ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени государственное образовательное учреждение высшего образования

Московский технический университет связи и информатики

Кафедра информатики

**ЛАБОРАТОРНАЯ РАБОТА № 8**

## Модифицированный веб-сканер

Выполнил студент

Группы БСТ1801

КОЛИ.Ж. П.

Москва-2020

**Crawler**

import java.io.\*;

import java.net.\*;

import java.util.LinkedList;

import java.util.Scanner;

public class Crawler {

public static void main(String[] args) {

LinkedList<URLDepthPair> visited = new LinkedList<>();

LinkedList<URLDepthPair> urls = new LinkedList<>();

LinkedList<URLDepthPair> finded = new LinkedList<>();

int maxDepth = Integer.parseInt(args[1]);

urls.add(new URLDepthPair(args[0], 0));

while (!urls.isEmpty()) {

for (URLDepthPair pair : urls) {

if (!visited.contains(pair) && pair.getDepth() <= maxDepth) {

System.out.println(pair);

finded.addAll(Crawler.getSites(pair));

visited.add(pair);

}

}

urls.clear();

urls.addAll(finded);

finded.clear();

}

}

public static LinkedList<URLDepthPair> getSites(URLDepthPair pair) {

LinkedList<URLDepthPair> links = new LinkedList<>();

try {

URL url = new URL(pair.getUrl());

Scanner scanner = new Scanner(url.openStream());

while (scanner.hasNextLine()) {

String line = scanner.nextLine();

parseLine(links, line, pair.getDepth());

}

} catch (IOException e) {

System.out.println("Cannot connect: " + pair.getUrl());

}

return links;

}

public static void parseLine(LinkedList<URLDepthPair> links, String line, int depth) {

if (line.contains("<a")) {

line = line.substring(line.indexOf("<a"));

if (line.contains("href=")) {

line = line.substring(line.indexOf("href="));

try {

line = line.substring(line.indexOf(URLDepthPair.URL\_PREFIX));

String suburl = line.substring(0, line.indexOf("\""));

links.add(new URLDepthPair(suburl, depth + 1));

line = line.substring(suburl.length());

parseLine(links, line, depth);

} catch (StringIndexOutOfBoundsException ignored) { }

}

}

}

}

**CrawlerTask**

public class CrawlerTask implements Runnable {

private final URLPool pool;

private final int threadsAmount;

public CrawlerTask(URLPool pool, int threadsAmount) {

this.pool = pool;

this.threadsAmount = threadsAmount;

}

@Override

public void run() {

// System.out.println(Thread.currentThread().getName() + ": Thread runned");

while (pool.getWaitingAmount() < threadsAmount) {

URLDepthPair pair = pool.first();

// System.out.println(Thread.currentThread().getName() + ": " + pair);

if (pair != null) {

for (URLDepthPair urlPair : Crawler.getSites(pair)) {

pool.add(urlPair);

}

} else {

try {

synchronized (pool) {

pool.incWaitingAmount();

System.out.println(pool.getWaitingAmount());

if (pool.getWaitingAmount() == threadsAmount) {

pool.notifyAll();

} else {

// System.out.println(Thread.currentThread().getName() + ": in waiting");

pool.wait();

// System.out.println(Thread.currentThread().getName() + ": wake up");

}

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

**URLDepthPair**

public class URLDepthPair {

public static final String URL\_PREFIX = "https://";

private String url;

private int depth;

public URLDepthPair(String url, int depth) {

this.url = url;

this.depth = depth;

}

public String getUrl() {

return url;

}

public int getDepth() {

return depth;

}

@Override

public String toString() {

return depth + " - " + url;

}

@Override

public boolean equals(Object obj) {

if (obj instanceof URLDepthPair) {

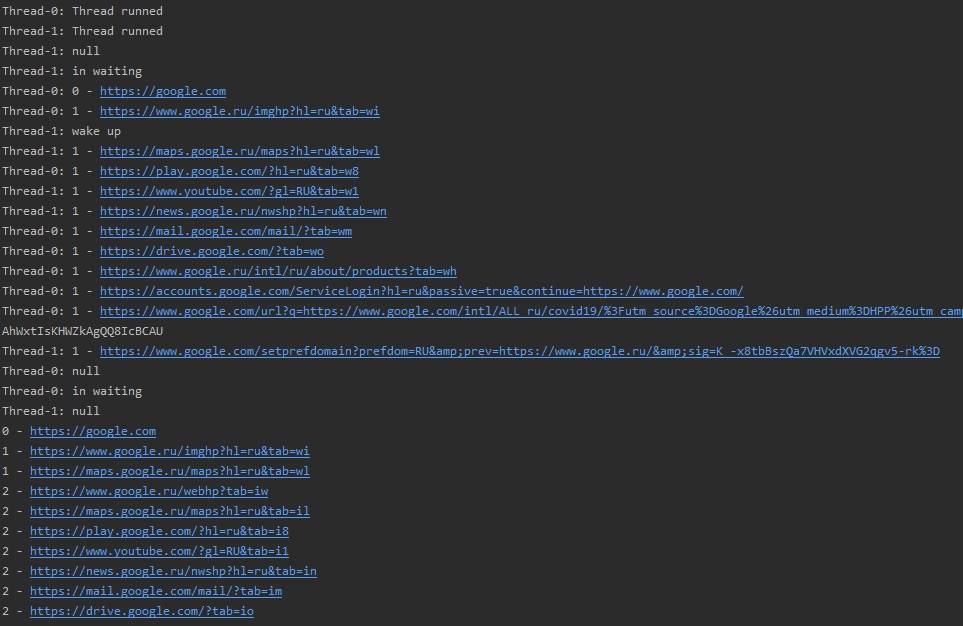
return url.equals(((URLDepthPair) obj).url);

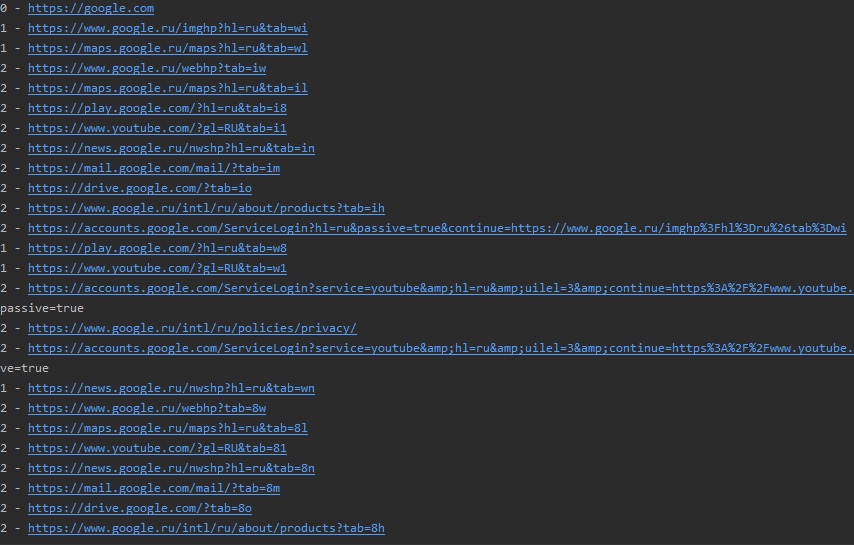
}

return false;

}

loSbaq5g1bU.jpg}





Запускаем

потом в консоль выписывается какую сылку берет поток, в примере их два когда они закончат работау то выводиться список все пройденых ссылок