МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
КАФЕДРА САПР

Практическая работа №1

по дисциплине

«Объектно-ориентированное программирование»

Студент гр. 4354 _____ Чучалин И.В.

Преподаватель _____ Кулагин М.В.

Санкт-Петербург
2025

Цель работы

Разработать программу для поиска и просмотра статей в Википедии через консоль. Данная программа должна быть устойчива к некорректному вводу и быть понятной для пользования.

Задача

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.

Спецификация программы

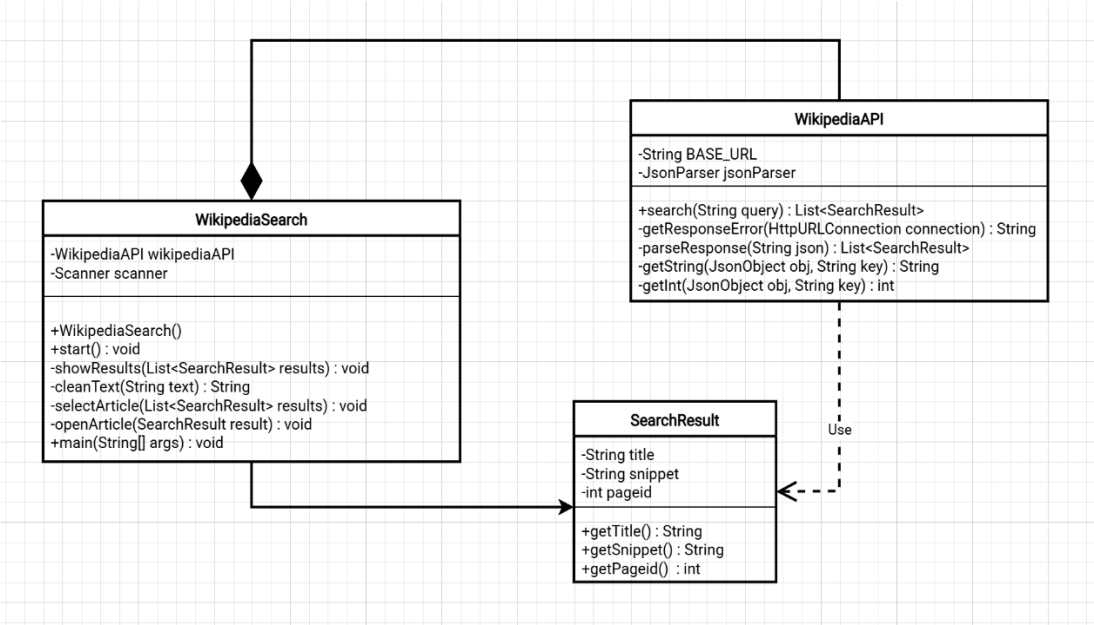

Рисунок 1 - Диаграмма классов

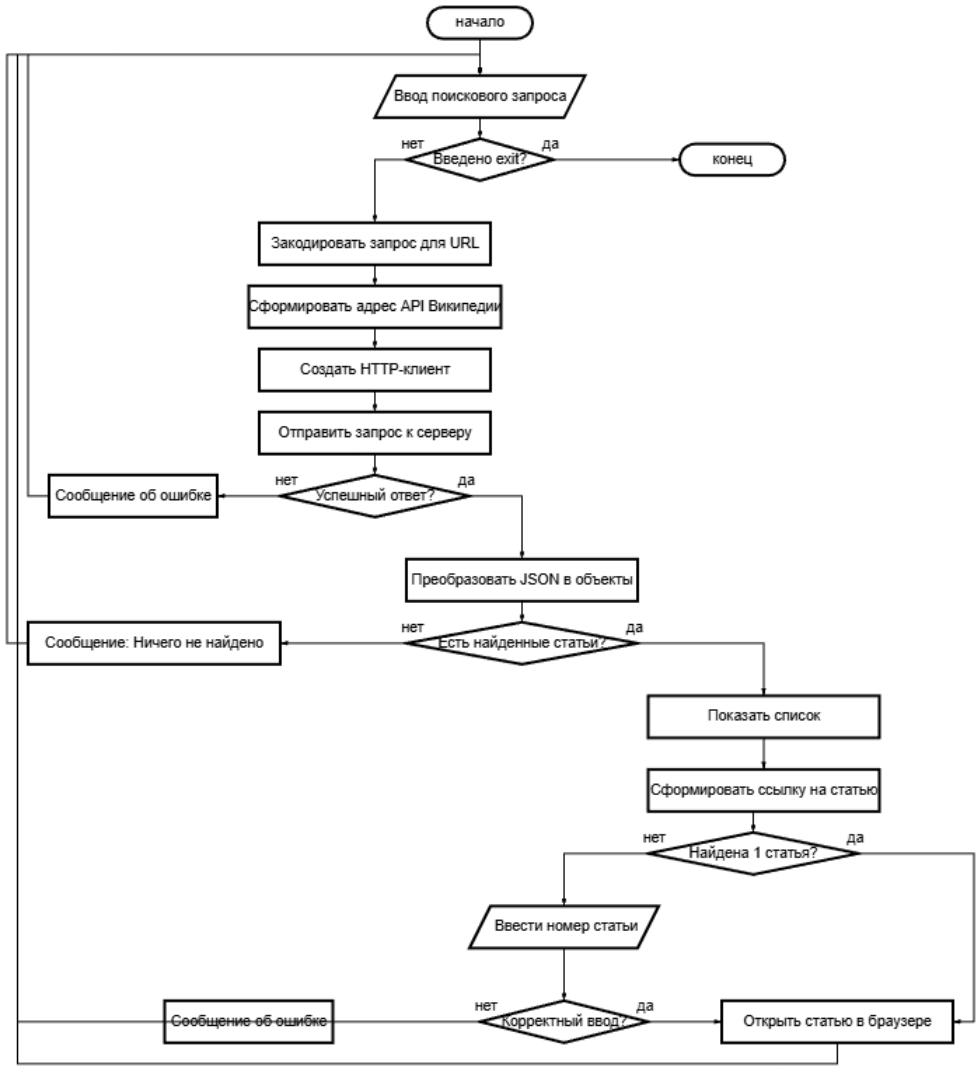

Рисунок 2 - Блок-схема

Пример работы программы

```
Программа поиска в Википедии
Для выхода введите 'exit'

Введите запрос: Телефон

Найдено 10 результатов:
1. Телефон
   Телефо́н (от др.-греч. τῆλε «далеко» + φωνή «голос», «звук») — аппарат, имеющий трубку и сигнальное
   ...

2. Сотовый телефон
   правилами написания статей. (25 августа 2025) Со́товый телефо́н, или моби́льный телефо́н, — телефон,
   ...

3. Чёрный телефон 2
