебной практике

Тема: VK Friends

Студент гр. 7382 _____ Глазунов С.А.

Студентка гр. 7382 _____ Петрова А.С.

Студент гр. 7382 _____ Токарев А.П.

Руководитель _____ Фирсов М.А.

Санкт-Петербург
2019

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Глазунов С.А. группы 7382

Студентка Петрова А.С. группы 7382

Студент Токарев А.П. группы 7382

Тема практики: VK Friends

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.

Алгоритм: построение бора.

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета: 11.07.2019

Дата защиты отчета: 12.07.2019

Студент	_____	Глазунов С.А.
Студентка	_____	Петрова А.С.
Студент	_____	Токарев А.П.
Руководитель	_____	Фирсов М.А.

АННОТАЦИЯ

Целью данной практической работы является получение навыков программирования на языке java. Для этого предлагается реализовать проект, в котором с помощью встроенных библиотек будет реализован графический интерфейс, а также сделать визуализацию необходимого алгоритма. Кроме этого, написать unit-тестирование, которое будет в полной мере показывать правильность работы не только целого проекта, но и его отдельных частей, а также получить навыки работы в команде. В результате слаженной командной работы получить итоговый проект, который будет отвечать всем заявленным требованиям.

SUMMARY

The purpose of this practical work is to obtain programming skills in the java language. To do this, it is proposed to implement a project in which a graphical interface will be implemented using the built-in libraries, as well as to make the visualization of the necessary algorithm. In addition, write unit-testing, which will fully show the correctness of not only the whole project, but also its individual parts, as well as to gain skills in team work. As a result of well-coordinated teamwork to get the final project that will meet all the stated requirements.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Основной целью практики является получение навыков программирования на языке java. Необходимо реализовать проект, в котором с помощью встроенных библиотек будет реализован графический интерфейс, а также сделать визуализацию необходимого алгоритма. В данном проекте требуется визуализировать граф общих друзей в социальной сети ВКонтакте для заданного пользователем списка участников социальной сети. Вершины графа должны однозначно идентифицировать пользователя социальной сети.

Для построения графа был использован алгоритм построения бора по списку ID пользователей социальной сети ВКонтаке. Для улучшения восприятия графа друзей алгоритм построения бора был изменен. В корне «бора» находится выбранный пользователь, а от него идут ребра к друзьям, которые распределены по алфавиту (т.е. пользователи, имя которых начинается на одинаковую букву, находятся в одной ветви графа). В вершине хранится информация о пользователе: имя, фамилия и фотография.

Дополнительная информация о пользователе появляется при нажатии на пользователя.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Требования к входным и выходным данным

Исходными данными являются ID клиента в социальной сети ВКонтакте. При вводе некорректных данных выводится соответствующее сообщение об ошибке. Некорректными данными считаются несуществующий или удаленный ID, приватный профиль, отрицательные числа или числа, которые не помещаются в INT_MAX.

1.1.2. Требования к графическому интерфейсу

В окне графического интерфейса должна быть строка, в которую пользователь может ввести исходные данные, а также кнопка для построения графа друзей. Пример такого окна показан на рис.1.

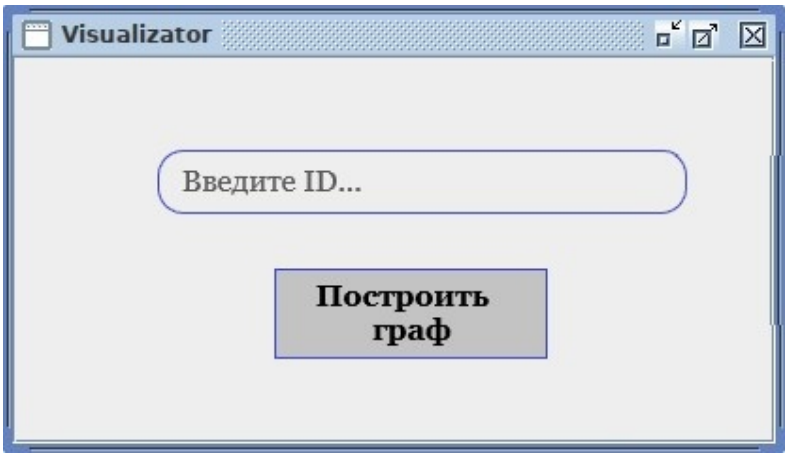

Рисунок 1 – Пример графического интерфейса.

1.1.3. Требования к оформлению вершины графа

Каждая вершина хранит данные о клиенте (имя и фамилию), а также фотографию пользователя из ВКонтакте. На рис.2 показано, как должен выглядеть примерный граф общих друзей.

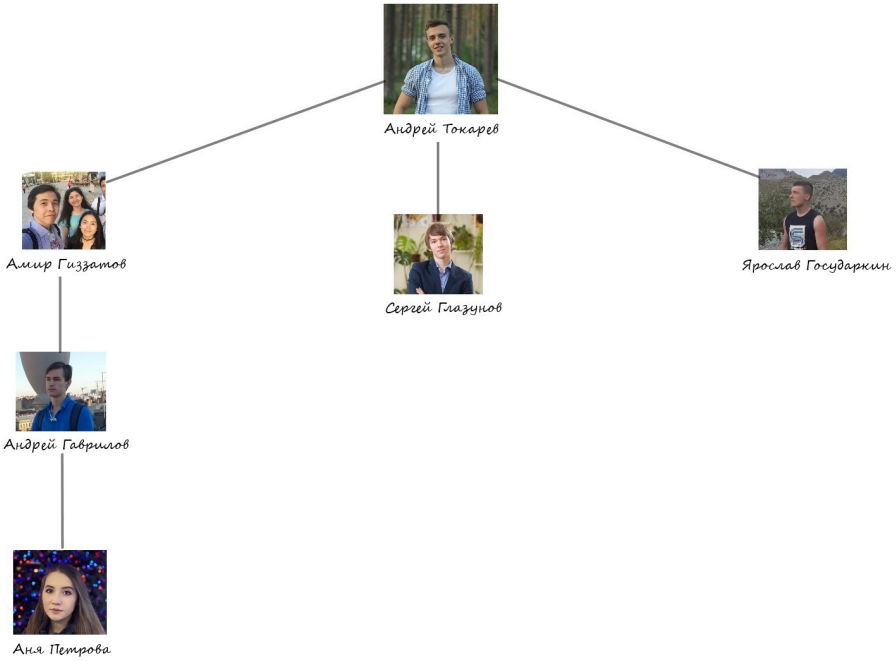

Рисунок 2 – Граф друзей.

