МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Знамени федеральное государственное

Бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И

ИНФОРМАТИКИ

Кафедра математической кибернетики и информационных технологий

Лабораторная работа № 7 по теме:

«Веб-сканер»

Выполнил: студент группы БИБ1902 \_\_\_\_\_\_\_Арустамян А.Б.

Проверил: \_\_\_\_\_\_\_\_Херсонский А. В.

Москва 2021

**Цель работы:**

В этой лабораторной работе 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:**

**Main.java**

package com.company;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.LinkedList;  
  
public class Main {  
  
 public static void main(String[] args) throws IOException {  
  
 String[] arg = new String[] {"http://slashdot.org/","1"};  
 int depth = 0;  
 try {  
 depth = Integer.*parseInt*(arg[1]);  
 }  
 catch (NumberFormatException nfe) {  
 System.*out*.println("usage: java Crawler <URL> <depth>");  
 System.*exit*(1);  
 }  
 LinkedList<URLDepthPair> pendingURLs = new LinkedList<URLDepthPair>();  
 LinkedList<URLDepthPair> processedURLs = new LinkedList<URLDepthPair>();  
 URLDepthPair currentDepthPair = new URLDepthPair(arg[0], 0);  
 pendingURLs.add(currentDepthPair);  
 ArrayList<String> seenURLs = new ArrayList<String>();  
 seenURLs.add(currentDepthPair.getURL());  
  
 while (pendingURLs.size() != 0) {  
 URLDepthPair depthPair = pendingURLs.pop();  
 processedURLs.add(depthPair);  
 int myDepth = depthPair.getDepth();  
 LinkedList<String> linksList;  
 linksList = Crawler.*getAllLinks*(depthPair);  
 if (myDepth < depth) {  
 for (int i=0;i<linksList.size();i++) {  
 String newURL = linksList.get(i);  
 if (seenURLs.contains(newURL)) {  
 continue;  
 }  
 else {  
 URLDepthPair newDepthPair = new URLDepthPair(newURL, myDepth + 1);  
 pendingURLs.add(newDepthPair);  
 seenURLs.add(newURL);  
 }  
 }  
 }  
 }  
 Iterator<URLDepthPair> iter = processedURLs.iterator();  
 while (iter.hasNext()) {  
 System.*out*.println(iter.next());  
 }  
 }  
}

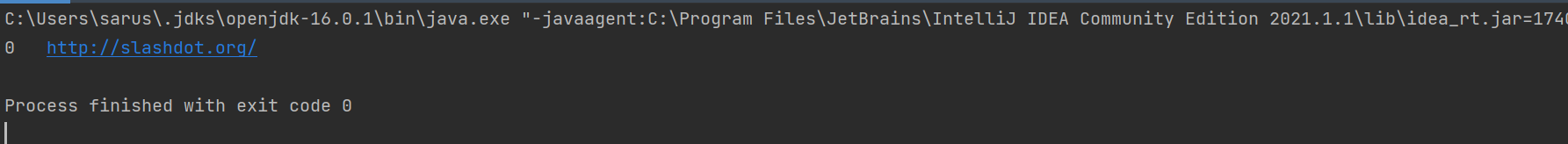
***Crawler.java***

package com.company;  
  
import java.io.\*;  
import java.net.\*;  
import java.util.\*;  
import java.util.regex.\*;  
  
