**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Информатики

****

**Отчет по лабораторной работе № 7**

по дисциплине «КТП»

на тему:

«**Веб-сканер»**

Выполнила: студентка группы БВТ1802

Лаврухина Елена Павловна

Руководитель:

Полянцева Ксения Андреевна

Москва 2020

Цель работы

Реализовать элементарный веб-сканер.

Выполнение

Код программы

1. Crawler

import java.io.\*;  
import java.net.\*;  
import java.util.\*;  
public class Crawler {  
 public static void main(String[] args) {  
 String[] arg = new String[] {"http://www.mtuci.ru/","1"};  
 *// переменная для текущей глубины* int depth = 0;  
 *// проверяем, правильное ли количество параметров было введено* if (arg.length != 2) {  
 System.*out*.println("Введите URL и глубину");  
 System.*exit*(1);  
 }  
 else {  
 try {  
 depth = Integer.*parseInt*(arg[1]);  
 }  
 catch (NumberFormatException nfe) {  
 *// проверяем является ли глубина цифрой* System.*out*.println("Введите URL и глубину");  
 System.*exit*(1);  
 }  
 }  
 *// создаем связный список для представления ожидающих URL-адресов* LinkedList<URLDepthPair> pendingURLs = new LinkedList<URLDepthPair>();  
 *// создаем связный список для представления обработанных URL* LinkedList<URLDepthPair> processedURLs = new LinkedList<URLDepthPair>();  
 *// создаем глубину URL-пары для представления веб-сайта* URLDepthPair currentDepthPair = new URLDepthPair(arg[0], 0);  
 *// добавляем введенный веб-сайт в ожидающие URL-адреса* pendingURLs.add(currentDepthPair);  
 *// добавляем текущий веб-сайт* ArrayList<String> seenURLs = new ArrayList<String>();  
 seenURLs.add(currentDepthPair.getURL());  
 *// получаем все ссылки от каждого посещенного сайта пока URL-адреса не станут пустыми* while (pendingURLs.size() != 0) {  
 *// добавляем URL в список обработанных и сохраняем глубину* URLDepthPair dp = pendingURLs.pop();  
 processedURLs.add(dp);  
 int Depth = dp.getDepth();  
 *// получаем все ссылки с сайта и сохраняем их в новом связном списке* LinkedList<String> linksList = new LinkedList<String>();  
 linksList = Crawler.*getAllLinks*(dp);  
 *// добавляем ссылки, которые раньше не были видны, если мы не достигли максимальной глубины* if (Depth < depth) {  
 *// перебирать ссылки с сайта* for (int i = 0; i < linksList.size(); i++) {  
 String newURL = linksList.get(i);  
 *// продолжаем, если мы уже видели ссылку* if (seenURLs.contains(newURL)) {  
 continue;  
 }  
 *// создаем новую пару URL-глубина, если ссылка не встречалась ранее, и добавляем в списки* else {  
 URLDepthPair newDepthPair = new URLDepthPair(newURL, Depth + 1);  
 pendingURLs.add(newDepthPair);  
 seenURLs.add(newURL);  
 }  
 }  
 }  
 }  
 *// выводим все обработанные URL с глубиной* Iterator<URLDepthPair> iter = processedURLs.iterator();  
 while (iter.hasNext()) {  
 System.*out*.println(iter.next());  
 }  
 }  
 private static LinkedList<String> getAllLinks(URLDepthPair DepthPair) {  
 *// создаем связанный список LinkedList<String>, в котором будут храниться ссылки, что мы находим* LinkedList<String> URLs = new LinkedList<String>();  
 *// создаём новый сокет* Socket sock;  
 *// инициализируем новый сокет из строки String, содержащей имя хоста, и из номера порта, равного 80 (http)* try {  
 sock = new Socket(DepthPair.getWebHost(), 80);  
 }  
 *// ловим исключения неизвестного хоста и возвращаем пустой список* catch (UnknownHostException e) {  
 *//System.err.println("UnknownHostException: " + e.getMessage());* return URLs;  
 }  
 *// ловим исключения ввода-вывода и возвращаем пустой список* catch (IOException ex) {  
 *//System.err.println("IOException: " + ex.getMessage());* return URLs;  
 }  
 try {  
 *// устанавливаем таймаут сокета на 3 секунды* sock.setSoTimeout(3000);  
 }  
 *// ловим исключения сокета и возвращаем пустой список* catch (SocketException exc) {  
 System.*err*.println("SocketException: " + exc.getMessage());  
 return URLs;  
 }  
 *// поля для хранения пути к файлу и хоста* String docPath = DepthPair.getDocPath();  
 String webHost = DepthPair.getWebHost();  
 *// создаем поток вывода* OutputStream outStream;  
 *// возвращаем OutputStream связанный с сокетом используемый для передачи данных* try {  
 outStream = sock.getOutputStream();  
 }  
 *// ловим исключения ввода-вывода и возвращаем пустой список* catch (IOException exce) {  
 *//System.err.println("IOException: " + exce.getMessage());* return URLs;  
 }  
 *// создаем PrintWriter, true означает, что PrintWriter будет сбрасываться после каждого вывода* PrintWriter pw = new PrintWriter(outStream, true);  
 *// отправляем запрос на сервер* pw.println("GET " + docPath + " HTTP/1.1");  
 pw.println("Host: " + webHost);  
 pw.println("Connection: close");  
 pw.println();  
 *// создаем поток ввода* InputStream inStream;  
 *// возвращает InputStream связанный с объектом Socket используемый для приема данных* try {  
 inStream = sock.getInputStream();  
 }  
 *// ловим исключения ввода-вывода и возвращаем пустой список* catch (IOException excep) {  
 *//System.err.println("IOException: " + excep.getMessage());* return URLs;  
 }  
 *// создаем новый InputStreamReader и BufferedReader для чтения строк с сервера* InputStreamReader in = new InputStreamReader(inStream);  
 BufferedReader br = new BufferedReader(in);  
 while (true) {  
 String line;  
 try {  
 line = br.readLine();  
 }  
 *// ловим исключения ввода-вывода и возвращаем пустой список* catch (IOException except) {  
 System.*err*.println("IOException: " + except.getMessage());  
 return URLs;  
 }  
 *// прекращаем чтения документа* if (line == null)  
 break;  
 *// переменные для индексов* int beginIndex;  
 int endIndex;  
 int index = 0;  
 while (true) {  
 *// константа для строки, указывающей на ссылку* String URL\_PREFIX = "<a href=\"http";  
 int p = "<a href=\"".length();  
 *// константа для строки, указывающей конец веб-хоста* String END\_URL = "\"";  
 *// поиск нашего начала в текущей строке* index = line.indexOf(URL\_PREFIX, index);  
 if (index == -1)  
 break;  
 *// двигаем текущий индекс и устанавливаем в beginIndex* index += p;  
 beginIndex = index;  
 *// ищем наш конец в текущей строке и устанавливаем в endIndex* endIndex = line.indexOf(END\_URL, index);  
 index = endIndex;  
 *// устанавливаем ссылку на подстроку между начальным и конечным индексом, добавляем к нашему списку URL* try {  
 String newLink = line.substring(beginIndex, endIndex);  
 URLs.add(newLink);  
 }  
 *// ловим исключения несуществующего индекса в массиве* catch (IndexOutOfBoundsException ignored)  
 {  
 line = "";  
 }  
 }  
 }  
 *// возвращаем список URL* return URLs;  
 }  
}

