ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени федеральное государственное

Бюджетное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра информатики

Лабораторная работа №7

По дисциплине: «Технология разработки программного обеспечения»

Тема: «Вэб-сканер»

Выполнил студент:

Группы: БФИ1902

Шацкий Е. И.

Проверила:

Мосева. М.С.

Москва, 2020 г.

**Содержание**

[1 Задачи на лабораторную работу. 3](#_Toc59486842)

[2 Ход решения лабораторной работы. 4](#_Toc59486843)

[2.1 Листинги программы 4](#_Toc59486844)

[Листинг 1 – класс Crawler 4](#_Toc59486845)

[Листинг 2 – класс URLDeathPair 6](#_Toc59486846)

[Листинг 3 – класс Main 6](#_Toc59486847)

[2.2 Результаты работы программы. 7](#_Toc59486848)

[Список используемых источников 8](#_Toc59486849)

# **1 Задачи на лабораторную работу.**

Для выполнения лабораторной работы необходимо:

а) Программа должна принимать в командной строке два параметра:

1) Строку, которая представляет собой URL-адрес, с которого можно начать просмотр страницы.

2) Положительное целое число, которое является максимальной глубиной поиска. Если указаны некорректные аргументы, программа должна немедленно остановиться и выдать сообщение об используемых аргументах, например: usage: java Crawler

б) Программа должна хранить URL-адрес в виде строки вместе с его глубиной (которая для начала будет равна 0). Вам будет необходимо создать класс для представления пар [URL, depth].

в) Программа должна подключиться к указанному сайту в URL-адресе на порт 80 с использованием сокета (см. ниже) и запросить указанную вебстраницу.

г) Программа должна проанализировать возвращаемый текст, построчно для любых подстрок, имеющих формат: [найденные URL-адреса должны быть сохранены в паре с новым значением глубины в LinkedList (URL, depth) (подробнее о LinkedLists см. ниже). Новое значение глубины должно быть больше, чем значение глубины URL-адреса, соответствующего анализируемой странице.](%5bлюбой_URL-адрес_начинающийся_с_http:/%5d)

д) Далее программа должна закрыть соединение сокета с хостом.

е) Программа должна повторять шаги с 3 по 6 для каждого нового URL-адреса, если глубина, соответствующая URL-адресу, меньше максимальной. Обратите внимание, что при извлечении и поиске определенного URL-адреса глубина поиска увеличивается на 1. Если глубина URL-адреса достигает максимальной глубины (или больше), не извлекайте и не ищите эту веб-страницу.

ж) Наконец, программа должна вывести все посещенные URLстраницы вместе с их глубиной поиска.

# **2 Ход решения лабораторной работы.**

## **2.1 Листинги программы**

Ниже представлены листинги классов программы, которые обеспечивают решение поставленных задач.

## Листинг 1 – класс Crawler

import java.io.\*;  
import java.net.Socket;  
import java.net.URL;  
import java.util.Arrays;  
import java.util.LinkedList;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Crawler { // Поиск ссылок (собственно сканер)  
 public LinkedList<UrlDepthPair> proccessed; // List по ссылкам (хранит ссылки на предыдущий и следующий элемент)  
  
 public Crawler() {  
 proccessed = new LinkedList<>();  
 }  
  
 public void startParsing(URL baseUrl, int maxDepth, int currentDepth) {  
 if (currentDepth > maxDepth) return;  
  
 LinkedList<UrlDepthPair> links = *getAllLinks*(baseUrl, currentDepth); // Запись всех эллементов (ссылок в новый лист)  
  
 for (UrlDepthPair link: links) {  
 startParsing(link.realUrl, maxDepth, currentDepth + 1); // Для нахождение "подссылок"  
 }  
  
 proccessed.addAll(links); // Запись всех ссылок в лист proccessed из links  
 }  
  
 private static LinkedList<UrlDepthPair> getAllLinks(URL url, int depth) {  
 try {  
 LinkedList<UrlDepthPair> links = new LinkedList<>();  
  
 int port = 80;  
 String hostname = url.getHost(); // Имя хоста  
  
 Socket socket = new Socket(hostname, port);  
 socket.setSoTimeout(3000); // Вызывается исключение после того как проходит 3000 мс  
  
