**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**КАФЕДРА САПР**

**Практическая работа №1**

**по дисциплине**

**«Объектно-ориентированное программирование»**

**Тема: «Консольная программа для поиска в Википедии»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3354 |  | Филимонова.Е.С. |
| Преподаватель |  | Кулагин М.В. |

Санкт-Петербург

2024

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. **Спецификация программы**

**Основной класс** — WikipediaSearch:

* Принимает запрос от пользователя.
* Использует API Википедии для поиска статей по запросу.
* Позволяет пользователю выбрать статью из списка результатов.
* Показывает текст статьи и открывает её в браузере.

**API класс** — WikipediaApi:

* Выполняет запросы к API Википедии.
* Возвращает результаты поиска и текст статей.

**Результат поиска** — WikipediaSearchResult:

* Хранит данные о статье, включая её идентификатор страницы и название.

**Открытие статьи в браузере** — WikipediaBrowser:

* Открывает выбранную статью в браузере, используя её идентификатор.

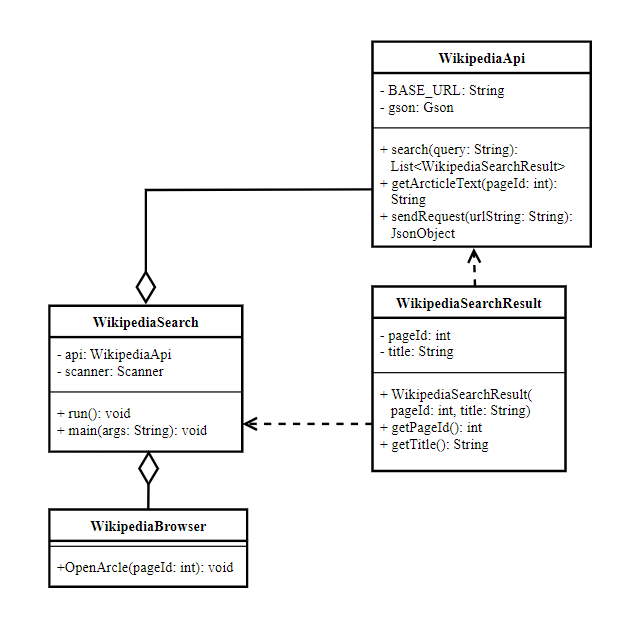


Рисунок 1. UML-диаграмма классов

**3. Описание интерфейса пользователя программы**

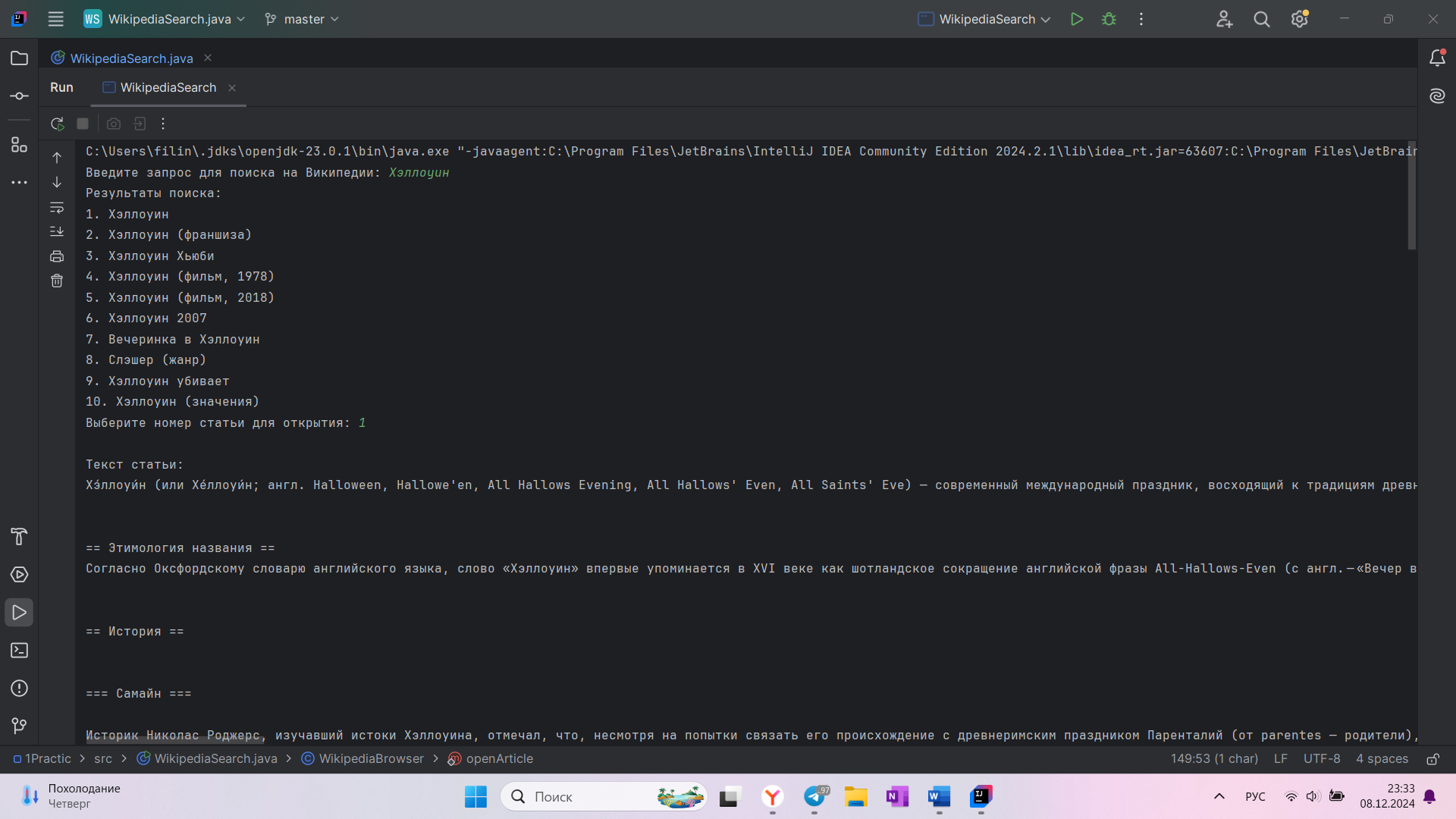


Рисунок 2. Результат работы программы в консоли

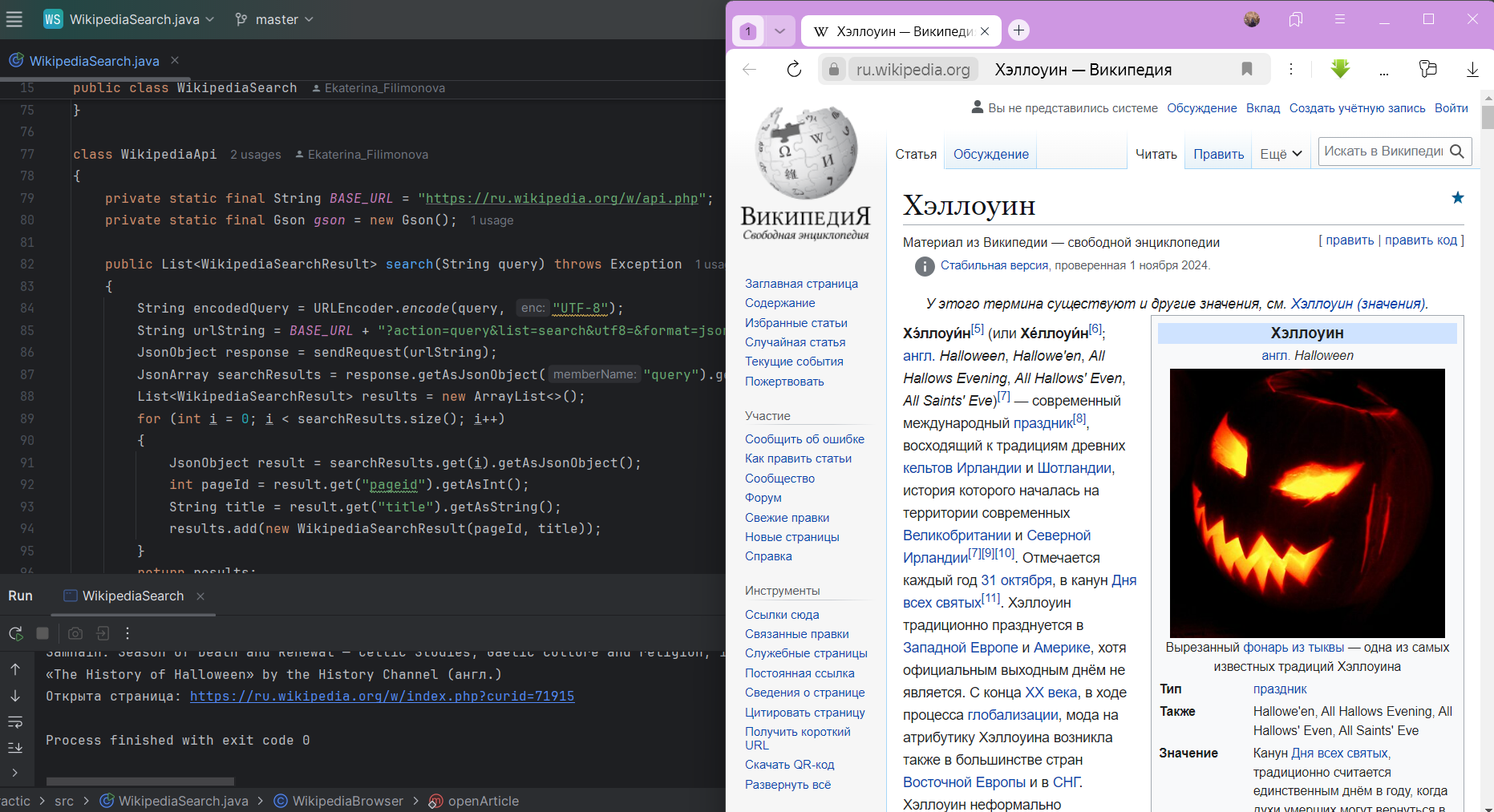


Рисунок 3. Результат работы программы в браузере