   «Чёрный телефо́н 2» (англ. Black Phone 2) — американский фильм ужасов о сверхъестественном режиссёра
   ...

4. Чёрный телефон
   «Чёрный телефо́н» (англ. The Black Phone) — фильм ужасов о сверхъестественном режиссёра Скотта Дерри
   ...

5. IP-телефония
   IP-телефони́я (произносится «айпи́-телефони́я») — телефонная связь по протоколу IP. Под IP-телефоние
   ...

6. Телефон (значения)
   Телефо́н: Телефон — аппарат для голосовой связи на больших расстояниях. Телефон — электроакустически
   ...

7. Яндекс.Телефон
   Яндекс.Телефон — смартфон российской компании «Яндекс» (ODM-производитель — Arima, Тайвань). Презент
   ...

8. Телефон доверия
   Телефо́н дове́рия (линия жизни, кризисная линия, горячая линия, линия помощи) — дистанционная служба
   ...

9. Испорченный телефон
   текущую статью с помощью перевода «Сло́манный телефо́н» (испо́рченный телефо́н, глухо́й телефо́н, ки
   ...

10. Телефон (мультфильм)
   Корнея Чуковского. Корней Чуковский читает созданную им же сказку «Телефон», а по телефону к нему об
   ...

Выберите номер статьи (1-10): 3
Открывается: Чёрный телефон 2
Введите запрос:
```