1.2. Уточнение требований после сдачи прототипа

Получение графа друзей должно происходить не только по ID клиента, но также по ссылке на страницу в социальной сети ВКонтакте. Ссылки на любые другие страницы в интернете считаются некорректными.

При нажатии на любого пользователя из графа левой кнопкой мыши строится граф общих друзей изначального и данного пользователя. При нажатии на пользователя правой кнопкой мыши появляется окно с информацией о пользователе, а именно:

ID, имя, фамилия, пол, домен, дата рождения, онлайн/офлайн.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

- распределение ролей (3.07.2019)
- получение списка ID друзей выбранного пользователя(4.07.2019)
- разработка прототипа GUI (5.07.2019)
- поиск библиотеки для визуализации графа (5.07.2019)
- разработка начальной версии графа друзей (6.07.2019)
- поиск библиотеки для тестирования (6.07.2019)
- написание алгоритма построения бора по списку ID друзей (6.07.2019)
- реализация диалогового окна (7.07.2019)
- написание UML-диаграммы (7.07.2019)
- первичная сборка проекта (8.07.2019)
- тестирование правильности построения списка ID (8.07.2019)
- тестирование правильности построения бора (9.07.2019)
- *исправление ошибок*
- добавление дополнительных опций (11.07.2019)
- тестирование дополнительных опций (11.07.2019)
- окончательная сборка проекта (12.07.2019)

2.2. Распределение ролей в бригаде

1. Глазунов Сергей:
    - Реализация работы с web-сервисом VkAPI;
    - Внутренняя логика программы.
2. Петрова Анна:
    - Написание Unit-тестов;
    - Написание отчетов;

- Написание алгоритма для построения графа.

3. Токарев Андрей:
    - Разработка графического интерфейса;
    - Визуализация графа.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Использованные структуры данных.

Массивы списков, списки смежности, пользовательская структура данных – класс VKUser, в котором хранится полученная из VKAPI информация о пользователе. Основные поля:

int userId – ID пользователя;

String firstName – имя пользователя;

String lastName – фамилия пользователя;

String urlImage_50 – ссылка на фотографию пользователя.

3.2. Алгоритм построения графа друзей.

Данный граф строится по лексикографически отсортированному списку друзей клиента.

1. Корнем графа является пользователь, чей ID был получен на вход.
2. Если буквы, на которую начинается имя друга еще не было, то добавить ребро из корня к вершине друга. Каждая вершина хранит имя и фамилию друга.
3. Ребро к следующему другу будет вести из вершины с одинаковой начальной буквой или из корня. Т.к. список отсортирован в лексикографическом порядке, то каждая следующая вершина будет располагаться под предыдущей.
4. Граф считается построенным, если в списке не осталось необработанных вершин.

3.3. Основные методы

Программа состоит из следующих классов: Main, VkAPI, VKUser, JsonConversation, Visualizator, VerticalLayout.

Класс Main – главный класс программы, который отвечает за запуск программы. В табл.1 представлены основные методы данного и всех последующих классов.

Класс VkAPI – класс, в котором происходит взаимодействие с API социальной сети ВКонтакте.

Класс JsonConversation – класс, который преобразует данные из Json структуры. Реализует паттерн Адаптер.

Класс VkFriendsVisualization – класс, который реализует графический интерфейс проекта. Также данный класс реализует построение графа по списку полученных ID.

Класс VerticalLayout – класс, который размещает панели по вертикали.

Таблица 1 – Методы написанных классов.

Класс	Название метода	Функционал
VkAPI	public ArrayList<VKUser> parseFriendsJson(String str)	Функция, которая преобразует полученную информацию в список полученных друзей
	public String getUserFriends(int userId, String order, String[] args)	Функция, которая возвращает друзей пользователя
	private VKUser getUser(int userId, String[] args)	Функция, которая возвращает информацию о пользователе
	private String createGetRequest(String method, int id, String order, String[] args)	Функция, которая создает запрос для получения информации о пользователе
	private String getRequest(String url)	Функция, которая отправляет запрос для получения информации
JsonConversation	JsonArray getItemsFromResponse(JsonObject obj)	Функция, которая возвращает ту информацию о пользователе, которая была запрошена
	int getObjectId(JsonObject obj)	Функция, которая возвращает ID клиента
VkFriendsVisualization	public JPanel initInterface ()	Функция для инициализации панели интерфейса
	public void initCommonFriendsGraph()	Функция, которая создает граф общих друзей
	public void clearPanel(Component comp)	Функция очищения панели

На рис.3 представлена UML-диаграмма проекта, отражающая иерархию классов.

