Лабораторная работа № 7

по дисциплине

“Объектно-ориентированное программирование”

Выполнил студент

группы БИБ1901

Бушеленков Роман

Москва 2021

**Цель работы:** реализовать элементарный веб-сканер.

**Задание на лабораторную работу:**

1. 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 (см. ниже) Если указаны некорректные аргументы, программа должна немедленно остановиться и выдать сообщение об используемых аргументах, например: usage: java Crawler

2. 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].

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

4. 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.

5. Далее программа должна закрыть соединение сокета с хостом.

6. 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.

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

**Результат выполнения работы:**

package lab7;  
  
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 {  
 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<>();  
 }  
 }  
}

package lab7;  
  
import java*.*net*.*MalformedURLException;  
import java*.*net*.*URL;  
  
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;  
 }  
}

package lab7;  
  
import java*.*net*.*MalformedURLException;  
import java*.*net*.*URL;  
import java*.*util*.*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);  
 });  
 }  
}