Программа начинает выполнение с вывода сообщения: «Введите запрос для поиска на Википедии:». После чего **пользователь вводит строку**, например, "Хэллоуин". После чего отправляется запрос к API википедии, используя введенную пользователем строку и кодируя её в URL. В ответ возвращается JSON-объект с идентификаторами страниц и названием статей. Данный ответ парсится, создается список из найденных статей, который выводится пользователю. Далее программа выводит сообщение «Выберите номер статьи для открытия:» и ожидает **номер, который вводит пользователь**, например, «1». После выбора пользователем номера статьи возвращается текст статьи в формате JSON, который извлекается и выводится в консоль. Далее статья открывается в Википедии в браузере.

В процессе пользователь может наткнуться на ошибки выполнения программы:

* Если введённый запрос оказался пустым, программа выдаст сообщение: «Запрос не должен быть пустым. Пожалуйста, попробуйте снова.»
* Если номер статьи будет выходить за пределы списка, то последует сообщение: «Неверный выбор. Попробуйте снова.».
* Если произойдёт ошибка при подключении к API или во время работы с википедией, будет выведено сообщение: «Произошла ошибка: [описание ошибки]».
* Если браузер не откроется, то последует «Не удалось открыть страницу: [описание ошибки]».

**4. Текст программы**

import java.awt.Desktop;  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.net.HttpURLConnection;  
import java.net.URI;  
import java.net.URL;  
import java.net.URLEncoder;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
import com.google.gson.Gson;  
import com.google.gson.JsonArray;  
import com.google.gson.JsonObject;  
  
public class WikipediaSearch  
{  
 WikipediaApi api = new WikipediaApi();  
 Scanner scanner = new Scanner(System.*in*);  
  