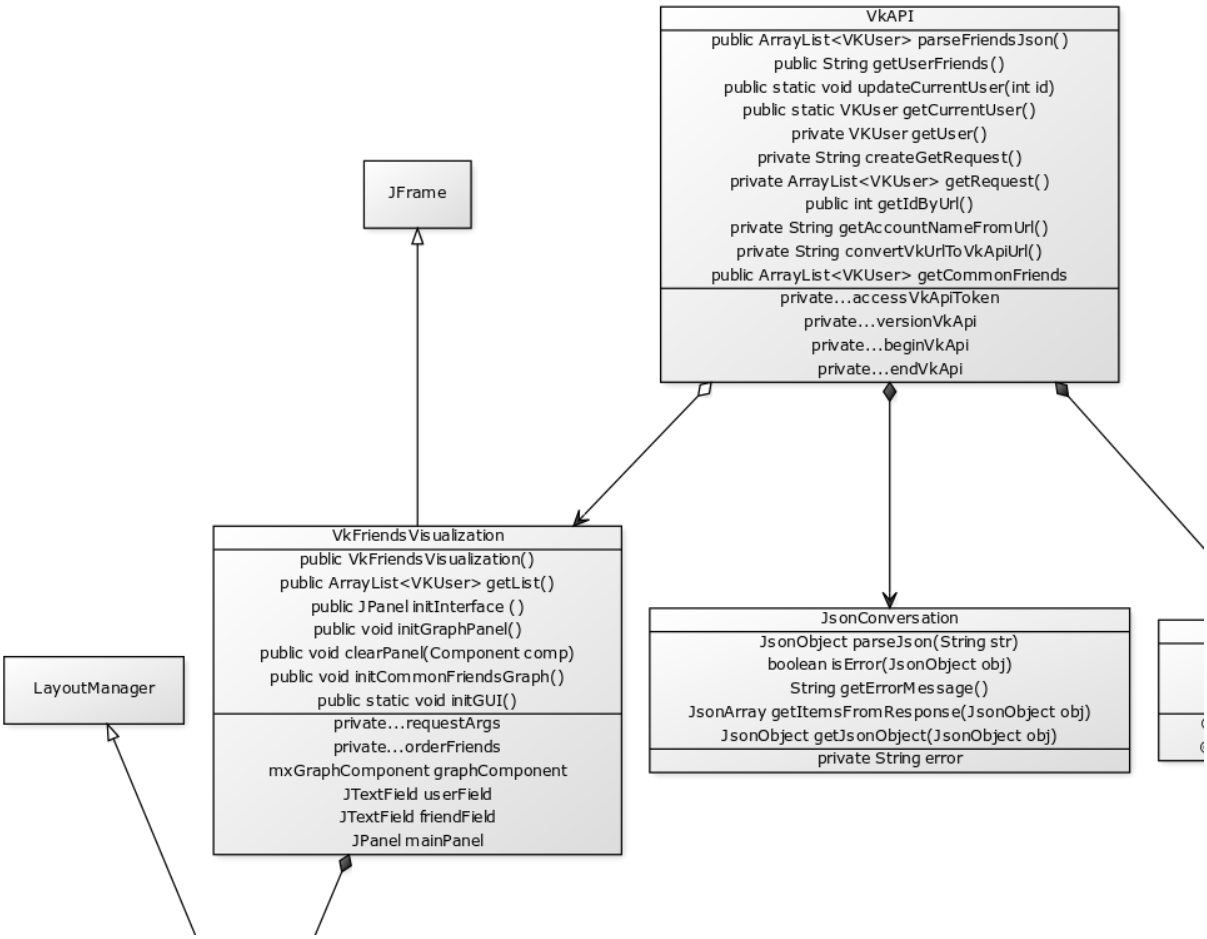

Рисунок 3 – UML.

4. ТЕСТИРОВАНИЕ

4.1. Unit-тестирование.

Для тестирования правильности работы были написаны unit-тесты. Для написания данных видов тестирования была использована библиотека Junit 4. Данная библиотека значительно упрощает тестирование кода алгоритма.

Для класса VkAPI были написаны три вида тестов. Так как данный класс реализует взаимодействие с сервером социальной сети и обработку полученной информации, то каждый из тестов проверяет корректность работы одного или нескольких методов класса.

Проверки были реализованы с помощью класса Assert. Данный класс выполняет ряд проверок, использованных в программе:

1. assertEquals – проверяет равенство полученных объектов. Чтобы данный метод работал корректно в рамках проекта он был перегружен.
2. assertNotNull – проверяет объект на пустоту. Если полученный объект был пустым, то данный метод выбрасывает исключение, которое обрабатывает программа.

Unit-тестирование к проекту представлено в Приложении Б.

4.2. Ручное тестирование.

Ручное тестирование кода проводилось для выявления слабых мест программы, а также для того, чтобы посмотреть, как выглядит итоговый графический интерфейс и реализованный граф друзей. Ниже, на рисунках 4-8 представлены результаты такого тестирования.

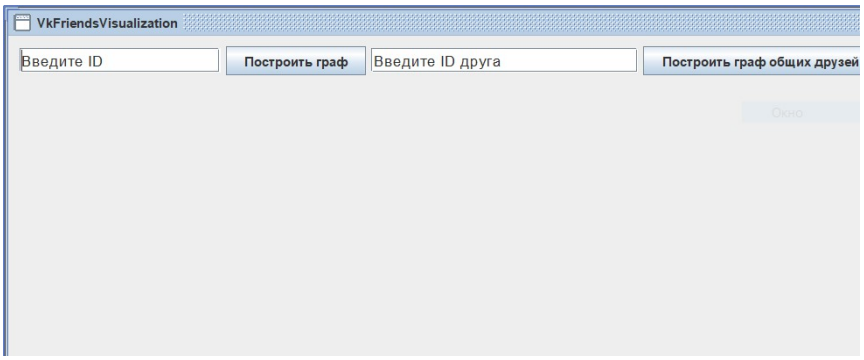

Рисунок 4 – Начальное окно GUI.

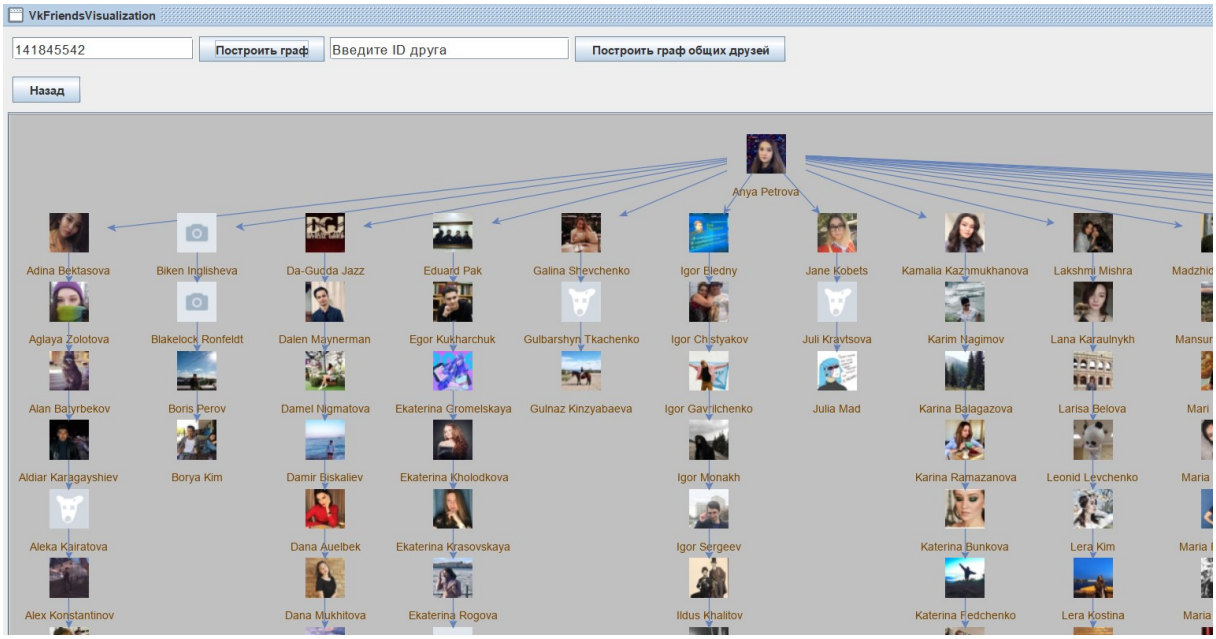

Рисунок 5 – Построенный граф друзей по ID.

