**Федеральное агентство связи**

**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 №8

по дисциплине «Объектно-ориентированное программирование»

Выполнил: студент группы БВТ2002

ФИО: Мосева Алеся Сергеевна

Проверил:Полянцева Ксения Андреевна

Москва, 2020

**Цель работы**: расширить сканер для использования поточной обработки Java так, чтобы несколько веб-страниц можно было сканировать параллельно..

**Код программы:**

import java.io.\*;

import java.net.\*;

import java.util.LinkedList;

public class Crawler {

public static void showResult(LinkedList<URLDepthPair> list){

for (URLDepthPair c:list){

System.out.println("Depth : "+c.getDepth() + "\tLink : "+c.getUrl());

}

}

public static boolean checkDigit(String a){

boolean isDigit = true;

for (int i = 0; i < a.length(); i ++){

isDigit = Character.isDigit(a.charAt(i));

}

return isDigit;

}

public static void main(String[] args) {

// write your code here

String[] arg = new String[]{"http://government.ru/","2", "3"};

if ((arg.length == 3) && checkDigit(arg[1])&&checkDigit(arg[2])) {

String url = arg[0];

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

int numThreads = Integer.parseInt(arg[2]);

URLPool pool = new URLPool(maxDepth);

pool.addPair(new URLDepthPair(url,0));

for(int i = 0; i < numThreads; i ++){

CrawlerTask c = new CrawlerTask(pool);

Thread t = new Thread(c);

t.start();

}

while(numThreads != pool.getWait()){

try{

Thread.sleep(500);

}

catch (InterruptedException e){}

}

try{

showResult(pool.getViewedLinks());

}

catch(NullPointerException e){

System.out.println("No Links");

}

System.exit(0);

}

else System.out.println("usage: java Crawler <URL> <maximum\_depth> <num\_threads>");

}

}

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.MalformedURLException;

import java.net.Socket;

import java.net.SocketTimeoutException;

import static com.company.URLDepthPair.URL\_PREFIX;

public class CrawlerTask implements Runnable {

URLPool urlPool;

public CrawlerTask(URLPool pool){

urlPool = pool;

}

public static void request(PrintWriter out, URLDepthPair pair) throws MalformedURLException {

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

out.println("Host: " + pair.getHost());

out.println("Connection: close");

out.println();

out.flush();

}

@Override

public void run() {

while (true) {

URLDepthPair currentPair = urlPool.getPair();

try {

Socket my\_socket = new Socket(currentPair.getHost(), 80);

my\_socket.setSoTimeout(1000);

try {

PrintWriter out = new PrintWriter(my\_socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(my\_socket.getInputStream()));

request(out,currentPair);

String line;

while ((line = in.readLine())!=null){

if (line.indexOf(URL\_PREFIX) != -1 && line.indexOf('"') != -1) {

StringBuilder currentLink = new StringBuilder();

int i = line.indexOf(URL\_PREFIX);

while (line.charAt(i) != '"' && line.charAt(i) != ' ') {

if (line.charAt(i) == '<') {

currentLink.deleteCharAt(currentLink.length() - 1);

break;

}

else {

currentLink.append(line.charAt(i));

i++;

}

}

URLDepthPair newPair = new URLDepthPair(currentLink.toString(), currentPair.depth + 1);

urlPool.addPair(newPair);

}

}

my\_socket.close();

} catch (SocketTimeoutException e) {

my\_socket.close();

}

}

catch (IOException e) {}

}

}

}

import java.net.MalformedURLException;

import java.net.URL;

import java.util.LinkedList;

//класс, который хранит url и глубину, на которой находится этот сайт

public class URLDepthPair {

//поддерживаемый протокол

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

//url адрес

public String url;

//глубина, на которой находится url

public int depth;

public URLDepthPair (String url, int depth) {

this.url = url;

this.depth = depth;

}

public int getDepth() {

return depth;

}

public String getUrl() {

return url;

}

public void setDepth(int depth) {

this.depth = depth;

}

public void setUrl(String url) {

this.url = url;

}

public String getHost() throws MalformedURLException {

URL host = new URL(url);

return host.getHost();

}

public String getPath() throws MalformedURLException {

URL path = new URL(url);

return path.getPath();

}

@Override

public String toString() {

return "URLDepthPair{" +

"url='" + url + '\'' +

", depth=" + depth +

'}';

}

public static boolean check(LinkedList<URLDepthPair> resultLink, URLDepthPair pair) {

boolean isAlready = true;

for (URLDepthPair c : resultLink)

if (c.getUrl().equals(pair.getUrl()))

isAlready=false;

return isAlready;

}

}

import java.util.LinkedList;

public class URLPool {

LinkedList<URLDepthPair> viewedLinks;

LinkedList<URLDepthPair> foundLink;

int maxDepth;

int cWait;

public URLPool(int maxDepth){

this.maxDepth = maxDepth;

foundLink = new LinkedList<URLDepthPair>();

viewedLinks = new LinkedList<URLDepthPair>();

cWait = 0;

}

public synchronized URLDepthPair getPair(){

while(foundLink.size() == 0){

cWait++;

try{

wait();

}

catch (InterruptedException e){

System.out.println("Ignoring Interrupted Exception");

}

cWait--;

}

URLDepthPair current = foundLink.removeFirst();

return current;

}

public synchronized void addPair(URLDepthPair pair){

if(URLDepthPair.check(viewedLinks,pair)){

viewedLinks.add(pair);

if (pair.getDepth()<maxDepth){

foundLink.add(pair);

notify();

}

}

}

public int getWait() {

return cWait;

}

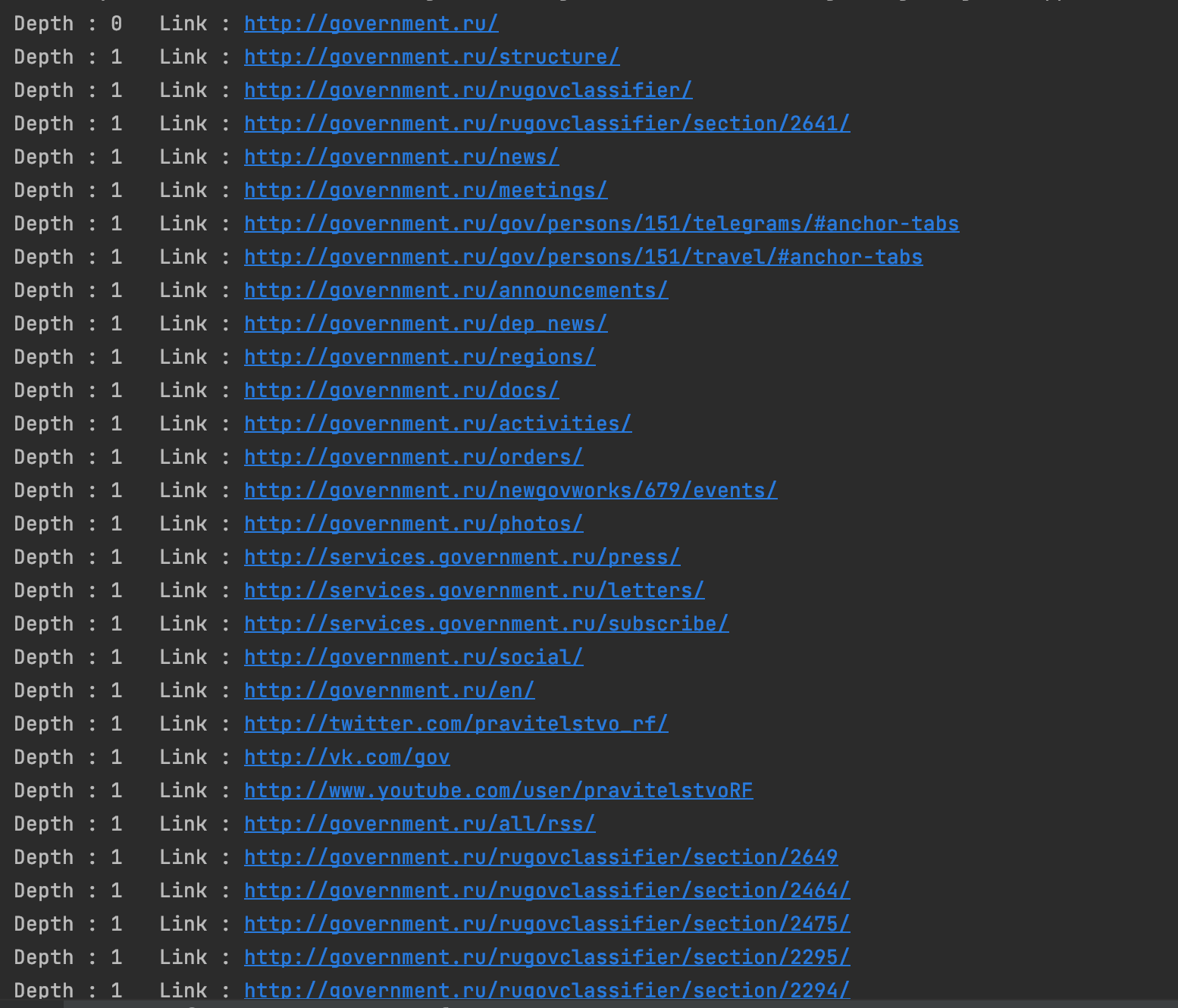
public LinkedList<URLDepthPair> getViewedLinks(){

return viewedLinks;

}

}

**Скриншоты выполнения:**

****

**Заключение:**

Таким, образом мы реализовали многопоточность, чтобы веб-сканнер мог работать параллельно с несколькими страницами.