 public void run()  
 {  
 try  
 {  
 System.*out*.print("Введите запрос для поиска на Википедии: ");  
 String query = scanner.nextLine().trim();  
  
 if (query.isEmpty())  
 {  
 System.*out*.println("Запрос не должен быть пустым. Пожалуйста, попробуйте снова.");  
 return;  
 }  
  
 List<WikipediaSearchResult> results = api.search(query);  
 if (results.isEmpty())  
 {  
 System.*out*.println("Результаты не найдены.");  
 return;  
 }  
  
 System.*out*.println("Результаты поиска:");  
 for (int i = 0; i < results.size(); i++)  
 {  
 System.*out*.printf("%d. %s\n", i + 1, results.get(i).getTitle());  
 }  
  
 System.*out*.print("Выберите номер статьи для открытия: ");  
 int choice = scanner.nextInt();  
 if (choice < 1 || choice > results.size())  
 {  
 System.*out*.println("Неверный выбор. Попробуйте снова.");  
 return;  
 }  
  
 WikipediaSearchResult selectedArticle = results.get(choice - 1);  
 String articleText = api.getArticleText(selectedArticle.getPageId());  
 System.*out*.println("\nТекст статьи:");  
 System.*out*.println(articleText);  
 WikipediaBrowser.*openArticle*(selectedArticle.getPageId());  
  
 }  
 catch (Exception e)  
 {  
 System.*out*.println("Произошла ошибка: " + e.getMessage());  
 }  
 finally  
 {  
 scanner.close();  
 }  
 }  
  
 public static void main(String[] args)  
 {  
 new WikipediaSearch().run();  
 }  
}  
  
class WikipediaApi  
{  
 private static final String *BASE\_URL* = "https://ru.wikipedia.org/w/api.php";  
 private static final Gson *gson* = new Gson();  
  