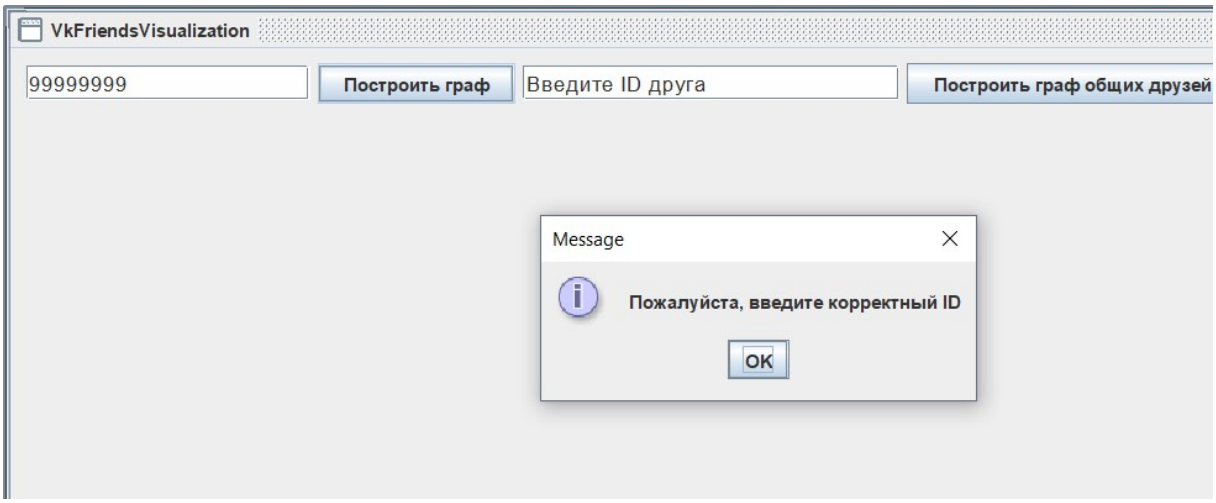

Рисунок 6 – Введен некорректный ID.

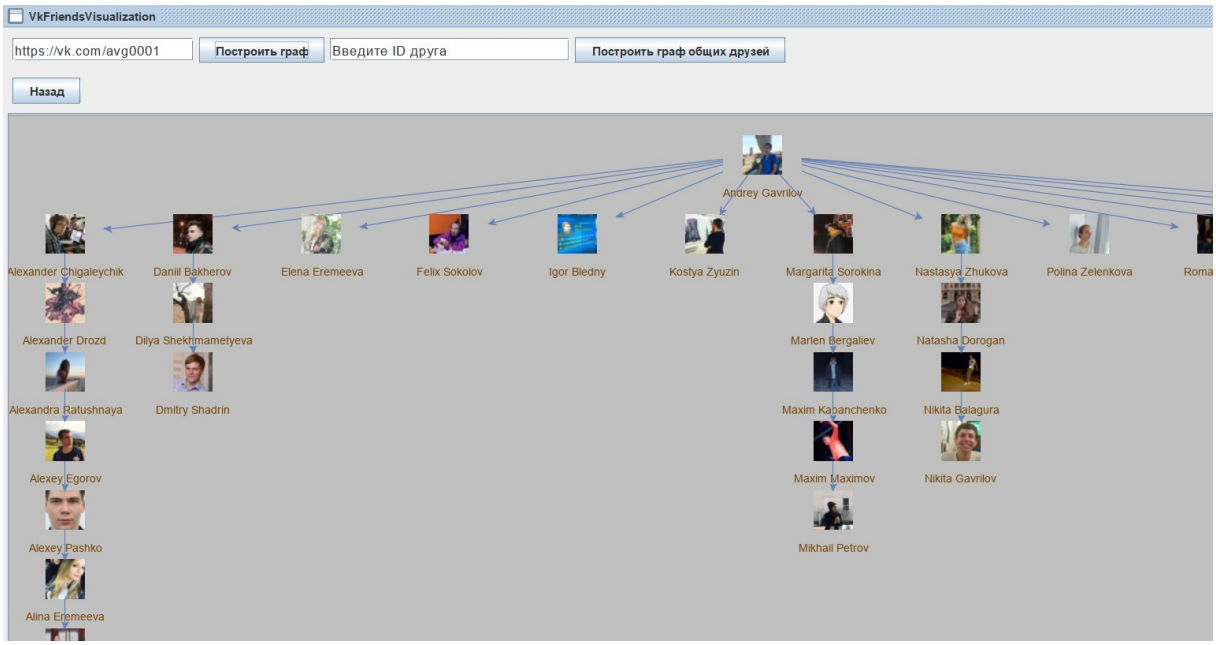

Рисунок 7 – Граф, построенный по url-ссылке.

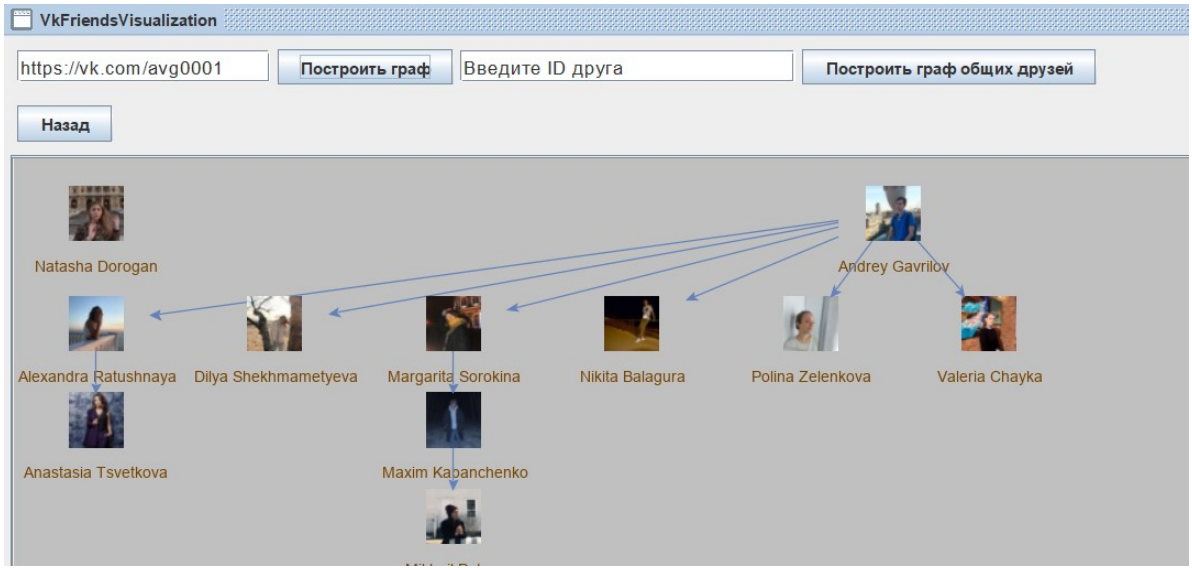

Рисунок 8 – Граф общих друзей.