1. URLDepthPair

import java.net.\*;  
public class URLDepthPair {  
 *// поля для хранения текущего URL и текущей глубины* private String currentURL;  
 private int currentDepth;  
 *// конструктор, который устанавливает ввод для текущих URL и глубины* public URLDepthPair(String URL, int depth) {  
 currentURL = URL;  
 currentDepth = depth;  
 }  
 *// метод, который возвращает текущий URL* public String getURL() {  
 return currentURL;  
 }  
 *// метод, который возвращает текущую глубину* public int getDepth() {  
 return currentDepth;  
 }  
 *// метод, который возвращает текущий URL и текущую глубину в строковом формате* public String toString() {  
 String stringDepth = Integer.*toString*(currentDepth);  
 return stringDepth + '\t' + currentURL;  
 }  
 *// метод, который возвращает путь документа текущего URL* public String getDocPath() {  
 try {  
 URL url = new URL(currentURL);  
 return url.getPath();  
 }  
 catch (MalformedURLException e) {  
 *//System.err.println("MalformedURLException: " + e.getMessage());* return null;  
 }  
 }  
 *// метод, который возвращает хост текущего URL* public String getWebHost() {  
 try {  
 URL url = new URL(currentURL);  
 return url.getHost();  
 }  
 catch (MalformedURLException e) {  
 *//System.err.println("MalformedURLException: " + e.getMessage());* return null;  
 }  
 }  
}

Скриншоты работы программы



Рисунок 1 – Работа программы с сайтом <http://www.mtuci.ru>



Рисунок 2 – Работа программы с сайтом <http://www.abitura.com/>

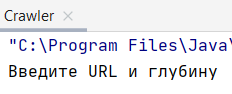


Рисунок 3 – Работа программы при неправильно введенных данных

Вывод

В ходе работы мы создали JAVA-приложение, которое реализует работу элементарного веб-сканера.