**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

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

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе № 7

по дисциплине «Информационные технологии и программирование»

Выполнил: студент группы БФИ2102

Голубева Елена Александровна

Проверила:

Мосева Марина Сергеевна

Москва, 2022

Цель работы

Реализовать элементарный веб-сканер.

Задание

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.

2. Программа должна хранить URL-адрес в виде строки вместе с его глубиной (которая для начала будет равна 0).

3. Программа должна подключиться к указанному сайту в URL-адресе на порт 80 с использованием сокета и запросить указанную веб страницу.

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).

5. Далее программа должна закрыть соединение сокета с хостом.

6. Наконец, программа должна вывести все посещенные URL страницы вместе с их глубиной поиска.

Код файла Crawlers.java

package com.company;  
import java.net.\*;  
import java.util.\*;  
import java.io.\*;  
  
public class Crawlers {  
  
  
 public static void main(String[] args) {  
 args = new String[]{"http://go.com","2"};  
  
 int depth = 0; //текущая глубина  
  
  
 if (args.length != 2) { //проверяет на корректность длины входных данных  
 System.*out*.println("usage: java Crawler <URL> <depth>");  
 System.*exit*(1);  
 }  
 else { //исключение при невыполнении условия  
 try {  
 depth = Integer.*parseInt*(args[1]); //значение глубины из строки в целое значение  
 }  
 catch (NumberFormatException nfe) {  
  
 System.*out*.println("usage: java Crawler <URL> <depth>");  
 System.*exit*(1);  
 }  
 }  
  
 // список для представления ожидающих URL-адресов  
 LinkedList<URLDepthPair> pendingURLs = new LinkedList<URLDepthPair>();  
  
 // список для обработанных url-адресов  
 LinkedList<URLDepthPair> processedURLs = new LinkedList<URLDepthPair>();  
  
 // Пара URL-адресов с глубиной для представления веб-сайта, который пользователь ввел с глубиной 0  
 URLDepthPair currentDepthPair = new URLDepthPair(args[0], 0);  
  
 // добавление url-адреса в список ожидающих  
 pendingURLs.add(currentDepthPair);  
  
 // Динамический массив в который добавляется текущий url-адрес  
 ArrayList<String> seenURLs = new ArrayList<String>();  
 seenURLs.add(currentDepthPair.getURL());  
  
 // проверка каждого адреса пока список ожидающих непустой  
 while (pendingURLs.size() != 0) {  
  
 // получение следующего адреса из ожидающих и добавление к обработанным, сохранение его глубины  
 URLDepthPair depthPair = pendingURLs.pop();  
 processedURLs.add(depthPair);  
 int myDepth = depthPair.getDepth();  
  
 // получение всех ссылок с сайта и сохранение в новом списке ссылок  
 LinkedList<String> linksList = new LinkedList<String>();  
 linksList = Crawlers.*getAllLinks*(depthPair);  
  
 // проверка глубины данной и заданной  
 if (myDepth < depth) {  
 // перебор ссылок с сайта  
 for (int i=0;i<linksList.size();i++) {  
 String newURL = linksList.get(i);  
 // если ссылка уже встречалась  
 if (seenURLs.contains(newURL)) {  
 continue;  
 }  
 else {  
 //создание новой пары и добавление адреса в списки  
 URLDepthPair newDepthPair = new URLDepthPair(newURL, myDepth + 1);  
 pendingURLs.add(newDepthPair);  
 seenURLs.add(newURL);  
 }  
 }  
 }  
 }  
 // вывод обработанных ссылок с глубиной  
 Iterator<URLDepthPair> iter = processedURLs.iterator();  
 while (iter.hasNext()) {  
 System.*out*.println(iter.next());  
 }  
 }  
 // метод который принимает пару адрес и глубина и возвращает список  
 private static LinkedList<String> getAllLinks(URLDepthPair myDepthPair) {  
  
 // инициализация списка  
 LinkedList<String> URLs = new LinkedList<String>();  
  
 // инициализация сокета  
 Socket sock;  
  
 // создание нового сокета с адресом, парой и портом 80  
 try {  
 sock = new Socket(myDepthPair.getWebHost(), 80);  
 }  
  
 catch (UnknownHostException e) {  
 System.*err*.println("UnknownHostException: " + e.getMessage());  
 return URLs;  
 }  
 // возвращает пустой список  
 catch (IOException ex) {  
 System.*err*.println("IOException: " + ex.getMessage());  
 return URLs;  
 }  
  
 // Установка времени ожидания сокета  
 try {  
 sock.setSoTimeout(3000);  
 }  
 catch (SocketException exc) {  
 System.*err*.println("SocketException: " + exc.getMessage());  
 return URLs;  
 }  
  