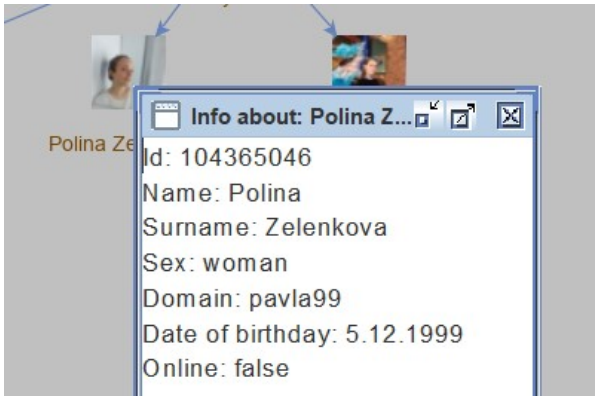

Рисунок 9 – Информация о пользователе.

ЗАКЛЮЧЕНИЕ

Разработка данного проекта была выполнена в соответствии с планом. Было написано приложение, которое при вводе ID или url-ссылки на страницу пользователя социальной сети ВКонтакте выводит граф друзей пользователя. Приложение использует реальные данные, которые хранятся на серверах разработчиков данной социальной сети. Был разработан графический интерфейс, который визуально отражает результаты работы и позволяет управлять возможностями приложения.

Написанные unit-тесты помогли выявить недочеты в реализованных алгоритмах получения информации о пользователе и построения графа. Графический интерфейс тестировался вручную.

В приложение были добавлены дополнительные функции:

1. Построение графа общих друзей по нажатию левой кнопки мыши, а также при введении ID пользователя или url-ссылки на его страницу.
2. При нажатии на пользователя правой кнопкой мыши появляется окно с дополнительной информацией о пользователе.

Таким образом, готовое приложение соответствует всем заявленным требованиям, реализует дополнительный функционал, а план разработки выполнен полностью.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальная документация к языку программирования Java: https://docs.oracle.com/javase/7/docs/
2. Сайт для работы с сервисами VKAPI: https://vk.com/dev/
3. Учебный курс по основам Java: https://stepik.org/course/187/syllabus
4. Дополнительные библиотеки Maven
5. https://habr.com/ru
6. http://www.javable.com
7. https://ru.stackoverflow.com

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД
MAIN CLASS

```
package Main;

import VkApi.VkAPI;
import VkApi.VKUser;

import Visualizator.*;

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

public class Main {

    private static final String[] requestArgs = {"photo_50", "education"};
    private static final String orderFriends = "name";
    public static void main(String[] args) {
        JFrame.setDefaultLookAndFeelDecorated(true);
        Visualizator.initGUI();
    }

    public static ArrayList<VKUser> getList() {
        VkAPI vk = new VkAPI();

        int userID = 179878269;
        VkAPI.updateCurrentUser(userID);
        ArrayList<VKUser> list = vk.getFriends(VkAPI.getCurrentUser().userId, orderFriends, requestArgs);

        int id = vk.getIdByUrl("https://vk.com/consolewritesj");
        System.out.println(id);

        var friendsAnnaSergey = vk.getCommonFriends(VkAPI.getCurrentUser().userId, 141845542);
        for (var i : friendsAnnaSergey)
            System.out.println(i.toString());

        return list;
    }
}
```

VKAPI

```
package VkApi;

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

import com.google.gson.*;

public class VkAPI {
```

```
    static int currentUserId;
    static VKUser currentVkUser;
    private final String accessVkApiToken =
"0c91fdfc0c91fdfc0c91fdfc3e0cfa8e5100c910c91fdfc518a61b257703bf5b0589264";
    private final String versionVkApi = "5.101";
    private final String beginVkApi = "https://api.vk.com/method/";
    private final String endVkApi = "&access_token=" + accessVkApiToken + "&v=" + versionVkApi;
    private final int lengthVkDomen = 15;
    private JsonConversation conversationJson;

    public VkAPI(){
        conversationJson = new JsonConversation();
        currentUserId = 1;
    }

    public static void updateCurrentUser(int id)
    {
        currentUserId = id;
        VkAPI vk = new VkAPI();
        currentVkUser = vk.getUser(id, null);
    }

    public static VKUser getCurrentUser()
    {
        return currentVkUser;
    }

    public ArrayList<VKUser> parseFriendsJson(String str) {
        try
        {
            JsonObject jo = conversationJson.parseJson(str);
            if (conversationJson.isError(jo))
            {
                System.out.println(conversationJson.getErrorMessage());
                return null;
            }

            JsonArray jsonArr = conversationJson.getItemsFromResponse(jo);
            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 String getUserFriends(int userId,  String order, String[] args)
{
    return getRequest(createGetRequest("friends.get?user_id=", userId, order, args));
}

private VKUser getUser(int userId, String[] args)
{
    if (args == null)
    {
        args = new String[]{"photo_50", "education"};
    }

    String request = getRequest(createGetRequest("users.get?user_ids=", userId,null, args));
    JsonObject jo = conversationJson.parseJson(request);

    if (conversationJson.isError(jo))
    {
        System.out.println(conversationJson.getErrorMessage());
        return null;
    }

    JsonObject response = conversationJson.getJsonObject(jo);

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

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

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();
}

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

public int getIdByUrl(String url)
{
    String account = getAccountNameFromUrl(url);
    if (account != null) {
        var request = getRequest(convertVkUrlToVkApiUrl(account));
        return conversationJson.getObjectId(conversationJson.parseJson(request));
    }
    return 0;
}

private String getAccountNameFromUrl(String url)
{
    var newStr = url.substring(lengthVkDomen);
    if (newStr.contains("/")||newStr.contains("?")||newStr.contains("&"))
        return null;
    return newStr;
}

private String convertVkUrlToVkApiUrl(String accountName)
{
    return beginVkApi + "utils.resolveScreenName?screen_name=" + accountName + endVkApi;
}

public ArrayList<VKUser> getCommonFriends(int firstId, int secondId)
{
    var first = getFriends(firstId, "name", new String[]{"photo_50"});
    var second = getFriends(secondId,"name", new String[]{"photo_50"});
    ArrayList<VKUser> commonFriends = new ArrayList<>();
    for (var i : second) {
        if (first.contains(i)) {
            commonFriends.add(i);
        }
   }
   return commonFriends;

}
                                                    }
```