Рисунок 3 – Вывод в консоль

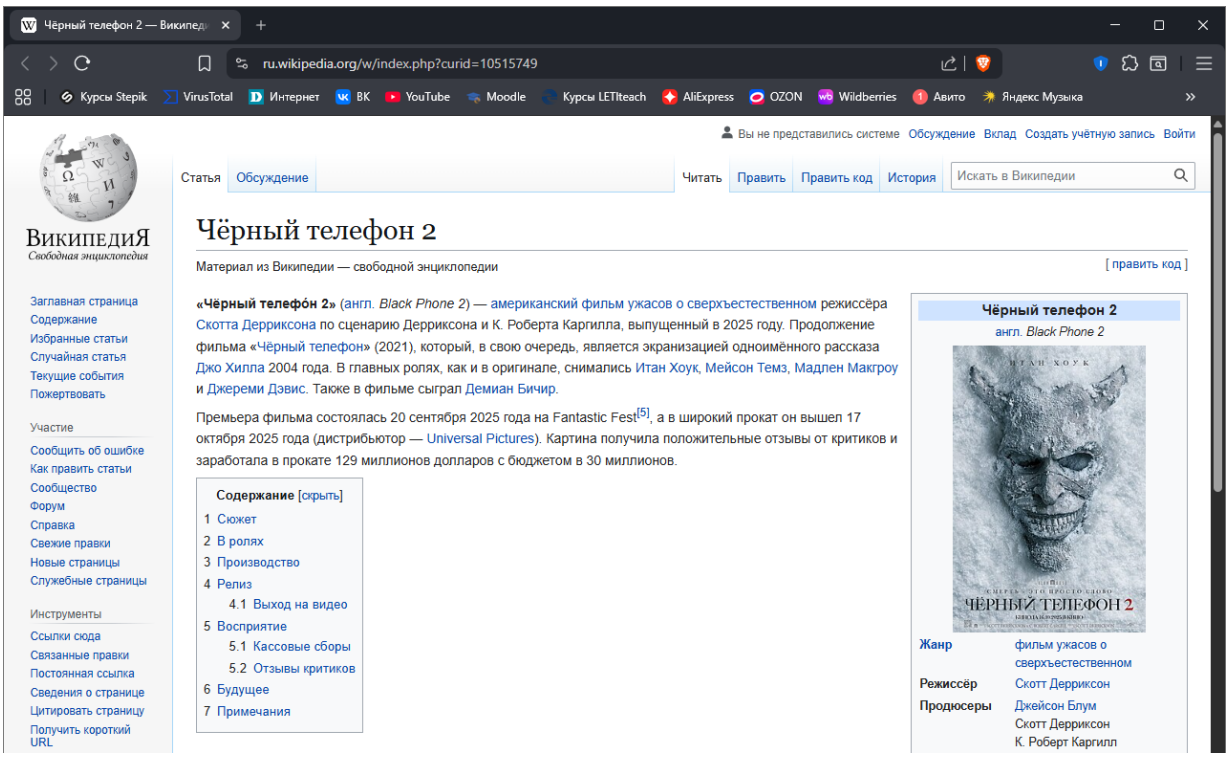

Рисунок 4 – Открытая статья

Текст программы

1) pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>OOP_1</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>25</maven.compiler.source>
        <maven.compiler.target>25</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <repositories>
        <repository>
            <id>central</id>
            <url>https://repo.maven.apache.org/maven2</url>
        </repository>
    </repositories>
    <dependencies>
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.13.2</version>
        </dependency>
    </dependencies>

</project>
```

2) WikipediaSearch.java

```java
package wikipedia;

import wikipedia.models.SearchResult;
import java.net.URI;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Scanner;

public class WikipediaSearch {
    private WikipediaAPI wikipediaAPI;
    private Scanner scanner;

    public WikipediaSearch() {
        wikipediaAPI = new WikipediaAPI();
        scanner = new Scanner(System.in);
    }

    public void start() {
        System.out.println("Программа поиска в Википедии");
        System.out.println("Для выхода введите 'exit'");
```

```
        System.out.println();

        while (true) {
            System.out.print("Введите запрос: ");

            try {
                String query = scanner.nextLine();

                if (query.equalsIgnoreCase("exit")) {
                    break;
                }

                if (query.trim().isEmpty()) {
                    System.out.println("Введите непустой запрос");
                    continue;
                }

