**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**ИДЗ**

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

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Новиков Г.В. |
| Студентка гр. 1302 |  | Романова О.В. |
| 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

Описание реализованных классов. UML-диаграмма классов.

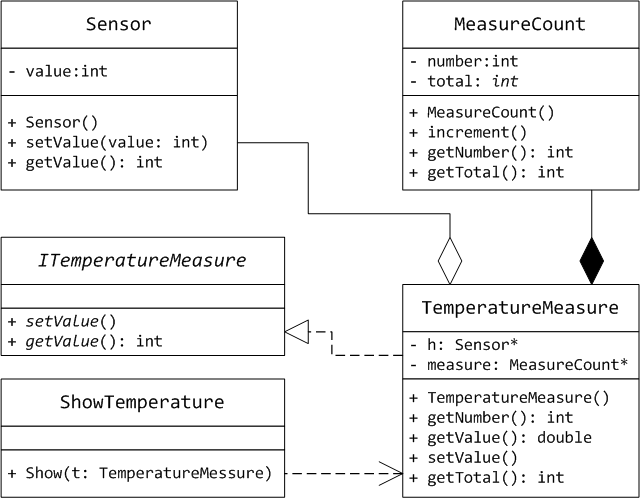


Рисунок 1. Диаграмма классов

# 1.3. Описание интерфейса пользователя программы

Пользователь вводит запрос для поиска по Википедии (рис.2).



Рис. 2

В консоль выводятся результаты поиска – заголовки и json (рис.3).

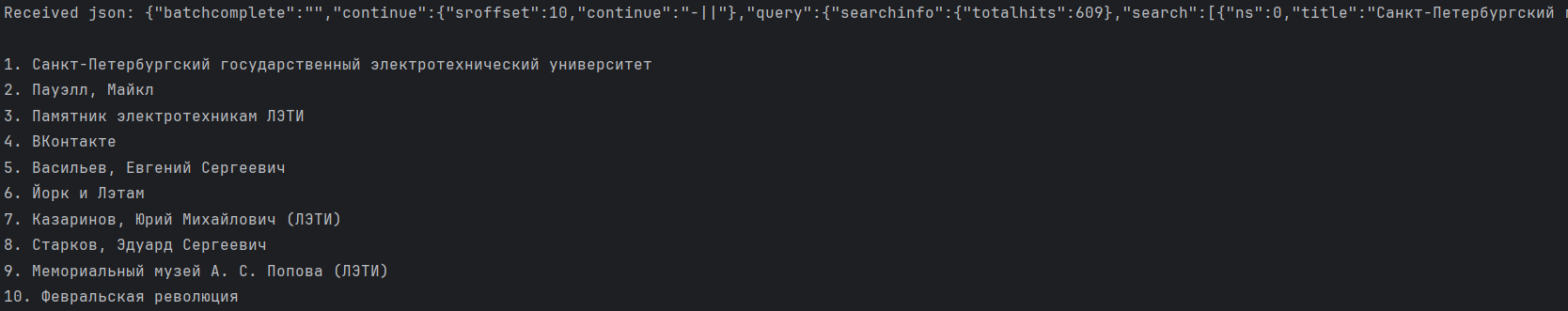


Рис. 3

Пользователь вводит номер заголовка, который необходимо открыть (рис.4).



Рис. 4

В консоль выводится URL и автоматически открывается ссылка (рис.5).



Рис. 5

# 1.4. Текст программы

Main.java:

import java.awt.Desktop;  
import java.io.IOException;  
import java.net.URISyntaxException;  
import java.util.Scanner;  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import com.google.gson.JsonObject;  
import com.google.gson.JsonArray;  
import java.net.URI;  
import java.util.regex.Pattern;  
  
public class Main  
{  
 private static String ReadWordFromConsole(Scanner in) {  
 String request;  
 while (true) {  
 System.*out*.print("Enter word: ");  
 request = in.nextLine();  
 if (request.isBlank()) {  
 System.*out*.println("Please enter word to search\n");  
 } else break;  
 }  
 return request;  
 }  
  
 public static int ReadArticleNumber(Scanner in, int max) {  
 String choice\_str;  
 int choice;  
  
 while (true) {  
 System.*out*.print("\nEnter article number: ");  
  
 choice\_str = in.nextLine().strip();  
 Pattern pattern = Pattern.*compile*("\\d+");  
 if (!choice\_str.isBlank() && pattern.matcher(choice\_str).matches()) {  
 choice = Integer.*parseInt*(choice\_str);  
 if (0 < choice && choice <= max) break;  
 }  
 System.*out*.println("Please enter number from 1 to " + max);  
 }  
 return choice;  
 }  
  
 public static void main(String[] args) throws IOException, URISyntaxException, InterruptedException {  
 Scanner in = new Scanner(System.*in*);  
  