VKUSER

```
package VkApi;

import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

class JsonConversation {

   private String error;

   JsonObject parseJson(String str)
   {
      JsonParser jsonParser = new JsonParser();
      return  (JsonObject)jsonParser.parse(str);
   }

   boolean isError(JsonObject obj)
   {
      if (obj.get("error") != null)
      {
          error = "Error code: " + obj.get("error").getAsJsonObject().get("error_code").getAsInt()
                + "\ndescription: " +
                obj.get("error").getAsJsonObject().get("error_msg").getAsString();
         return true;
      }
      return false;
   }

   String getErrorMessage()
   {
      return error;
   }

   JsonArray getItemsFromResponse(JsonObject obj)
   {
      return obj.get("response").getAsJsonObject().getAsJsonArray("items");
   }

   JsonObject getJsonObject(JsonObject obj)
   {
      return obj.get("response").getAsJsonArray().get(0).getAsJsonObject();
   }

   int getObjectId(JsonObject obj)
   {
      return obj.get("response").getAsJsonObject().get("object_id").getAsInt();
   }
}
```

JSONCONVERSATION

```
package VkApi;

import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

class JsonConversation {
```

```
    private String error;

    JsonObject parseJson(String str)
    {
        JsonParser jsonParser = new JsonParser();
        return  (JsonObject)jsonParser.parse(str);
    }

    boolean isError(JsonObject obj)
    {
        if (obj.get("error") != null)
        {
            error = "Error code: " + obj.get("error").getAsJsonObject().get("error_code").getAsInt()
                    + "\ndescription: " +
                    obj.get("error").getAsJsonObject().get("error_msg").getAsString();
            return true;
        }
        return false;
    }

    String getErrorMessage()
    {
        return error;
    }

    JsonArray getItemsFromResponse(JsonObject obj)
    {
        return obj.get("response").getAsJsonObject().getAsJsonArray("items");
    }

    JsonObject getJsonObject(JsonObject obj)
    {
        return obj.get("response").getAsJsonArray().get(0).getAsJsonObject();
    }

    int getObjectId(JsonObject obj)
    {
        return obj.get("response").getAsJsonObject().get("object_id").getAsInt();
    }
}
```