                List<SearchResult> results;
                try {
                    results = wikipediaAPI.search(query);
                } catch (IllegalArgumentException e) {
                    System.out.println("Ошибка: " + e.getMessage());
                    continue;
                } catch (java.net.UnknownHostException e) {
                    System.out.println("Ошибка сети: не удается найти
сервер");
                    continue;
                } catch (java.net.SocketTimeoutException e) {
                    System.out.println("Таймаут подключения. Проверьте
интернет");
                    continue;
                } catch (java.net.ConnectException e) {
                    System.out.println("Не удается подключиться к серверу");
                    continue;
                } catch (java.io.IOException e) {
                    System.out.println("Ошибка ввода-вывода: " +
e.getMessage());
                    continue;
                } catch (java.lang.Exception e) {
                    System.out.println("Ошибка: " + e.getMessage());
                    continue;
                }

                if (results.isEmpty()) {
                    System.out.println("Ничего не найдено");
                    continue;
                }

                showResults(results);

                if (results.size() > 1) {
                    selectArticle(results);
                } else {
                    openArticle(results.get(0));
                }

            } catch (java.util.NoSuchElementException e) {
                System.out.println("Ошибка ввода");
                break;
            } catch (java.lang.IllegalStateException e) {
                System.out.println("Ошибка сканера");
                break;
            }
```

```
        }

        scanner.close();
        System.out.println("Программа завершена");
    }

    private void showResults(List<SearchResult> results) {
        System.out.println();
        System.out.println("Найдено " + results.size() + " результатов:");

        for (int i = 0; i < results.size(); i++) {
            SearchResult result = results.get(i);
            String title = result.getTitle();
            String snippet = cleanText(result.getSnippet());

            System.out.println((i + 1) + ". " + title);
            if (!snippet.isEmpty()) {
                System.out.println("   " + snippet.substring(0,
Math.min(snippet.length(), 100)));
                if (snippet.length() > 100) {
                    System.out.println("   ...");
                }
            }
            System.out.println();
        }
    }

    private String cleanText(String text) {
        return text.replaceAll("<[^>]+>", "")
                .replaceAll("&[a-z]+;", "")
                .trim();
    }

    private void selectArticle(List<SearchResult> results) {
        System.out.print("Выберите номер статьи (1-" + results.size() + "):
");

        try {
            String input = scanner.nextLine();
            int choice = Integer.parseInt(input);

            if (choice >= 1 && choice <= results.size()) {
                openArticle(results.get(choice - 1));
            } else {
                System.out.println("Неверный номер");
            }
        } catch (NumberFormatException e) {
            System.out.println("Введите число");
        } catch (InputMismatchException e) {
            System.out.println("Неверный формат ввода");
        }
    }

    private void openArticle(SearchResult result) {
        try {
            String url = "https://ru.wikipedia.org/w/index.php?curid=" +
result.getPageid();
            System.out.println("Открывается: " + result.getTitle());

            if (java.awt.Desktop.isDesktopSupported()) {
                java.awt.Desktop.getDesktop().browse(new URI(url));
            } else {
                System.out.println("Ссылка: " + url);
```

```
            }
        } catch (java.net.URISyntaxException e) {
            System.out.println("Ошибка формата URL");
        } catch (java.io.IOException e) {
            System.out.println("Ошибка при открытии браузера");
        } catch (java.lang.UnsupportedOperationException e) {
            System.out.println("Операция не поддерживается");
        }
    }

    public static void main(String[] args) {
        WikipediaSearch search = new WikipediaSearch();
        search.start();
    }
}
```

3) WikipediaAPI.java

```
package wikipedia;

import com.google.gson.JsonArray;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import com.google.gson.JsonSyntaxException;
import wikipedia.models.SearchResult;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.net.URLEncoder;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;

public class WikipediaAPI {
    private static final String BASE_URL =
"https://ru.wikipedia.org/w/api.php";
    private static final JsonParser jsonParser = new JsonParser();

    public List<SearchResult> search(String query) throws IOException {
        if (query == null || query.trim().isEmpty()) {
            throw new IllegalArgumentException("Запрос пустой");
        }

        String encodedQuery;
        try {
            encodedQuery = URLEncoder.encode(query.trim(), "UTF-8");
        } catch (java.io.UnsupportedEncodingException e) {
            throw new IOException("Ошибка кодирования запроса", e);
        }

        String urlString = BASE_URL +
"?action=query&list=search&format=json&srsearch=" + encodedQuery;

