**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**КАФЕДРА САПР**

**ИДЗ**

**ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ KOTLIN»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 1301 |  | Чебесова И.Д.  Верхозина А.А. |
| Преподаватель |  | Кулагин М.В. |

Санкт-Петербург

2023

# Разработка приложения для взаимодействия с внешними источниками

# 1.1. Задание

Напишите программу, которая с консоли считывает поисковый запрос пользователя, и выводит результат поиска по Википедии. После выбора нужной статьи программа должна открывать ее в браузере. Программа должна реагировать корректно на любой пользовательский ввод.

Задача разбивается на 5 этапов:

1. Считать введенные пользователем данные
2. Сделать запрос к серверу
3. Распарсить ответ
4. Вывести результат поиска
5. Открыть нужную страницу в браузере

Использовать готовые библиотеки для работы с Википедией нельзя.

# 1.2. Спецификация программы

В данной программе реализованы 4 класса: Main, Get\_response, Parsing и Result:

* В Main происходит ввод пользователем запроса, инициализация url строки и вызов методов других классов;
* В Get\_response метод response\_api устанавливает соединение с url строкой и возвращает данные, полученные после подключения в формате String;
* В методе output класса Parsing с помощью библиотеки gson происходит выделение нужных полей исходной строки – title и pageid – и занесение последнего значения в массив для дальнейшего использования;
* В Results происходит открытие браузера с полученной добавлением нужного pageid к строке ссылкой на статью Википедии с помощью класса Desktop.

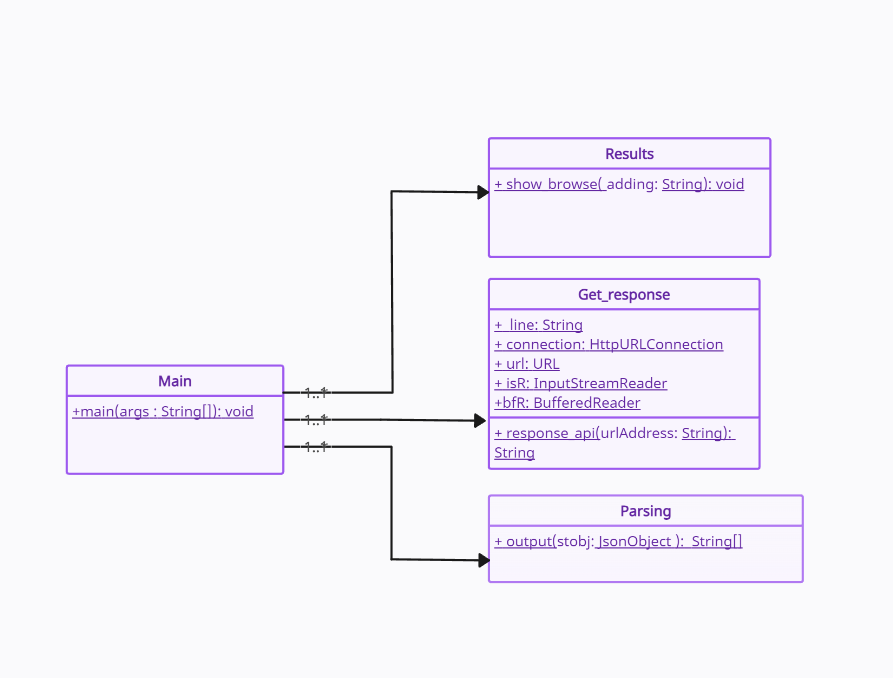
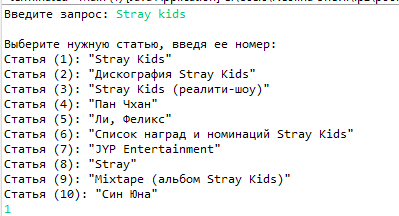
На Рисунке 1 отображена общая структура иерархии классов системы.

Рисунок 1. Диаграмма классов

# 1.3. Описание интерфейса пользователя программы

По заданию было реализовано консольное приложение, отображающее статью в браузере по выбору пользователя.

После запуска программы в консоль выводится текстовое сообщение, предлагающее пользователю ввести поисковой запрос для Википедии. Запрос может состоять как из одного слова, так и быть строкой. Далее выводятся первые 10 статей из поиска (или меньшее количество при отсутствии 10), из которых пользователь должен выбрать одну, введя ее номер (Рисунок 2). При корректном вводе происходит открытие браузера с выбранной статьей Википедии (Рисунок 3). В противном случае программа отображает текст, оповещающий о том, что запрос необходимо повторить, после чего заканчивает свою работу (Рисунок 4).

