Лабораторная работа №7

по дисциплине

“Объектно-ориентированное программирование”

Выполнил студент

группы БФИ1901

Гасанов Гамзат

Москва 2020

**Цель:**

Реализовать необходимо будет реализовать элементарный веб-сканер. Сканер будет автоматически загружать веб-страницы из Интернета, искать новые ссылки на этих страницах и повторять их. Он будет просто искать новые URL-адреса (местоположения веб-страниц) на каждой странице, собирать их и выводит в конце работы программы.

**Код программы:**

public class UrlDepthPair {  
 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;  
 }  
}

public class Crawler {  
 public LinkedList<UrlDepthPair> proccessed;  
  
 public Crawler() {  
 proccessed = new LinkedList<>();  
 }  
  
 public void startParsing(URL baseUrl, int maxDepth, int currentDepth) {  
 if (currentDepth > maxDepth) return;  
  
 LinkedList<UrlDepthPair> links = *getAllLinks*(baseUrl, currentDepth);  
  
 for (UrlDepthPair link: links) {  
 startParsing(link.realUrl, maxDepth, currentDepth + 1);  
 }  
  
 proccessed.addAll(links);  
 }  
  
 private static LinkedList<UrlDepthPair> getAllLinks(URL url, int depth) {  
 try {  
 LinkedList<UrlDepthPair> links = new LinkedList<>();  
  
 int port = 80;  
 String hostname = url.getHost();  
  
 Socket socket = new Socket(hostname, port);  
 socket.setSoTimeout(3000);  
  
 OutputStream outStream = socket.getOutputStream();  
  
 PrintWriter writer = new PrintWriter(outStream, true);  
  
 if (url.getPath().length() == 0) {  
 writer.println("GET / HTTP/1.1");  
 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");  
 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();  
 BufferedReader reader = new BufferedReader(new InputStreamReader(input));  
  
 String htmlLine;  
  
 Pattern patternURL = Pattern.*compile*(  
 "(href=\"http|href=\"https)://([\\w\_-]+(?:(?:\\.[\\w\_-]+)+))([\\w.,@?^=%&:/~+#-]\*[\\w@?^=%&/~+#-])?"  
 );  
  
 while ((htmlLine = reader.readLine()) != null) {  
 Matcher matcherURL = patternURL.matcher(htmlLine);  
 while (matcherURL.find()) {  
 String link = htmlLine.substring(matcherURL.start() + 6,  
 matcherURL.end());  
  
 links.add(new UrlDepthPair(link, depth));  
// if (links.size() >= 10) return links;  
 }  
 }  
 socket.close();  
  
 return links;  
 } catch (Exception e) {  
 System.*out*.print(e.getMessage());  
 System.*out*.print(Arrays.*toString*(e.getStackTrace()));  
  
 return new LinkedList<>();  
 }  
 }  
}

public class Main {  
 public static void main(String[] args) throws MalformedURLException {  
 URL url = new URL("http://mtuci.ru/");  
  
 var parser = new Crawler();  
 parser.startParsing(url, 2, 0);  
  
 LinkedList<UrlDepthPair> result = parser.proccessed;  
  
 result.forEach(urlDepthPair -> {  
 System.*out*.println(  
 "URL is " + urlDepthPair.url +  
 " | depth: " + urlDepthPair.depth);  
 });  
 }  
}

**Примеры работы программы:**

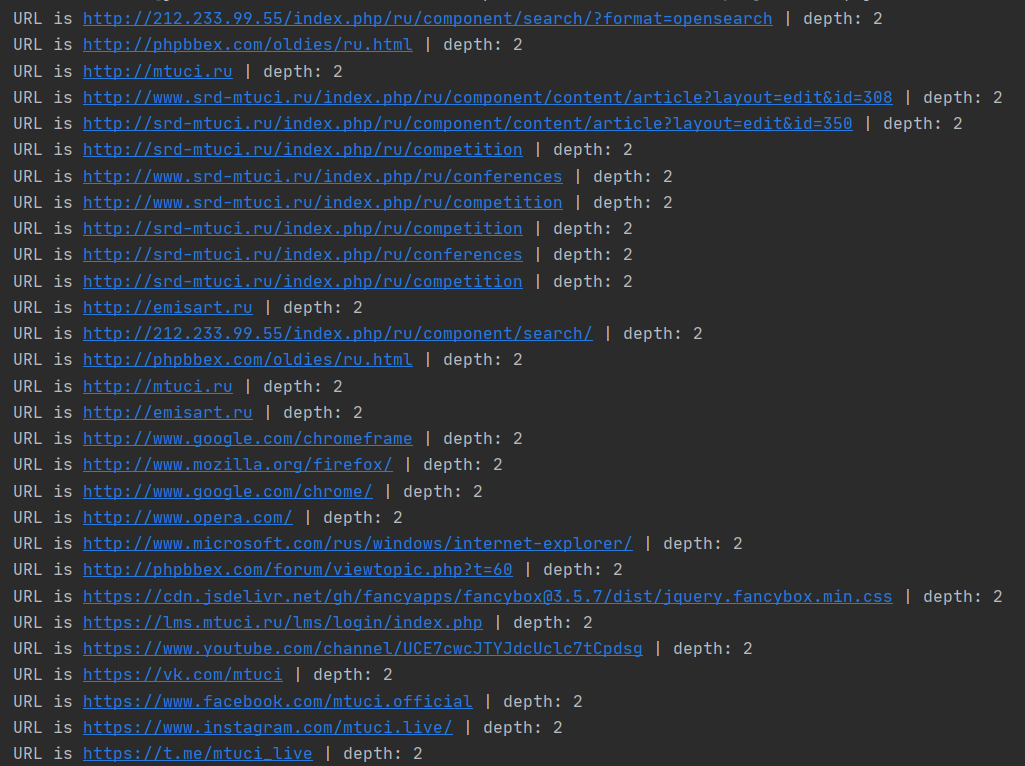


Рисунок 1 – Результат работы алгоритма