 // строки для пути адреса из пары и для хоста  
 String docPath = myDepthPair.getDocPath();  
 String webHost = myDepthPair.getWebHost();  
  
 // Инициализация OutputStream позволяет сокету отправлять данные на другую стороны соединения  
 OutputStream outStream;  
  
  
 try {  
 outStream = sock.getOutputStream();  
 }  
  
 catch (IOException exce) {  
 System.*err*.println("IOException: " + exce.getMessage());  
 return URLs;  
 }  
  
 // инициализация PrintWriter, сброс после каждого вывода  
 PrintWriter myWriter = new PrintWriter(outStream, true);  
  
 // Отправка запроса на сервер  
 myWriter.println("GET " + docPath + "HTTP/1.1");  
 myWriter.println("Host: " + webHost+ ":80");  
 myWriter.println("Connection: close");  
 myWriter.println();  
  
 // Инициализация InputStream, позволяет получать данные с другой стороны  
 InputStream inStream;  
  
  
 try {  
 inStream = sock.getInputStream();  
 }  
  
 catch (IOException excep){  
 System.*err*.println("IOException: " + excep.getMessage());  
 return URLs;  
 }  
 // Создание новых InputStreamReader и BufferedReader для чтения строк с сервера  
 InputStreamReader inStreamReader = new InputStreamReader(inStream);  
 BufferedReader BuffReader = new BufferedReader(inStreamReader);  
  
  
  
 //чтение строк  
 while (true) {  
 String line;  
 try {  
 line = BuffReader.readLine();  
 }  
  
 catch (IOException except) {  
 System.*err*.println("IOException: " + except.getMessage());  
 return URLs;  
 }  
 // строки закончились  
 if (line == null)  
 break;  
  
  
 // переменные начального, конечного и текущего индекса ссылки  
 int beginIndex = 0;  
 int endIndex = 0;  
 int index = 0;  
  
 while (true) {  
  
 //константа для строки указывающей на ссылку  
 String URL\_INDICATOR = "href=\"";  
  
 //строка указывающая конец хоста  
 String END\_URL = "\"";  
  
  
 // индекс начала ссылки  
 index = line.indexOf(URL\_INDICATOR, index);  
 if (index == -1)  
 break;  
  
 // изменение текущего индекса и задание начального индекса  
 index += URL\_INDICATOR.length();  
 beginIndex = index;  
  
 // нахождение конца хоста(веб-узла) и присвоение текущему индексу значение конечного  
 endIndex = line.indexOf(END\_URL, index);  
 index = endIndex;  
  
 // установка ссылки меду начальным и конечным индексом и добавление адреса в список  
 String newLink = line.substring(beginIndex, endIndex);  
 URLs.add(newLink);  
 }  
  
 }  
 // возвращение списка  
 return URLs;  
 }  
}

Код класса URLDepthPair

package com.company;  
import java.net.\*; //для использования сокетов  
  
  
public class URLDepthPair { //Класс для представления пар [URL, depth]  
  
 private int currentDepth; //поле для глубины  
 private String currentURL; //поле для адреса  
  
  
 public URLDepthPair(String URL, int depth) { //конструктор задает текущий url-адрес и глубину  
 currentDepth = depth;  
 currentURL = URL;  
 }  
  
 public String getURL() {  
 return currentURL;  
 } //возвращает текущий url-адрес  
  
 public int getDepth() {  
 return currentDepth;  
 } //возвращает текущую глубину  
  
 public String toString() {  
 String stringDepth = Integer.*toString*(currentDepth); //возвращает глубину и url-адрес в виде строки  
 return stringDepth + '\t' + currentURL;  
 }  
  
 public String getDocPath() { //Возвращает путь данного url-адреса иначе вывод сообщения об ошибке  
 try {  
 URL url = new URL(currentURL);  
 return url.getPath();  
 }  
 catch (MalformedURLException e) {  
 System.*err*.println("MalformedURLException: " + e.getMessage());  
 return null;  
 }  
 }  
  
 public String getWebHost() { //возвращает хост для url-адреса или сообщение при ошибке  
 try {  
 URL url = new URL(currentURL);  
 return url.getHost();  
 }  
 catch (MalformedURLException e) {  
 System.*err*.println("MalformedURLException: " + e.getMessage());  
 return null;  
 }  
 }  
  
  
}

Вывод

В процессе выполнения данной лабораторной работы мы реализовали элементарный веб-сканер.

Ссылка на Github: https://github.com/ElenaGolubeva/Laboratory-7-Java

Список использованной литературы

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа,

2006.

1. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.