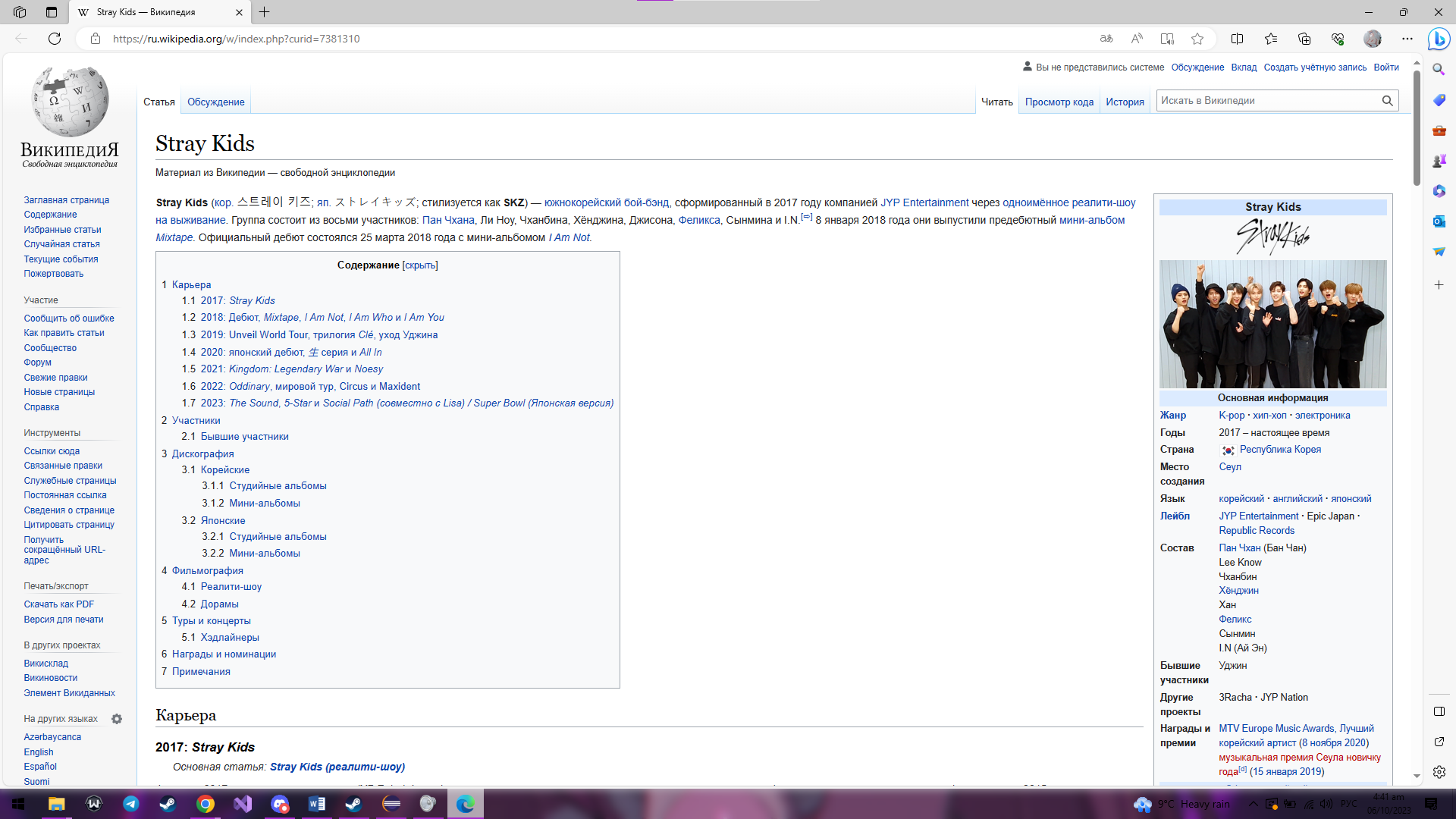
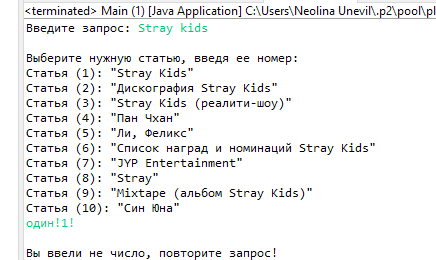
Рисунок 2

Рисунок 3

Рисунок 4

# 1.4. Текст программы

**Файл Main.java:**

**import** java.net.URLEncoder;

**import** java.nio.charset.StandardCharsets;

**import** java.util.InputMismatchException;

**import** java.util.Scanner;

**import** com.google.gson.JsonObject;

**import** com.google.gson.JsonParser;

**public** **class** Main {

**public** **static** void main(**String** args[])

{

//ввод запроса

**Scanner** in = **new** **Scanner**(**System**.in);

**System**.out.print("Bведите запрос: ");

**String** ask =in.nextLine();

**String** urlAddress = "https://ru.wikipedia.org/w/api.php?action=query&list=search&utf8=&format=json&srsearch="+ **URLEncoder**.encode(ask, **StandardCharsets**.UTF\_8);