VKFRIENDSVISUALIZATION

```
package VkFriendsVisualization;

import javax.swing.*;
import java.awt.*;
import com.mxgraph.swing.mxGraphComponent;
import com.mxgraph.model.mxCell;
import com.mxgraph.view.mxGraph;
import java.util.ArrayList;
import VkApi.VkAPI;
import VkApi.VKUser;
import java.awt.Font;
import java.awt.event.*;

public class VkFriendsVisualization extends JFrame {
```

```
private static final String[] requestArgs = {"photo_50", "education"};
private static final String orderFriends = "name";
static int userID;
static int friendID;
//final int userID = 179878269;
final int xRoot = 600, yRoot = 20, xBorder = 20;
mxGraphComponent graphComponent;
JTextField userField;
JTextField friendField;
JPanel mainPanel;
JScrollPane scrollPane = new JScrollPane(graphComponent);
JScrollPane scrollPanel;
JPanel checkPanel = new JPanel();
JPanel exitPanel = new JPanel();
VkAPI vk = new VkAPI();

public VkFriendsVisualization() {

   super("VkFriendsVisualization"); //call jframe constructor

   setDefaultCloseOperation(EXIT_ON_CLOSE);
   mainPanel = new JPanel(new VerticalLayout());
   JPanel interfacePanel = initInterface();
   mainPanel.add(interfacePanel);
   setContentPane(mainPanel);
   setPreferredSize(new Dimension(980,400));
   setLocation(200, 200);
   pack();
   setVisible(true);

}

public JPanel initInterface (){

   userField = new JTextField("Введите ID", 15);
   userField.setToolTipText("Введите ID");
   userField.setFont(new Font("Dialog", Font.PLAIN, 14));
   userField.setHorizontalAlignment(JTextField.LEFT);

   friendField = new JTextField("Введите ID друга", 20);
   friendField.setToolTipText("Введите ID друга");
   friendField.setFont(new Font("Dialog", Font.PLAIN, 14));
   friendField.setHorizontalAlignment(JTextField.LEFT);

   JPanel interfacePanel = new JPanel(new FlowLayout());
   interfacePanel.add(userField);

   JButton buildButton = new JButton("Построить граф");
   buildButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
         try {
            try {
               userID = Integer.parseInt(userField.getText());
            } catch (Exception ex) {
               userID = vk.getIdByUrl(userField.getText());
```

```
                }
                clearPanel(scrollPane);
                clearPanel(exitPanel);
                clearPanel(checkPanel);
                initGraphPanel();
                mainPanel.updateUI();
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(VkFriendsVisualization.this,"Пожалуйста, введите
корректный ID");
            }
        }
    });

    JButton buildCommonButton = new JButton("Построить граф общих друзей");
    buildCommonButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                try {
                    friendID = Integer.parseInt(friendField.getText());
                } catch (Exception ex) {
                    friendID = vk.getIdByUrl(friendField.getText());
                }
                clearPanel(scrollPane);
                clearPanel(scrollPanel);
                clearPanel(exitPanel);
                clearPanel(checkPanel);
                initCommonFriendsGraph();
                mainPanel.updateUI();
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(VkFriendsVisualization.this,"Пожалуйста, введите
корректный ID");
            }
        }
    });

    interfacePanel.add(buildButton);
    interfacePanel.add(friendField);
    interfacePanel.add(buildCommonButton);
    return interfacePanel;
  }


  public void initGraphPanel () {

    mxGraph friendsGraph = new mxGraph();
    Object parent = friendsGraph.getDefaultParent();
    friendsGraph.getModel().beginUpdate();
    Dimension sSize = Toolkit.getDefaultToolkit ().getScreenSize ();
    sSize.width-=80;
    sSize.height-=180;
    try {
        VkAPI.updateCurrentUser(userID);
        String id = ""+userID;
        ArrayList<VKUser> friendList = vk.getFriends(VkAPI.getCurrentUser().userId, orderFriends,
requestArgs);
        VKUser currentUser = VkAPI.getCurrentUser();
```

```
        ArrayList vertexes = new ArrayList();
        String in = currentUser.firstName + " " + currentUser.lastName;
        String ava = "shape=image;image="+currentUser.urlImage_50+";verticalLabelPosition=bottom";
        vertexes.add(friendsGraph.insertVertex(parent, id, in, sSize.width/2, yRoot, 80, 40, ava));
        int y = -1, x = -1;
        for (int i = 0; i < friendList.size(); ) {
            mxCell c = (mxCell) vertexes.get(i);
            id = ""+friendList.get(i).userId;
            in = friendList.get(i).firstName + " " + friendList.get(i).lastName;
            ava = "shape=image;image="+friendList.get(i).urlImage_50+";verticalLabelPosition=bottom";
            if (c.getValue().toString().charAt(0) == in.charAt(0)) {
                if (i == 0) x++;
                vertexes.add(friendsGraph.insertVertex(parent, id, in,  xBorder + x * 130, yRoot + 80 + (y + 1)
* 70, 80, 40, ava));
                i++;
                y++;
                friendsGraph.insertEdge(parent, null, "", vertexes.get(i - 1), vertexes.get(i));
            } else {
                y = 0;
                x++;
                vertexes.add(friendsGraph.insertVertex(parent, id, in, xBorder + x * 130, yRoot + 80 + y * 70,
80, 40, ava));
                i++;
                friendsGraph.insertEdge(parent, null, "", vertexes.get(0), vertexes.get(i));
            }
        }
    } finally {
        friendsGraph.getModel().endUpdate();
    }
    graphComponent = new mxGraphComponent(friendsGraph);
    graphComponent.getViewport().setOpaque(true);
    Color newColor = new Color(192, 192, 192);
    graphComponent.getViewport().setBackground(newColor);
    graphComponent.getGraphControl().addMouseListener(new MouseAdapter() {
        @Override
        public void mousePressed(MouseEvent e) {
            mxCell cell = (mxCell)graphComponent.getCellAt(e.getX(), e.getY());
            if (cell != null && cell.isEdge() == false) {

                friendID = Integer.parseInt(cell.getId());
                if (e.getButton() == MouseEvent.BUTTON1) {
                    try {
                        clearPanel(scrollPane);
                        clearPanel(exitPanel);
                        clearPanel(checkPanel);
                        initCommonFriendsGraph();
                    } catch (Exception ex) {
                        mainPanel.add(exitPanel);
                        mainPanel.add(scrollPane);
                        mainPanel.add(checkPanel);
                        JOptionPane.showMessageDialog(VkFriendsVisualization.this, "Невозможно построить
граф для данного пользователя");
                    }
                    mainPanel.updateUI();
                }
                else if (e.getButton() == MouseEvent.BUTTON3)
                {
```

```
                VkAPI.updateCurrentUser(friendID);
                VKUser curInfo = VkAPI.getCurrentUser();
                JFrame info = new JFrame("Info about: " + curInfo.firstName + " " + curInfo.lastName);
                JTextArea textArea = new JTextArea(curInfo.toString());
                textArea.setFont(new Font("Dialog", Font.PLAIN, 14));
                textArea.setTabSize(10);
                info.setContentPane(textArea);
                info.setPreferredSize(new Dimension(220,160));
                info.setLocation(e.getXOnScreen(), e.getYOnScreen());
                info.pack();
                info.setVisible(true);
            }
        }
    }
});
graphComponent.setEnabled(false);
scrollPane = new JScrollPane(graphComponent);
checkPanel = new JPanel();
scrollPane.setPreferredSize(sSize);
scrollPane.setWheelScrollingEnabled(true);
exitPanel = new JPanel();
JButton exitButton = new JButton("Назад");
exitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        clearPanel(scrollPane);
        clearPanel(exitPanel);
        clearPanel(checkPanel);
    }
});
exitPanel.add(exitButton);
mainPanel.add(exitPanel);
mainPanel.add(scrollPane, BorderLayout.WEST);
final JCheckBox checkBox1 = new JCheckBox("Show vertical scrollbar");
checkBox1.setSelected(true);
checkBox1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (checkBox1.isSelected()) {
            scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
        } else {
            scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_NEVER);
        }
    }
});
checkPanel.add(checkBox1);

final JCheckBox checkBox2 = new JCheckBox("Show horizontal scrollbar");
checkBox2.setSelected(true);
checkBox2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (checkBox2.isSelected()) {

scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        } else {
            scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
        }
    }
```

```
        });
        checkPanel.add(checkBox2);

        mainPanel.add(checkPanel, BorderLayout.SOUTH);
    }

    public void clearPanel(Component comp){
        mainPanel.remove(comp);
        mainPanel.updateUI();
    }

    public void initCommonFriendsGraph(){

        mxGraph CommonfriendsGraph = new mxGraph();
        Object parent = CommonfriendsGraph.getDefaultParent();
        CommonfriendsGraph.getModel().beginUpdate();
        try {
            vk.updateCurrentUser(userID);
            VKUser currentUser = VkAPI.getCurrentUser();
            vk.updateCurrentUser(friendID);
            VKUser currentFriend = VkAPI.getCurrentUser();
            ArrayList vertexes = new ArrayList();
            String id = ""+friendID;
            String ava = "shape=image;image="+currentFriend.urlImage_50+";verticalLabelPosition=bottom;";
            String in = currentFriend.firstName + " " + currentFriend.lastName;
            vertexes.add(CommonfriendsGraph.insertVertex(parent, id, in, xBorder, yRoot, 80, 40, ava));
            ArrayList<VKUser> friendList = vk.getCommonFriends(userID, friendID);
            id = ""+userID;
            in = currentUser.firstName + " " + currentUser.lastName;
            ava = "shape=image;image="+currentUser.urlImage_50+";verticalLabelPosition=bottom";
            vertexes.add(CommonfriendsGraph.insertVertex(parent, id, in, xRoot, yRoot, 80, 40, ava));
            int y = -1, x = -1;
            for (int i = 1; i < friendList.size() + 1; ) {
                mxCell c = (mxCell) vertexes.get(i);
                id = ""+friendList.get(i-1).userId;
                in = friendList.get(i-1).firstName + " " + friendList.get(i-1).lastName;
                ava = "shape=image;image="+friendList.get(i-1).urlImage_50+";verticalLabelPosition=bottom";
                if (c.getValue().toString().charAt(0) == in.charAt(0)) {
                    if (i == 1) x++;
                    vertexes.add(CommonfriendsGraph.insertVertex(parent, id, in,  xBorder + x * 130, yRoot + 80
+ (y + 1) * 70, 80, 40, ava));
                    i++;
                    y++;
                    CommonfriendsGraph.insertEdge(parent, null, "", vertexes.get(i - 1), vertexes.get(i));
                } else {
                    y = 0;
                    x++;
                    vertexes.add(CommonfriendsGraph.insertVertex(parent, id, in, xBorder + x * 130, yRoot + 80 +
y * 70, 80, 40, ava));
                    i++;
                    CommonfriendsGraph.insertEdge(parent, null, "", vertexes.get(1), vertexes.get(i));
                }
            }
        } finally {
            CommonfriendsGraph.getModel().endUpdate();
        }
        mxGraphComponent commonGraphComponent = new mxGraphComponent(CommonfriendsGraph);
```

```
commonGraphComponent.getViewport().setOpaque(true);
Color newColor = new Color(192, 192, 192);
commonGraphComponent.getViewport().setBackground(newColor);
commonGraphComponent.getGraphControl().addMouseListener(new MouseAdapter() {
   @Override
   public void mousePressed(MouseEvent ev) {
      mxCell cell = (mxCell) commonGraphComponent.getCellAt(ev.getX(), ev.getY());
      if (cell != null && cell.isEdge() == false) {

         int infoID = Integer.parseInt(cell.getId());

         if (ev.getButton() == MouseEvent.BUTTON3)
         {
            VkAPI.updateCurrentUser(infoID);
            VKUser curInfo = VkAPI.getCurrentUser();
            JFrame info = new JFrame("Info about: " + curInfo.firstName + " " + curInfo.lastName);
            JTextArea textArea = new JTextArea(curInfo.toString());
            textArea.setFont(new Font("Dialog", Font.PLAIN, 14));
            textArea.setTabSize(10);
            info.setContentPane(textArea);
            info.setPreferredSize(new Dimension(220,160));
            info.setLocation(ev.getXOnScreen(), ev.getYOnScreen());
            info.pack();
            info.setVisible(true);
         }
      }
   }
});
commonGraphComponent.setEnabled(false);
scrollPanel = new JScrollPane(commonGraphComponent);
checkPanel = new JPanel();
Dimension sSize = Toolkit.getDefaultToolkit ().getScreenSize ();
sSize.width-=80;
sSize.height-=180;
scrollPanel.setPreferredSize(sSize);
scrollPanel.setWheelScrollingEnabled(true);
exitPanel = new JPanel();
JButton exitButton = new JButton("Назад");
exitButton.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      clearPanel(scrollPanel);
      clearPanel(checkPanel);
      mainPanel.add(scrollPane);
      mainPanel.add(checkPanel);
      mainPanel.updateUI();
   }
});
exitPanel.add(exitButton);
mainPanel.add(exitPanel);
mainPanel.add(scrollPanel, BorderLayout.WEST);
final JCheckBox checkBox1 = new JCheckBox("Show vertical scrollbar");
checkBox1.setSelected(true);
checkBox1.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      if (checkBox1.isSelected()) {
         scrollPanel.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
```

```
            } else {
               scrollPanel.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_NEVER);
            }
         }
      });
      checkPanel.add(checkBox1);

      final JCheckBox checkBox2 = new JCheckBox("Show horizontal scrollbar");
      checkBox2.setSelected(true);
      checkBox2.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            if (checkBox2.isSelected()) {

scrollPanel.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
            } else {

scrollPanel.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
            }
         }
      });
      checkPanel.add(checkBox2);

      mainPanel.add(checkPanel, BorderLayout.SOUTH);
   }

   public static void initGUI(){
      VkFriendsVisualization frame = new VkFriendsVisualization();
   }
}
```

