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 _____ Фиалковский 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: 7.07.2020
Дата защиты отчета: 7.07.2020

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

АННОТАЦИЯ

Целью работы является получение 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

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 рисунке 1

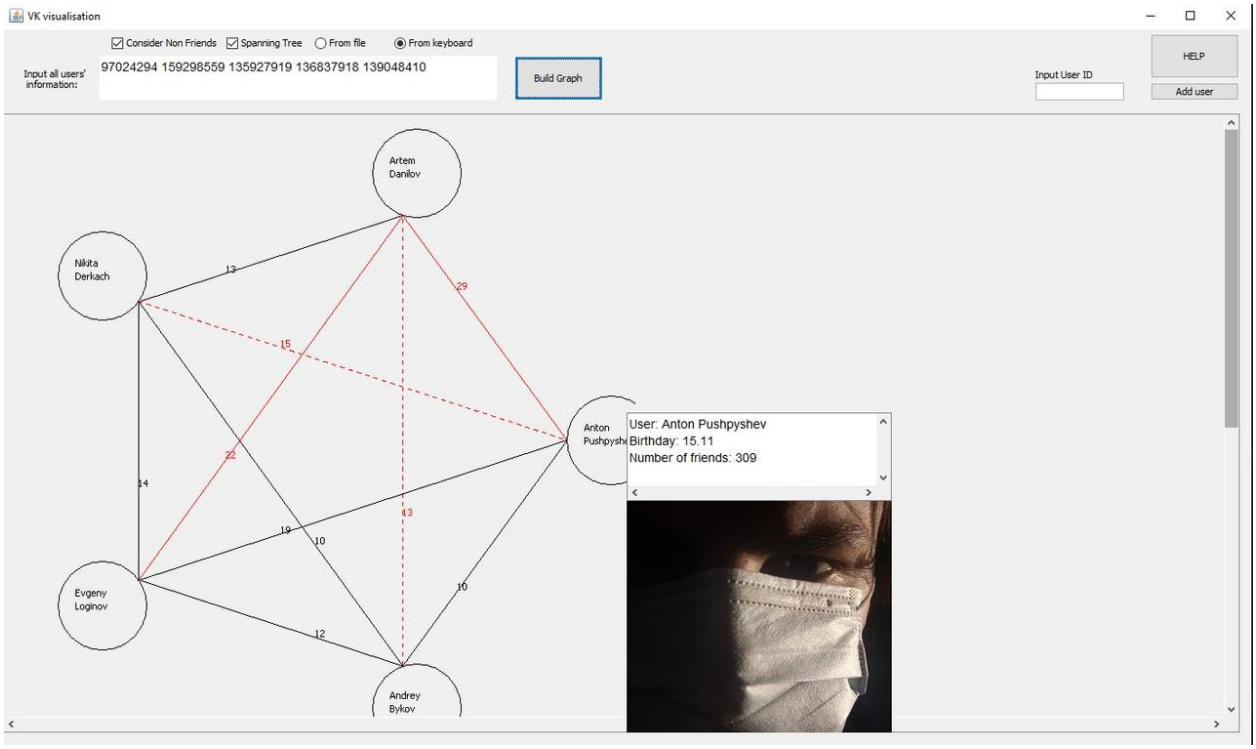

Рисунок 1 – прототип интерфейса.

Приложение имеет интерактивную область с графом. С вахней стороны располагаются следующие элементы управления:

- Поля для ввода информации, считывание информации

возможно с клавиатуры или из файла

- Поле для добавления нового пользователя
- Кнопка начала построения графа общих друзей
- Кнопка “Spanning Tree” предназначена для построения для построения минимального остового дерева по признаку наибольшего числа общих друзей
- Кнопка “Consider Non Friends” рисует пунктирные ребра между людьми которые не являются непосредственными друзьями.
- Кнопка “INFO” выводит информацию о данном приложении

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 ко 2 июля 2020 года.
2) Создать прототип интерфейса. Данный пункт задания необходимо выполнить ко 3 июля 2020 года.
3) Реализация структуры данных. Данный пункт задания необходимо выполнить ко 6 июля 2020 года.
4) Реализация работы с социальной сетью “ВКотакте” при помощи api, используя для этого сервисный ключ доступа. Выполнить к 7 июля 2020 года.
5) Первичная сборка проекта рабочего прототипа программы и первичное тестирование его функций. Выполнить к 8 июля 2020 года.
6) Полноценное тестирование программы. Выполнить к 11 июля 2020 года.

2.2 Распределение ролей в бригаде

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

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1 Использование структуры данных

Для реализации проекта потребовалось разработать следующие структуры данных:

Класс Graph предназначен для представления графа:

- Поля:
    - public UsersContainer users – доступ к полям класса контейнера пользователя.
    - public EdgesContainer edges – доступ к ребрам графа.
    - public final boolean nonFriends – реализация кнопки с соотвествующем именем;
    - public final boolean spanningTree - реализация кнопки с соотвествующем именем;
- Методы:
    - public Graph() – конструктор класса, инициализирует переменные.
    - 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:

- Поля:
  - public boolean isFriends – флаг, показывающий являются ли пользователи друзами.
  - public boolean isInSpanningTree - построение минимального остового дерева по признаку наибольшего числа общих друзей, если конечно нажата соответствующая кнопка.
  - public User user1, user2 – ссылка на двух соседних пользователей.
  - public int weight – хранение веса ребра.
  - public Cords cords – обхект класса с координатами.
- Методы:
  - public Edge() – конструктор класса, начальная инициализация переменных.
  - 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:

