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

Практическая работа №1

по дисциплине

«Объектно-ориентированное программирование»

Студенты гр. 4351 _____ Пашаян А.Н.
Погодин Д.А.

Преподаватель _____ Кулагин М.В.

Санкт-Петербург
2025

1. Задание

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

2. Спецификация программы

Для работы программы реализован класс WikipediaApp, в котором реализован функционал программы, а также класс WikipediaClient, в котором реализованы функции для работы с API Википедии.

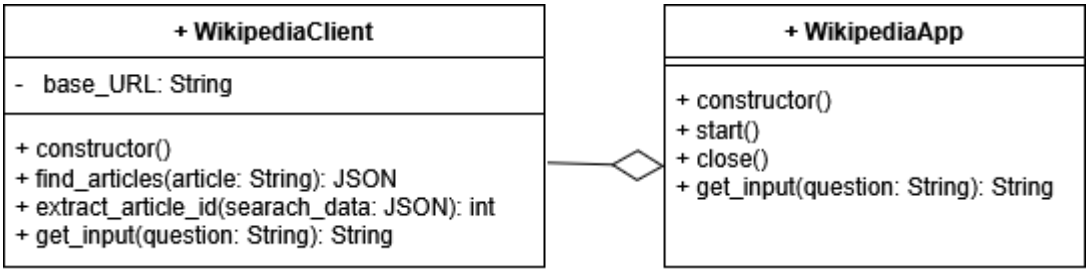

Рисунок 1. Диаграмма классов

3. Описание интерфейса пользователя программы

При запуске программы пользователю будет предложено ввести статью, которую он хочет найти, затем выбрать из списка предложенных (рисунок 2). После выбора статьи она будет открыта в браузере.

```
C:\Users\Admin>cd C:\Users\Admin\Desktop\jjj

C:\Users\Admin\Desktop\jjj>node wikipedia.js
Поиск статьи: диарея
0. Диарея
1. Стронгилоидоз
2. Дефекация
3. Обезвоживание организма
4. Химиотерапия злокачественных новообразований
5. Глюкагонома
6. Макролиды
7. Синдром Вернера — Моррисона
8. Детская смесь
9. Фасциолопсидоз

Выберите статью:
```

Рисунок 2

В случае, если статья не найдена, программа выведет об этом сообщение (рисунок 3).

