**Федеральное агентство связи**

**Государственное бюджетное образовательное учреждение высшего**

**образование**

**Ордена Трудового Красного Знамени**

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

**Кафедра «МКиИТ»**

**дисциплина «Объектно-ориентированное программирование»**

Отчет по Лабораторной работе №8

Подготовила студентка

группы БВТ1901: Нкурикийе Х

Проверил: Мосева М.С.

Москва 2021

**Crawler**

//package com.company;

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Crawler {

public static final String URL\_INDICATOR = "a href=\"";

public static final String URL\_ENDING = "\"";

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int maxDepth = 0;

int threads = 0;

String[] strings;

System.out.println("usage: <URL> <depth> <threads> (0 - for exit)");

while (true) {

strings = br.readLine().split(" ");

if (strings[0].equals("0"))return;

if (strings.length != 3) {

System.out.println("usage: <URL> <depth> <threads> (0 - for exit)");

} else {

try {

maxDepth = Integer.parseInt(strings[1]);

threads = Integer.parseInt(strings[2]);

} catch (NumberFormatException e) {

System.out.println("usage: <URL> <depth> <threads> (0 - for exit)");

}

}

break;

}

URLPool pool = new URLPool(maxDepth);

pool.put(new URLDepthPair(strings[0], 0));

int initialActive = Thread.activeCount();

while (pool.getWaitThreads() != threads) {

if (Thread.activeCount() - initialActive < threads) {

CrawlerTask crawler = new CrawlerTask(pool);

new Thread(crawler).start();

}

else {

try {

Thread.sleep(100);

}

catch (InterruptedException e) {

System.out.println("InterruptedException, ignoring...");

}

}

}

for (URLDepthPair pair:pool.processedURLs) {

System.out.println(pair);

}

System.exit(1);

}

public static ArrayList<String> getURLs(URLDepthPair pair){

ArrayList<String> URLs = new ArrayList<String>();

Socket socket;

try {

socket = new Socket(pair.getHost(),80);

} catch (UnknownHostException e) {

System.out.println("UnknownHostException: " + e.getMessage());

return URLs;

} catch (IOException e) {

System.out.println("IOException: " + e.getMessage());

return URLs;

}

try {

socket.setSoTimeout(3000);

} catch (SocketException e) {

System.out.println("SocketException: " + e.getMessage());

return URLs;

}

InputStream inputStream;

OutputStream outputStream;

try {

outputStream = socket.getOutputStream();

} catch (IOException e) {

System.out.println("IOException: " + e.getMessage());

return URLs;

}

PrintStream printStream = new PrintStream(outputStream, true);

printStream.println("GET " + pair.getPath() + " HTTP/1.1");

printStream.println("Host: " + pair.getHost() + ":80");

printStream.println("Connection: close");

printStream.println();

try {

inputStream = socket.getInputStream();

} catch (IOException e) {

System.out.println("IOException: " + e.getMessage());

return URLs;

}

InputStreamReader in = new InputStreamReader (inputStream);

BufferedReader reader = new BufferedReader(in);

while (true){

String string;

int beginIndex = 0;

int endIndex = 0;

try {

string = reader.readLine();

} catch (IOException e) {

System.out.println("IOException: " + e.getMessage());

return URLs;

}

if(string == null)return URLs;

while (true){

beginIndex = string.indexOf(URL\_INDICATOR,beginIndex);

if(beginIndex == -1)break;

//if(string.contains("https://"))break;

beginIndex += URL\_INDICATOR.length();

endIndex = string.indexOf(URL\_ENDING, beginIndex);

String temp = string.substring(beginIndex,endIndex);

if(!temp.contains("http"))temp = "https://" + pair.getHost() + temp;

URLs.add(temp);

beginIndex = endIndex;

}

}

}

}

**URLPool**

//package com.company;

import java.util.ArrayList;

import java.util.LinkedList;

public class URLPool {

private LinkedList<URLDepthPair> inQueueURLs = new LinkedList <URLDepthPair>();

public LinkedList <URLDepthPair> processedURLs = new LinkedList <URLDepthPair>();

private ArrayList<String> seenURLs = new ArrayList<String>();

public int waitingThreads;

private int maxDepth;

URLPool(int maxDepth){

this.maxDepth = maxDepth;

waitingThreads = 0;

}

public synchronized void put(URLDepthPair depthPair) {

if (depthPair.getDepth() < maxDepth && !seenURLs.contains(depthPair.getUrl())) {

inQueueURLs.addLast(depthPair);

seenURLs.add(depthPair.getUrl());

if (waitingThreads > 0) waitingThreads--;

this.notify();

}

}

public synchronized URLDepthPair get() {

if (inQueueURLs.size() == 0) {

waitingThreads++;

try {

this.wait();

}

catch (InterruptedException e) {

System.err.println("MalformedURLException: " + e.getMessage());

return null;

}

}

URLDepthPair myDepthPair = inQueueURLs.removeFirst();

processedURLs.add(myDepthPair);

return myDepthPair;

}

public synchronized int getWaitThreads() {

return waitingThreads;

}

public synchronized ArrayList<String> getSeenList() {

return seenURLs;

}

}

**CrawlerTask**

//package com.company;

import java.util.ArrayList;

public class CrawlerTask implements Runnable {

public URLDepthPair pair;

public URLPool pool;

CrawlerTask(URLPool pool) {

this.pool = pool;

}

public void run() {

pair = pool.get();

ArrayList<String> links = new ArrayList<String>();

links = Crawler.getURLs(pair);

for (String link : links) {

URLDepthPair newPair = new URLDepthPair(link, pair.getDepth() + 1);

if (!pool.getSeenList().contains(link)) {

pool.put(newPair);

}

}

}

}

**URlDepthPair**

//package com.company;

import java.net.MalformedURLException;

import java.net.URL;

public class URLDepthPair {

private String url;

private String path;

private String host;

private int depth;

URLDepthPair(String url, int depth){

this.url = url;

this.depth = depth;

try {

URL myurl = new URL(url);

path = myurl.getPath();

host = myurl.getHost();

if(path.length() == 0 || path.charAt(path.length() - 1) != '/'){

path += "/";

}

} catch (MalformedURLException e) {

System.out.println("MalformedURLException: " + e.getMessage());

}

}

@Override

public String toString() {

return depth + " " + url;

}

public int getDepth() {

return depth;

}

public String getUrl() {

return url;

}

public String getHost() {

return host;

}

public String getPath() {

return path;

}

}