- Поля:
  - public Edge [] edges – перменная для досупа к полям класса Edge.
  - private int length – длина контейнера, кол-во ребер в контейнере.
- Методы:
  - public EdgesContainer() – конструктор класса, изначальная инициализация переменных.
  - public void addEdge(User u1, User u2) – добавление ребра между двумя друзямим.
  - public boolean find(Edge edge) – проверяет, входит ли ребро в контейнер.
  - public int buildEdges(int vertexNum) – определение кординат 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 и пользователе непосредственно из ВК:

- 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– присвоение переменным информации о пользователе.
    - static public String [] getUserFriends(String VkID) – получение списка ID друзей пользователя.
    - static public String [] getUserInfo(String VkID) – получение информации о пользователе.
    - public String getName() – получение имени.
    - public String getSurname() – получение фамилии.
    - public String getAge() – получение возвраста.
    - 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 пользователе:

- Поля:
    - public User [] users – массив пользователей.
    - private int length – длина контейнера.
- Методы:

- public UsersContainer() – конструктор класса, изначальная 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 Основные методы

Основные методы для работы были реализованы в классах User, который реализует запрос к ВКонтакте для считывания информации о пользователе при помощи api и класс Graph, отвечающий за построение графа и считывание информации из поля ввода.

Class User:

- 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.

Class Graph:

- public void addUser(String VkID) – добавление нового по id из ВК.
- private void buildGraph() – построение графа общих друзей.

- private void processText(String text) – считывание строки с ID пользователя и проверка на корректность введенных данных.

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 2.

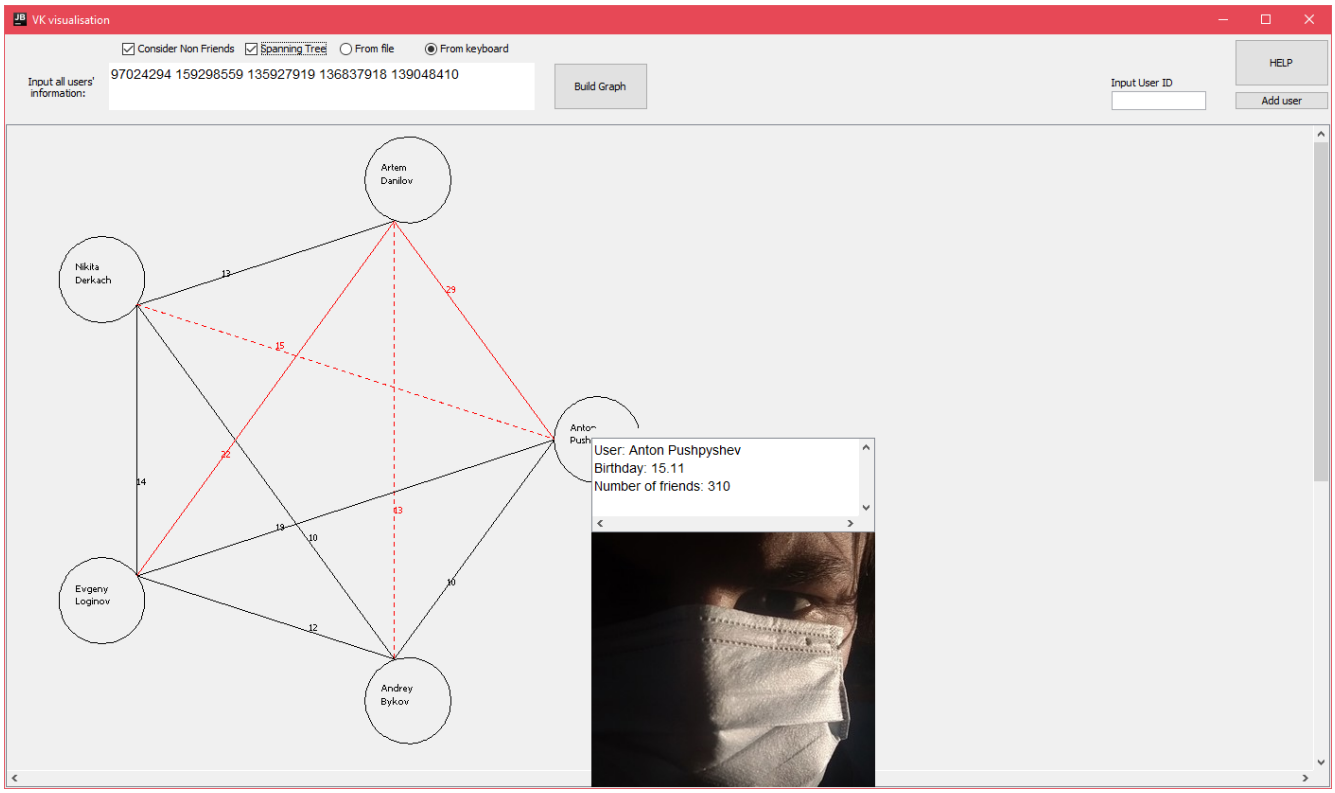

Рисунок 2 – пример ручного тестирования.

ЗАКЛЮЧЕНИЕ

Разработка поставленной задачи была выполнена в соответствии с планом. Было спроектировано и реализовано приложения “VK Clique” для визуализации общих друзей в виде графа. Также был разработан графический интерфейс, визуально отображающий результат работы программы.

В приложение была добавлена дополнительная функция, позволяющая просмотреть основную информацию о данном аккаунте при нажатии на вершину с пользователем.

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/