 OutputStream outStream = socket.getOutputStream(); // Ссылка на поток в который мы будем записывать данные  
  
 PrintWriter writer = new PrintWriter(outStream, true); //Класс PrintStream - это именно тот класс, который используется для вывода на консоль. Когда мы выводим на консоль некоторую информацию с помощью вызова System.out.println(), то тем самым мы задействует PrintStream, так как переменная out в классе System как раз и представляет объект класса PrintStream, а метод println() - это метод класса PrintStream.  
  
 if (url.getPath().length() == 0) { //вывод в консоль  
 writer.println("GET / HTTP/1.1"); //если нет пары  
 writer.println("Host: " + hostname);  
 writer.println("Accept: text/html");  
 writer.println("Accept-Language: en,en-US;q=0.9,ru;q=0.8");  
 writer.println("Connection: close");  
 writer.println();  
 }  
 else {  
 writer.println("GET " + url.getPath() + " HTTP/1.1"); //есть пара  
 writer.println("Host: " + hostname);  
 writer.println("Accept: text/html");  
 writer.println("Accept-Language: en,en-US;q=0.9,ru;q=0.8");  
 writer.println("Connection: close");  
 writer.println();  
 }  
  
 InputStream input = socket.getInputStream(); //открывает входной поток данных  
 BufferedReader reader = new BufferedReader(new InputStreamReader(input)); //считывает текст из потока ввода данных  
  
 String htmlLine;  
  
 Pattern patternURL = Pattern.*compile*( //compline создаем неизменяемый обьект потом мы его используем  
 "(href=**\"**http|href=**\"**https)://([\\w\_-]+(?:(?:\\.[\\w\_-]+)+))([\\w.,@?^=%&:/~+#-]\*[\\w@?^=%&/~+#-])?" //шаблон - по которому мы ориентируемся  
 );  
  
 while ((htmlLine = reader.readLine()) != null) { //считываем текст из потока  
 Matcher matcherURL = patternURL.matcher(htmlLine); //определяет совпадение со строкой  
 while (matcherURL.find()) { //ищет совпадения  
 String link = htmlLine.substring(matcherURL.start() + 6, //считываем посимвольно  
 matcherURL.end());  
  
 links.add(new UrlDepthPair(link, depth));  
 }  
 }  
 socket.close();  
  
 return links;  
 } catch (Exception e) { //если сработало исключение  
 System.*out*.print(e.getMessage());  
 System.*out*.print(Arrays.*toString*(e.getStackTrace()));  
  
 return new LinkedList<>();  
 }  
 }  
}

## Листинг 2 – класс URLDeathPair

import java.net.MalformedURLException;  
import java.net.URL;  
  
public class UrlDepthPair { //Программа должна хранить URL-адрес в виде строки вместе с его глубиной (которая для начала будет равна 0). Вам будет необходимо создать класс для представления пар [URL, depth].  
  
 public String url;  
 public int depth;  
 public URL realUrl;  
  
 public UrlDepthPair (String url, int depth) throws MalformedURLException {  
 realUrl = new URL(url);  
 this.url = url;  
 this.depth = depth;  
 }  
}

## Листинг 3 – класс Main

import java.net.MalformedURLException;  
import java.net.URL;  
import java.util.LinkedList;  
  
public class Main {  
 public static <var> void main(String[] args) throws MalformedURLException {  
 URL url = new URL("http://mtuci.ru/"); //URL-адрес, с которого можно начать просмотр страницы.  
  
 Crawler parser = new Crawler();  
 parser.startParsing(url, 2, 0); //Положительное целое число, которое является максимальной глубиной поиска  
  
 LinkedList<UrlDepthPair> result = parser.proccessed;  
  
 result.forEach(urlDepthPair -> { //выводит адрес и глубину  
 System.*out*.println(  
 "URL is " + urlDepthPair.url +  
 " | depth: " + urlDepthPair.depth);  
 });  
 }  
}

## **2.2 Результаты работы программы.**

На рисунке 1 представлены результат работы программы.