        HttpURLConnection connection = null;
        BufferedReader reader = null;
```

```
        try {
            URL url = new URL(urlString);
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setConnectTimeout(10000);
            connection.setReadTimeout(10000);
            connection.setRequestProperty("User-Agent", "WikiSearchBot/1.0");
            connection.setRequestProperty("Accept", "application/json");

            int responseCode = connection.getResponseCode();

            if (responseCode != HttpURLConnection.HTTP_OK) {
                String errorMessage = getResponseError(connection);
                throw new IOException("HTTP ошибка: " + responseCode + " - "
+ errorMessage);
            }

            reader = new BufferedReader(new
InputStreamReader(connection.getInputStream(), "UTF-8"));

            StringBuilder response = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                response.append(line);
            }

            return parseResponse(response.toString());

        } catch (MalformedURLException e) {
            throw new IOException("Некорректный URL", e);
        } catch (UnknownHostException e) {
            throw new IOException("Не удается найти сервер Википедии", e);
        } catch (SocketTimeoutException e) {
            throw new IOException("Таймаут подключения", e);
        } catch (java.net.ConnectException e) {
            throw new IOException("Не удается подключиться к серверу", e);
        } finally {
            if (reader != null) {
                try {
                    reader.close();
                } catch (IOException e) {
                }
            }
            if (connection != null) {
                connection.disconnect();
            }
        }
    }

    private String getResponseError(HttpURLConnection connection) {
        try {
            BufferedReader errorReader = new BufferedReader(
                    new InputStreamReader(connection.getErrorStream(), "UTF-
8")
            );
            StringBuilder errorResponse = new StringBuilder();
            String line;
            while ((line = errorReader.readLine()) != null) {
                errorResponse.append(line);
            }
            errorReader.close();
            return errorResponse.toString();
```

```
        } catch (Exception e) {
            return "Не удалось получить сообщение об ошибке";
        }
    }

    private List<SearchResult> parseResponse(String json) throws IOException
{
        try {
            JsonObject jsonObject = jsonParser.parse(json).getAsJsonObject();

            if (!jsonObject.has("query")) {
                return new ArrayList<>();
            }

            JsonObject queryObj = jsonObject.getAsJsonObject("query");
            if (!queryObj.has("search")) {
                return new ArrayList<>();
            }

            JsonArray searchArray = queryObj.getAsJsonArray("search");
            List<SearchResult> results = new ArrayList<>();

            for (JsonElement element : searchArray) {
                JsonObject obj = element.getAsJsonObject();

                SearchResult result = new SearchResult() {
                    private final String title = getString(obj, "title");
                    private final String snippet = getString(obj, "snippet");
                    private final int pageid = getInt(obj, "pageid");

                    @Override
                    public String getTitle() {
                        return title;
                    }

                    @Override
                    public String getSnippet() {
                        return snippet;
                    }

                    @Override
                    public int getPageid() {
                        return pageid;
                    }
                };

                results.add(result);
            }
            return results;

        } catch (JsonSyntaxException e) {
            throw new IOException("Ошибка формата JSON", e);
        } catch (IllegalStateException e) {
            throw new IOException("Некорректный ответ сервера", e);
        } catch (NullPointerException e) {
            throw new IOException("Пустой ответ от сервера", e);
        }
    }

    private String getString(JsonObject obj, String key) {
        if (obj.has(key) && !obj.get(key).isJsonNull()) {
            return obj.get(key).getAsString();
        }
```

```
        return "";
    }

    private int getInt(JsonObject obj, String key) {
        if (obj.has(key) && !obj.get(key).isJsonNull()) {
            return obj.get(key).getAsInt();
        }
        return 0;
    }
}
```

4) SearchResult.java

```
package wikipedia.models;

public class SearchResult {
    private String title;
    private String snippet;
    private int pageid;

    public String getTitle() {
        if (title == null) {
            return "";
        }
        return title;
    }

    public String getSnippet() {
        if (snippet == null) {
            return "";
        }
        return snippet;
    }

    public int getPageid() {
        return pageid;
    }
}
```

Выводы

В результате выполнения практической работы №1 было разработано консольное приложение для поиска статей в Википедии. Программа получает запрос пользователя, выполняет поиск, выводит результаты и открывает выбранную статью в браузере, при этом корректно реагирует на любой пользовательский ввод.

В ходе работы освоены навыки выполнения HTTP-запросов к внешним API, обработки JSON-ответов с использованием библиотеки Gson, реализации кодирования URL и открытия веб-страниц через Java Desktop API.

Разработанный программный код собирался с помощью системы автоматизированной сборки Maven. Результаты были выложены на Github: https://github.com/tethranialn/OOP_1