 // Search and send request to the server  
 SearchEngine engine = new SearchEngine();  
 String result\_json;  
  
 String request;  
 while (true) {  
 try {  
 request = *ReadWordFromConsole*(in);  
 result\_json = engine.WikiSearch(request);  
 break;  
 } catch (IOException e) {  
 System.*out*.println("Error: cannot connect to Wikipedia. Please check your internet connection");  
 }  
 }  
  
 System.*out*.println("Received json: " + result\_json + "\n");  
  
 // Parsing  
 GsonBuilder builder = new GsonBuilder();  
 Gson gson = builder.create();  
  
 JsonObject json = gson.fromJson(result\_json, JsonObject.class);  
 JsonObject query = json.getAsJsonObject("query");  
 JsonArray result = query.getAsJsonArray("search");  
  
 if (result.isEmpty())  
 {  
 System.*out*.println("No results found");  
 System.*exit*(0);  
 }  
 // Titles output  
 String title;  
 for (int i = 0; i < result.size(); i++)  
 {  
 JsonObject titles = result.get(i).getAsJsonObject();  
 title = titles.get("title").getAsString();  
 System.*out*.println((i + 1) + ". " + title);  
 }  
  
 int choice = *ReadArticleNumber*(in, result.size());  
  
 // Getting pageId  
 JsonObject id = result.get(choice - 1).getAsJsonObject();  
 String pageId = id.get("pageid").getAsString();  
  
 // Open wiki  
 String wiki\_url = "https://ru.wikipedia.org/w/index.php?curid=" + pageId;  
 System.*out*.println("URL:" + wiki\_url);  
 Desktop.*getDesktop*().browse(new URI(wiki\_url));  
  
 in.close();  
 }  
}

SearchEngine.java:

import java.io.IOException;  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.net.URLEncoder;  
import java.net.http.HttpClient;  
import java.net.http.HttpRequest;  
import java.net.http.HttpResponse;  
import java.net.http.HttpResponse.BodyHandlers;  
import java.nio.charset.StandardCharsets;  
  
public class SearchEngine  
{  
 public String WikiSearch(String requestString) throws IOException, InterruptedException {  
 URI uri;  
 try  
 {  
 uri = new URI("https://ru.wikipedia.org/w/api.php?action=query&list=search&utf8=&format=json&srsearch=" + URLEncoder.*encode*(requestString, StandardCharsets.*UTF\_8*));  
 } catch (URISyntaxException e)  
 {  
 throw new RuntimeException(e);  
 }  
 HttpRequest request = HttpRequest.*newBuilder*(uri).build();  
  
 HttpClient client = HttpClient.*newBuilder*()  
 .version(HttpClient.Version.*HTTP\_2*)  
 .build();  
  
 HttpResponse<String> response = client.send(request, BodyHandlers.*ofString*());  
  
 return response.body();  
 }  
}

# 1.5. Выводы

В ходе выполнения лабораторной работы мы впервые поработали с Java, познакомились с библиотекой Gson и классом Desktop. Также в лабораторной работе использовались API, JSON и URL. Библиотека Gson используется для преобразования JSON, класс Desktop используется для автоматического открытия статьи Википедии в Google.

# Разработка приложения для обработки данных

# Это 2 практика.

# Разработка серверного приложения

# Это 3 практика.

# 4. Подготовка модели данных и подключение СУБД

# Это 4 практика.

# 5. Сборка и публикация результатов

Разработанный программный код собирался с помощью системы автоматизированной сборки Intellij IDEA. Результаты были выложены на Github:

<https://github.com/Gregory-hub/leti_2/tree/main/Kotlin(Java)/lab1>

<https://github.com/Tulenchi/kotlin>