VERTICALLAYOUT

```
package Visualizator;

import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.LayoutManager;

class VerticalLayout implements LayoutManager
{
   private Dimension size = new Dimension();

   public void addLayoutComponent   (String name, Component comp) {}
   public void removeLayoutComponent(Component comp) {}

   public Dimension minimumLayoutSize(Container c) {
      return calculateBestSize(c);
   }

   public Dimension preferredLayoutSize(Container c) {
      return calculateBestSize(c);
   }

   public void layoutContainer(Container container)
   {
```

```
        Component list[] = container.getComponents();
        int currentY = 5;
        for (int i = 0; i < list.length; i++) {
            Dimension pref = list[i].getPreferredSize();
            list[i].setBounds(5, currentY, pref.width, pref.height);
            currentY += 5;
            currentY += pref.height;
        }
    }

    private Dimension calculateBestSize(Container c)
    {
        Component[] list = c.getComponents();
        int maxWidth = 0;
        for (int i = 0; i < list.length; i++) {
            int width = list[i].getWidth();
            if ( width > maxWidth )
                maxWidth = width;
        }
        size.width = maxWidth + 5;
        int height = 0;
        for (int i = 0; i < list.length; i++) {
            height += 5;
            height += list[i].getHeight();
        }
        size.height = height;
        return size;
    }
}
```

ПРИЛОЖЕНИЕ Б

UNIT-тестирование

```
package VkApi;

import org.junit.Assert;
import org.junit.Test;

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;

public class VkAPITest extends Assert{


   private static final String[]  requestArgs = {"photo_50"};
   private static final String orderFriends = "name";
   VkAPI test = new VkAPI();

   @Test //тест для начального этапа разработки
   public void parseFriendsJson() {
      int userID = 179878269;
      String response = test.getUserFriends(userID, orderFriends, requestArgs);
      ArrayList<VKUser> expected =  test.parseFriendsJson(response);
      String path = System.getProperty("user.dir")+("\\Test\\VkApi\\test1.txt");
      try(BufferedReader br = new BufferedReader(new FileReader(path))){
         String actual;
         int i=0;
         while((actual=br.readLine())!=null){
            Assert.assertEquals(expected.get(i).toString().replace(" ", ""),actual.replace(" ", ""));
            System.out.println(i+1 + ". " + expected.get(i).toString() + " " + actual + " EQUALS");
            i++;
         }
      }
      catch (Throwable e){

         System.out.println("Error"+e);
   }

   }

   @Test
   public void getUser(){
      String path = System.getProperty("user.dir")+("\\Test\\VkApi\\test2.txt");
      try(BufferedReader br = new BufferedReader(new FileReader(path))){
         String actual;
         int i=0;
         while((actual=br.readLine())!=null){
            int userID = Integer.parseInt(actual);
            //test.getUser(userID,requestArgs);
            Assert.assertNotNull(test.getUser(userID,requestArgs));
            System.out.println(test.getUser(userID,requestArgs));
            i++;
         }
      }
      catch (Throwable e){
```

```
            System.out.println("Error"+e);
        }
    }

    @Test
    public void getCommonFriends(){
        String path = System.getProperty("user.dir")+("\\Test\\VkApi\\test3.txt");
        int ID = 206043986;
        try(BufferedReader br = new BufferedReader(new FileReader(path))){
            String actual;
            int i=0;
            while((actual=br.readLine())!=null){
                int userID = Integer.parseInt(actual);
                 test.getCommonFriends(userID,ID);
                //Assert.assertNotNull(test.getUser(userID,requestArgs));
                System.out.println(test.getCommonFriends(userID,ID));
                i++;
            }
        }
        catch (Throwable e){
            System.out.println("Error"+e);
        }
    }
}
```