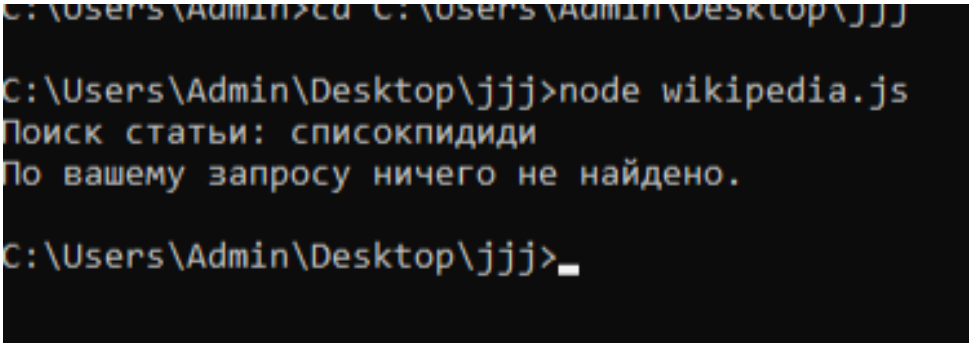

Рисунок 3

4. Текст программы

```
const https = require("https")
const readline = require("readline")
const open = require('open').default

class WikipediaClient {
    constructor(base_URL) {
        this.base_URL = base_URL
    }

    find_articles(article) {
        return new Promise((resolve, reject) => {
            const options = {
                hostname: this.base_URL,
                path:
encodeURI(`/w/api.php?action=query&list=search&utf8=&format=json&srsearch="${arti
cle}"`),
                headers: {
                    'User-Agent': 'MyCustomApp/1.0 (Node.js HTTPS Client)'
                }
            }
            const req = https.get(options, (response) => {
                let data = ''

                response.on('data', (chunk) => {
                    data += chunk
                })
                response.on('end', () => {
                    try {
                        const data_JSON = JSON.parse(data)
                        resolve(data_JSON)
                    }
                    catch (error) {
```

```
                        reject(new Error('Ошибка парсинга JSON: ' +
error.message))
                    }
                })
            })

            req.on('error', (error) => {
                reject(new Error('Ошибка запроса: ' + error.message))
            })

            req.end()
        })
    }

    get_articles_num(search_data) {
        // Проверка на существование searchinfo
        if (!search_data.query || !search_data.query.searchinfo) {
            return 0
        }
        const num = search_data.query.searchinfo.totalhits
        return num > 10 ? 10 : num
    }

    get_article(search_data, num) {
        // Проверка на существование search массива
        if (!search_data.query || !search_data.query.search) {
            return null
        }
        return search_data.query.search[num];
    }

    get_articles_preview(search_data) {
        const articles_num = this.get_articles_num(search_data)

        // Если статей нет, возвращаем сообщение
        if (articles_num === 0) {
            return "По вашему запросу ничего не найдено."
        }

        let preview = ''
        for (let i = 0; i < articles_num; i++) {
            const article = this.get_article(search_data, i)
            if (article) {
                preview += `${i}. ${article.title}\n`
            }
        }
        return preview
    }

    // Метод для проверки наличия результатов
```

```
    has_results(search_data) {
        return this.get_articles_num(search_data) > 0
    }
}

class WikipediaApp {
    constructor() {
        this.client = new WikipediaClient("ru.wikipedia.org")
        this.rl = readline.createInterface({
            input: process.stdin,
            output: process.stdout
        })
    }

    get_input(question) {
        return new Promise((resolve) => {
            this.rl.question(question, resolve)
        })
    }

    close() {
        this.rl.close()
    }

    async start() {
        try {
            const article = await this.get_input('Поиск статьи: ')

            // Проверяем, не пустой ли запрос
            if (!article || article.trim() === '') {
                console.log('Вы ввели пустой запрос. Пожалуйста, введите
существующий термин.')
                this.close()
                return
            }

            const search_data = await this.client.find_articles(article)

            console.log(this.client.get_articles_preview(search_data))

            // Если нет результатов, завершаем программу
            if (!this.client.has_results(search_data)) {
                this.close()
                return
            }

            let article_num
            let is_input_invalid = true
            while(is_input_invalid) {
                article_num = Number(await this.get_input('Выберите статью: '))
```

```
                is_input_invalid = !(Number.isInteger(article_num) && article_num
<= 9 && article_num >= 0)

                if (is_input_invalid) {
                    console.log('Пожалуйста, введите число от 0 до 9')
                }
            }

            const selectedArticle = this.client.get_article(search_data,
article_num)
            if (!selectedArticle) {
                console.log('Ошибка: выбранная статья не найдена')
                this.close()
                return
            }

            const article_pageid = selectedArticle.pageid
            const articleUrl =
`https://ru.wikipedia.org/w/index.php?curid=${article_pageid}`

            console.log('Открываю статью в браузере...')

            await open(articleUrl, { wait: true })

            console.log('Статья успешно открыта!')

            await new Promise(resolve => setTimeout(resolve, 1000))

        }
        catch (error) {
            console.log('Ошибка: ' + error.message)
        }
        finally {
            this.close()
        }
    }
}

const app = new WikipediaApp()
app.start()
```

5. Выводы

В ходе данной практической работы была написана программа, которая с консоли считывает поисковый запрос пользователя и выводит результат поиска по Википедии. Программа принимает от пользователя поисковый запрос и выводит пронумерованный список до 10 найденных статей из Wikipedia. Пользователь выбирает номер статьи для просмотра в браузере. Приложение отображает сообщения о процессе открытия для понятной обратной связи.

В ходе разработки были изучены принципы работы с сетевыми запросами: выполнение HTTPS-запросов к API Wikipedia, обработка JSON-ответов, кодирование параметров URL и автоматизированное открытие веб-страниц с использованием пакета 'open'.

Разработанный программный код необходимо запускать в среде NodeJS. Результаты были выложены на GitHub:

https://github.com/JoyKkk/OOPLabor1