 public List<WikipediaSearchResult> search(String query) throws Exception  
 {  
 String encodedQuery = URLEncoder.*encode*(query, "UTF-8");  
 String urlString = *BASE\_URL* + "?action=query&list=search&utf8=&format=json&srsearch=" + encodedQuery;  
 JsonObject response = sendRequest(urlString);  
 JsonArray searchResults = response.getAsJsonObject("query").getAsJsonArray("search");  
 List<WikipediaSearchResult> results = new ArrayList<>();  
 for (int i = 0; i < searchResults.size(); i++)  
 {  
 JsonObject result = searchResults.get(i).getAsJsonObject();  
 int pageId = result.get("pageid").getAsInt();  
 String title = result.get("title").getAsString();  
 results.add(new WikipediaSearchResult(pageId, title));  
 }  
 return results;  
 }  
  
 public String getArticleText(int pageId) throws Exception  
 {  
 String urlString = *BASE\_URL* + "?action=query&prop=extracts&explaintext&format=json&pageids=" + pageId;  
 JsonObject response = sendRequest(urlString);  
 JsonObject pages = response.getAsJsonObject("query").getAsJsonObject("pages");  
 JsonObject article = pages.getAsJsonObject(String.*valueOf*(pageId));  
 return article.get("extract").getAsString();  
 }  
  
 private JsonObject sendRequest(String urlString) throws Exception  
 {  
 HttpURLConnection connection = (HttpURLConnection) new URL(urlString).openConnection();  
 BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
 StringBuilder response = new StringBuilder();  
 String line;  
 while ((line = in.readLine()) != null)  
 {  
 response.append(line);  
 }  
 in.close();  
 return *gson*.fromJson(response.toString(), JsonObject.class);  
 }  
}  
  
class WikipediaSearchResult  
{  
 private int pageId;  
 private String title;  
  
 public WikipediaSearchResult(int pageId, String title)  
 {  
 this.pageId = pageId;  
 this.title = title;  
 }  
  
 public int getPageId()  
 {  
 return pageId;  
 }  
  
 public String getTitle()  
 {  
 return title;  
 }  
}  
  
class WikipediaBrowser  
{  
 public static void openArticle(int pageId)  
 {  
 String pageUrl = "https://ru.wikipedia.org/w/index.php?curid=" + pageId;  
 try  
 {  
 if (Desktop.*isDesktopSupported*())  
 {  
 Desktop.*getDesktop*().browse(new URI(pageUrl));  
 System.*out*.println("Открыта страница: " + pageUrl);  
 }  
 else  
 {  
 System.*out*.println("Откройте страницу сами :)");  
 }  
 }  
 catch (Exception e)  
 {  
 System.*out*.println("Не удалось открыть страницу: " + e.getMessage());  
 }  
 }  
}

1. **Выводы**

Для начала создается объект класса WikipediaSearch, который инициализирует два важных компонента: api объект класса WikipediaApi, который будет использоваться для поиска и получения текста статей; scanner объект класса Scanner, с помощью которого программа будет считывать ввод пользователя. В методе run() начинается выполнение основного алгоритма, а метод main() вызывает его.

Метод run(): программа ожидает пользователем ввода запроса и сохраняет полученную строку в query, проверяет является ли она пустой: если да, то завершает работу, если нет, то вызывает метод search(query) объекта api для поиска статей. После получения результатов поиска программа проверяет, есть ли результаты. Если результаты не найдены (список пуст), программа завершает работу. Если результаты найдены, программа выводит список найденных статей, предлагает выбор пользователю, считывает номер статьи и если он корректен извлекает объект WikipediaSearchResult по выбранному индексу (номер статьи) из списка results. Далее вызывает метод getArticleText(pageId) объекта api, передавая идентификатор страницы статьи. Выводит полученный текст статьи на экран и вызывает метод openArticle(pageId) из класса WikipediaBrowser для открытия статьи в браузере. В завершении работы программа закрывает объект scanner.

Метод search(query): Кодирует строку запроса в формат URL, формирует URL для запроса к API Википедии, отправляет HTTP-запрос к API Википедии, используя метод sendRequest().

Метод sendRequest(): получает JSON-ответ от API и парсит его с помощью библиотеки Gson в объект JsonObject. Из объекта JsonObject извлекаются данные: pageId и title для каждой найденной статьи. Эти данные используются для создания объектов типа WikipediaSearchResult, которые добавляются в список.

Метод getArticleText(pageId): Формирует URL для получения текста статьи, отправляет запрос и получает ответ от API, который содержит текст статьи в формате JSON, извлекает текст статьи из ответа и возвращает его.

Метод openArticle(pageId): Формирует URL страницы статьи в Википедии, проверяет, поддерживает ли система возможность открытия браузера с помощью Java. Если да, то открывает URL в браузере.

Результаты были выложены на Github:

[**https://github.com/Filimonova-Ekaterina/OOP\_Practice\_1/blob/master/src/WikipediaSearch.java**](https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2FFilimonova-Ekaterina%2FOOP_Practice_1%2Fblob%2Fmaster%2Fsrc%2FWikipediaSearch.java&utf=1)