**String** line="";

line= Get\_response.response\_api(urlAddress);//получение api

//распределение полученной строки по json объектам

JsonObject stobj= JsonParser.parseString(line).getAsJsonObject();

**String** pageid[]=Parsing.output(stobj);

int j=0;//индекс массива pageid

**try**

{

j =in.nextInt();

}

**catch**(**InputMismatchException** ex)

{

**System**.out.print("\nBы ввели не число, повторите запрос!\n");

**return**;

}

//этап отображения выбранной статьи в браузере

**if**(j>0&&j<pageid.length+1)

{

**String** adding = pageid[j-1];

in.close();

Results.show\_browse(adding);

}

**else**

**System**.out.print("\nBаше число выходит за границы заданного диапазона, повторите запрос!\n");

}

}

**Файл Parsing.java:**

**import** com.google.gson.JsonArray;

**import** com.google.gson.JsonObject;

**public** **class** Parsing

{

**static** **String**[] output(JsonObject stobj)

{

//создание объектов json по кодовым словам

JsonObject dataobj = stobj.getAsJsonObject("query");

JsonArray search = dataobj.getAsJsonArray("search");

**String**[] pageid =**new** **String**[search.size()];//строка для хранения pageid

**System**.out.print("\nBыберите нужную статью, введя ее номер: \n");

**for**(int i=0;i<search.size();i++)

{

JsonObject temp = search.get(i).getAsJsonObject();//получение i-того поля массива

**System**.out.printf("Статья (%d): %s\n",i+1, temp.getAsJsonPrimitive("title").toString());

pageid[i]= temp.getAsJsonPrimitive("pageid").toString();

}

**return** pageid;

}

}

**Файл Get\_response.java:**

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.net.HttpURLConnection;

**import** java.net.URL;

**public** **class** Get\_response

{

**static** **String** line;

**static** **HttpURLConnection** connection;

**static** **URL** url;

**static** **InputStreamReader** isR;

**static** **BufferedReader** bfR;

**static** **String** response\_api(**String** urlAddress)

{

**try**

{

url = **new** **URL**(urlAddress);

connection = (**HttpURLConnection**) url.openConnection();//возвращение в connection объекта URLConnection

**if** (**HttpURLConnection**.HTTP\_OK == connection.getResponseCode()) //удалось установить соединение

{

isR = **new** **InputStreamReader**(connection.getInputStream());//декодирование считанных байтов в символы во входном потоке

bfR = **new** **BufferedReader**(isR);//буферизация символов с потока для чтения далее

line = bfR.readLine();

}

**else**

{

**System**.out.printf("Fail %s", connection.getResponseCode());

}

}

**catch** (**IOException** e)

{

e.printStackTrace();

}

**return** line;

}

}

**Файл Results.java:**

**import** java.awt.Desktop;

**import** java.io.IOException;

**import** java.net.URI;

**import** java.net.URISyntaxException;

**import** java.net.URLEncoder;

**import** java.nio.charset.StandardCharsets;

**public** **class** Results

{

**static** void show\_browse(**String** adding)

{

**try**

{ //проверка на то, поддерживается ли класс и возможно ли открытие браузера

**if** (Desktop.isDesktopSupported() && Desktop.getDesktop().isSupported(Desktop.**Action**.BROWSE))

{

//возвращает экземпляр контекста браузера с указанным URI

Desktop.getDesktop().browse(**new** **URI**("https://ru.wikipedia.org/w/index.php?curid="+**URLEncoder**.encode(adding, **StandardCharsets**.UTF\_8)));

}

}

**catch** (**IOException** e)

{

**throw** **new** **RuntimeException**(e);

}

**catch** (**URISyntaxException** e)

{

**throw** **new** **RuntimeException**(e);

}

}

}

# 1.5. Выводы

В ходе данной практики были получены навыки работы с некоторыми функциями языка java для работы со ссылками. Каждый шаг выполнения представлял собой изучение возможностей новых для нас библиотек, классов и методов. Наибольший интерес в программе представляет парсинг строк с использованием библиотеки gson, так как возможно большое количество реализаций данного задания. Еще один новый для нас класс – Desktop. С помощью этого класса было реализовано открытие ссылки в браузере, среди других его функций есть также открытие приложения почты. Подводя итоги, можно сказать, что умения, полученные при написании программы, обязательно пригодятся в дальнейшем, так как они затрагивают возможности работы с интернет-пространством.