public class Crawler {  
  
 static LinkedList<String> getAllLinks(URLDepthPair myDepthPair) throws IOException {  
 LinkedList<String> URLs = new LinkedList<String>();  
 Socket sock;  
 try {  
 sock = new Socket(myDepthPair.getWebHost(), 80);  
 }  
 catch (UnknownHostException e) {  
 System.*err*.println("UnknownHostException: " + e.getMessage());  
 return URLs;  
 }  
 catch (IOException ex) {  
 return URLs;  
 }  
 try {  
 sock.setSoTimeout(3000);  
 }  
 catch (SocketException exc) {  
 System.*err*.println("SocketException: " + exc.getMessage());  
 return URLs;  
 }  
 String docPath = myDepthPair.getDocPath();  
 String webHost = myDepthPair.getWebHost();  
 OutputStream outStream;  
 try {  
 outStream = sock.getOutputStream();  
 }  
 catch (IOException exce) {  
 return URLs;  
 }  
 PrintWriter myWriter = new PrintWriter(outStream, true);  
 if (docPath.length() == 0) {  
 myWriter.println("GET / HTTP/1.1");  
 myWriter.println("Host: " + webHost);  
 myWriter.println("Connection: close");  
 myWriter.println();  
 }  
 else {  
 myWriter.println("GET " + docPath + " HTTP/1.1");  
 myWriter.println("Host: " + webHost);  
 myWriter.println("Connection: close");  
 myWriter.println();  
 }  
 InputStream inStream;  
 try {  
 inStream = sock.getInputStream();  
 }  
 catch (IOException excep) {  
 return URLs;  
 }  
 InputStreamReader inStreamReader = new InputStreamReader(inStream);  
 BufferedReader BuffReader = new BufferedReader(inStreamReader);  
 int serverCode = 0;  
 String lineCode;  
 try {  
 lineCode = BuffReader.readLine();  
 }  
 catch (IOException except) {  
 return URLs;  
 }  
 Pattern patternCode = Pattern.*compile*("([234])[0-9]{2}");  
 Matcher matcherCode = patternCode.matcher(lineCode);  
 while (matcherCode.find()) {  
 serverCode = Integer.*parseInt*(lineCode.substring(matcherCode.start(), matcherCode.end() - 2));  
 }  
 if (serverCode == 2) {  
 while (true) {  
 String line;  
 try {  
 line = BuffReader.readLine();  
 }  
 catch (IOException except) {  
 return URLs;  
 }  
 if (line == null) {  
 break;  
 }  
 Pattern patternURL = Pattern.*compile*(  
 "[\"]" + "[https?://]{7,8}" + "([w]{3})?" + "[\\w\\.\\-]+" + "\\." + "[A-Za-z]{2,6}" + "[\\w\\.-/]\*" + "[\"]");  
 Matcher matcherURL = patternURL.matcher(line);  
 while (matcherURL.find()) {  
 String newLink = line.substring(matcherURL.start() + 1,  
 matcherURL.end() - 1);  
 URLs.add(newLink);  
 }  
 }  
 sock.close();  
 return URLs;  
 }  
 if (serverCode == 3) {  
 String newURL = "";  
 String tempLine;  
 while (true) {  
 try {  
 tempLine = BuffReader.readLine();  
 }  
 catch (IOException except) {  
 return URLs;  
 }  
 if (tempLine == null) {  
 break;  
 }  
 Pattern patternNewURL = Pattern.*compile*("(Location: ){1}[\\S]+");  
 Matcher matcherNewURL = patternNewURL.matcher(tempLine);  
 while (matcherNewURL.find()) {  
 newURL = tempLine.substring(matcherNewURL.start() + 10,  
 matcherNewURL.end());  
 }  
 }  
 if (newURL.equals(myDepthPair.getURL())) {  
 sock.close();  
 return URLs;  
 }  
 URLDepthPair newDepthPair;  
 newDepthPair = new URLDepthPair(newURL, myDepthPair.getDepth());  
 return *getAllLinks*(newDepthPair);  
 }  
 return URLs;  
 }  
}

***URLDepthPair.java***

package com.company;  
  
import java.net.\*;  
public class URLDepthPair {  
  
 private final String currentURL;  
 private final int currentDepth;  
  
 public URLDepthPair(String URL, int depth) {  
 currentURL = URL;  
 currentDepth = depth;  
 }  
  
 public String getURL() {  
 return currentURL;  
 }  
  
 public int getDepth() {  
 return currentDepth;  
 }  
  
 public String toString() {  
 String stringDepth = Integer.*toString*(currentDepth);  
 return stringDepth + '\t' + currentURL;  
 }  
  
 public String getDocPath() {  
 try {  
 URL url = new URL(currentURL);  
 return url.getPath();  
 }  
 catch (MalformedURLException e) {  
 return null;  
 }  
 }  
  
 public String getWebHost() {  
 try {  
 URL url = new URL(currentURL);  
 return url.getHost();  
 }  
 catch (MalformedURLException e) {  
 return null;  
 }  
 }  
}

**Выполнение программы:**



**Вывод:**В этой лабораторной работе — я создал веб-сканер, который 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 выводить в конце работы программы