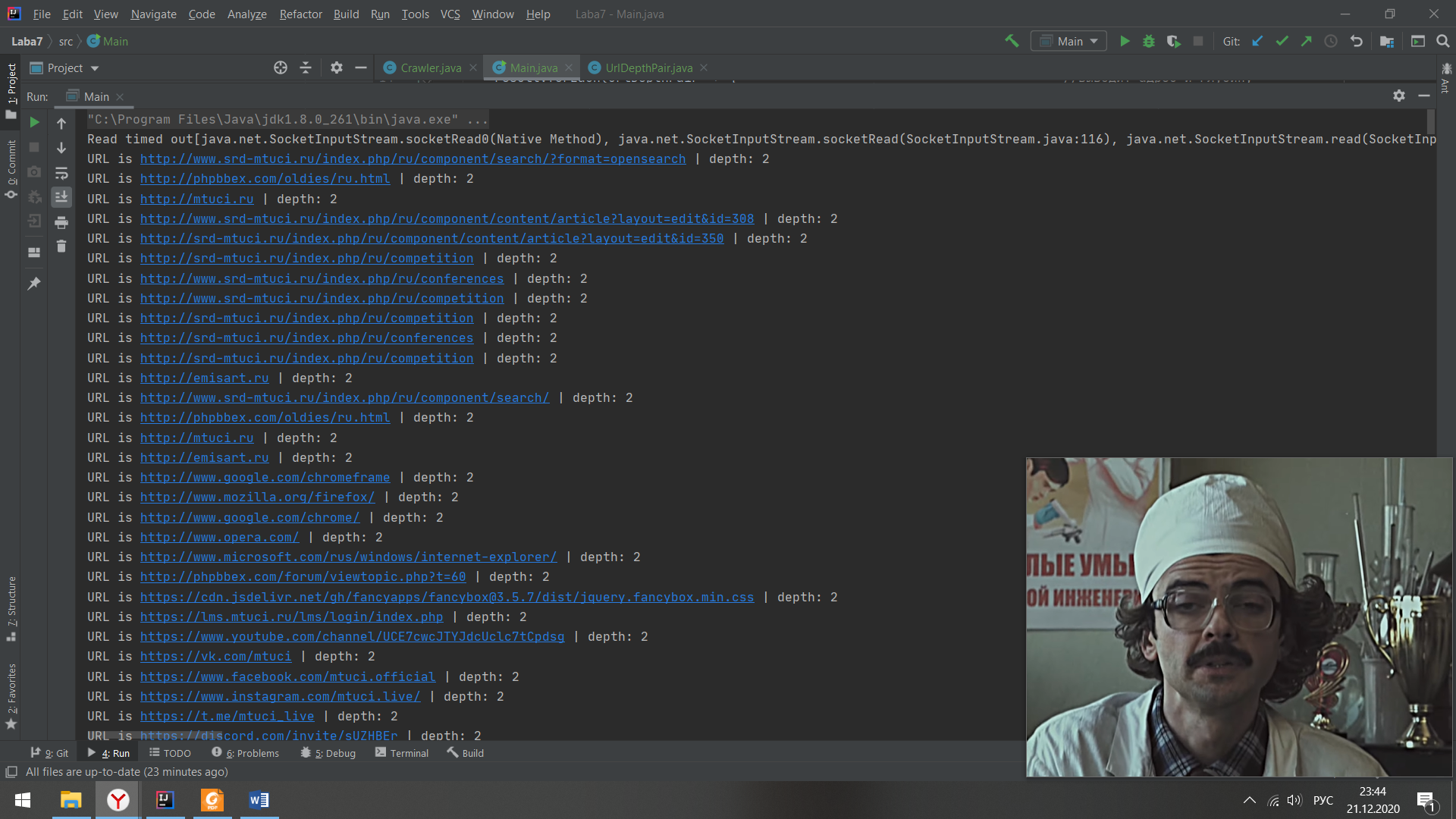


Рисунок 1 – Результат работы программы

# **Список используемых источников**

1.ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчёт о научно-исследовательской работе. Структура и правила оформления.

2.ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления.