**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

**Кафедра Информатики**

****

**Отчет по лабораторной работе №8**

по предмету «КТП»:

Выполнил: студент группы БВТ1802

Самаков Владислав Владимирович

Руководитель:

Ксения Андреевна Полянцева

Москва 2020

**1 Цель работы**

Цель работы: сделать web-сканер многопоточным.

**2 Задание**

Написать программу, которая будет получать в аргументах командной строки URL-адрес, глубину поиска и количество потоков и посещать все ссылки, которые находятся на исходной web-странице в пределах указанной глубины поиска.

**3 Текст программы**

**ScannerApp.java**

import java.io.IOException;  
  
public class ScannerApp {  
 public static void main(String args[]) throws IOException, InterruptedException {  
 URLPool pool = new URLPool(args[0], Integer.*parseInt*(args[1]), Integer.*parseInt*(args[2]));  
 for (int i = 0; i < Integer.*parseInt*(args[2]); i++) {  
 CrawlerTask crawler = new CrawlerTask(pool);  
 new Thread(crawler).start();  
 }  
 }  
}

**CrawlerTask.java**

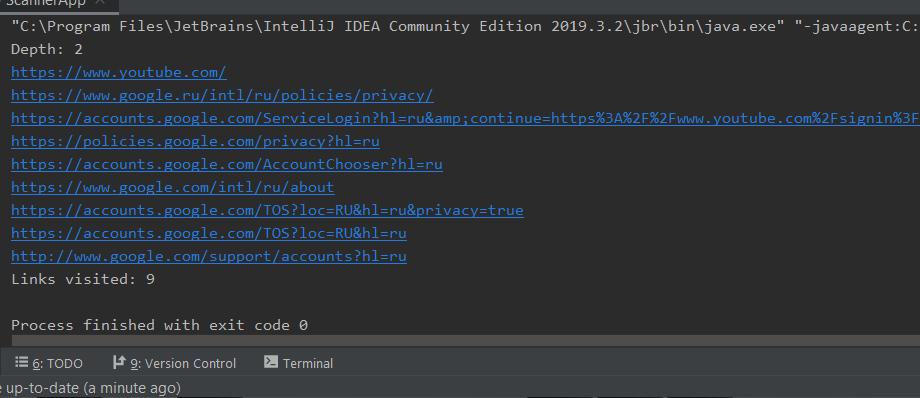
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.net.URL;  
import java.net.URLConnection;  
  
  
public class CrawlerTask implements Runnable {  
  
  
 final static int *AnyDepth* = 0;  
  
 private URLPool pool;  
  
 private String prefix = "http";  
  
 @Override  
 public void run() {  
 try {  
 Scan();  
 } catch (IOException | InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public CrawlerTask(URLPool pool) {  
 this.pool = pool;  
 }  
  
 private void Scan() throws IOException, InterruptedException {  
 while (true) {  
 Process(pool.get());  
  
 }  
 }  
  
 private void Process(URLDepthPair pair) throws IOException{  
  
 URL url = new URL(pair.getURL());  
 URLConnection connection = url.openConnection();  
  
 String redirect = connection.getHeaderField("Location");  
 if (redirect != null) {  
 connection = new URL(redirect).openConnection();  
 }  
  
 pool.addProcessed(pair);  
 if (pair.getDepth() == 0) return;  
  
  
 BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
 String input;  
 while ((input = reader.readLine()) != null) {  
 while (input.contains("a href=\"" + prefix)) {  
 input = input.substring(input.indexOf("a href=\"" + prefix) + 8);  
 String link = input.substring(0, input.indexOf('\"'));  
 if(link.contains(" "))  
 link = link.replace(" ", "%20");  
  
 if (pool.getNotProcessed().contains(new URLDepthPair(link, *AnyDepth*)) ||  
 pool.getProcessed().contains(new URLDepthPair(link, *AnyDepth*))) continue;  
 pool.addNotProcessed(new URLDepthPair(link, pair.getDepth() - 1));  
 }  
 }  
 reader.close();  
  
 }  
  
  
}

**URLPool.java**

import java.util.LinkedList;  
  
public class URLPool {  
  
 private LinkedList<URLDepthPair> processed = new LinkedList<URLDepthPair>();  
 private LinkedList<URLDepthPair> notProcessed = new LinkedList<URLDepthPair>();  
 private int depth;  
 private int waiting;  
 private int threads;  
  
 public URLPool(String url, int depth, int threads) {  
 notProcessed.add(new URLDepthPair(url, depth));  
 this.depth = depth;  
 this.threads = threads;  
 }  
  
 private boolean isEmpty() {  
 if (notProcessed.size() == 0) return true;  
 return false;  
 }  
  
 public void getSites() {  
 System.*out*.println("Depth: " + depth);  
 for (int i = 0; i < processed.size(); i++) {  
 System.*out*.println( depth - processed.get(i).getDepth() + " " + processed.get(i).getURL());  
 }  
 System.*out*.println("Links visited: " + processed.size());  
 }  
  
 public synchronized URLDepthPair get() throws InterruptedException {  
 if (isEmpty()) {  
 waiting++;  
 if (waiting == threads) {  
 getSites();  
 System.*exit*(0);  
 }  
 wait();  
 }  
 return notProcessed.removeFirst();  
 }  
 public synchronized void addNotProcessed(URLDepthPair pair) {  
 notProcessed.add(pair);  
 if (waiting > 0) {  
 waiting--;  
 notify();  
 }  
 }  
  
 public void addProcessed(URLDepthPair pair) {  
 processed.add(pair);  
 }  
  
 public LinkedList<URLDepthPair> getProcessed()  
 {  
 return processed;  
 }  
  
 public LinkedList<URLDepthPair> getNotProcessed()  
 {  
 return notProcessed;  
 }  
  
}

**4 Работа программы**

Стартовый url - <https://www.youtube.com/> Глубина поиска - 2



Вывод

С помощью средств языка программирования java я улучшил web-сканер, сделав его многопоточным. Время работы программы значительно уменьшилось, так как теперь обработкой ссылок занимаются несколько потоков.