**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

Кафедра Математической кибернетики и информационных технологий

**Отчет по лабораторной работе №8**

по дисциплине «Технологии разработки программного обеспечения»

Выполнил: студент группы БВТ1801

Клюшкин Дмитрий Алексеевич

Руководитель:

Мосева Марина Сергеевна

Москва 2020

**Цель работы:** расширить сканер для использования поточной обработки.

**Выполнение:**

**Socket:**

import java.io.\*;  
  
public class Socket {  
 java.net.Socket socket;  
 Socket (String host, int port) throws IOException {  
 socket = new java.net.Socket(host,port);  
 socket.setSoTimeout(5000);  
 }  
 void setSoTimeout(int timeout)throws IOException {  
 socket.setSoTimeout(timeout);  
 }  
 InputStream getInputStream()throws IOException {  
 return socket.getInputStream();  
 }  
 OutputStream getOutputStream()throws IOException{  
 return socket.getOutputStream();  
 }  
 void close()throws IOException{  
 socket.close();  
 }  
}

**URLDepthPair:**

import java.net.\*;  
  
public class URLDepthPair {  
 String url;  
 int depth;  
  
 URLDepthPair(String url, int depth){  
 this.url=url;  
 this.depth=depth;  
 }  
  
 public boolean isURL(){  
 return url.matches("\\b(http)://[-a-zA-Z0-9+&@#/%?=~\_|!:,.;]\*[-a-zA-Z0-9+&@#/%=~\_|]");  
 }  
  
 public String getUrl() {  
 try {  
 URL url = new URL(this.url);  
 return url.getHost();  
 }  
 catch (MalformedURLException e) {  
 System.*err*.println("MalformedURLException: " + e.getMessage());  
 return null;  
 }  
 }  
  
 public int getDepth() {  
 return depth;  
 }  
 public String toString(){  
 return url+" ["+depth+"]";  
 }  
}

**Crawler:**

import java.io.\*;  
import java.util.LinkedList;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Crawler {  
 Socket socket;  
 BufferedReader bufferedReader;  
 PrintWriter printWriter;  
 LinkedList<URLDepthPair> siteList;  
 int depth;  
 Pattern regHTTP;  
  
 public LinkedList <URLDepthPair> work(URLDepthPair urlDepthPair,int port) throws IOException{  
 try {  
 //init  
 socket = new Socket(urlDepthPair.getUrl(), port);  
 bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 printWriter = new PrintWriter(socket.getOutputStream(), true);  
 depth = urlDepthPair.depth;  
 siteList = new LinkedList<URLDepthPair>();  
 regHTTP = Pattern.*compile*("(http:\\/\\/[\\w\\-\\.!~?&=+\\\*'(),\\/\\#\\:]+)((?!\\<\\/\\w\\>))\*?");  
 }  
 catch (Exception exc){  
 System.*out*.println(exc);  
 return new LinkedList<URLDepthPair>();  
 }  
 //end  
 //connection  
 printWriter.println("GET / HTTP/1.1");  
 printWriter.println("Host: "+urlDepthPair.getUrl()+":"+port);  
 printWriter.println("Connection: Close");  
 printWriter.println();  
 //end  
 try{  
 String line;  
 while ((line=bufferedReader.readLine())!=null) {  
 while(line.contains("<a")){  
 while (line.indexOf(">", line.indexOf("<a"))==-1) line+=bufferedReader.readLine();  
  
 String http = line.substring(line.indexOf("<a"),line.indexOf(">", line.indexOf("<a")));  
 if (http.contains("http://")){  
 Matcher matcher = regHTTP.matcher(http);  
 matcher.find();  
 String url = matcher.group();  
 siteList.add(new URLDepthPair(url,depth+1));  
 }  
 line=line.replace(http,"");  
 }  
 }  
 }  
 catch (IOException except){  
 System.*out*.println(except);  
 }  
 socket.close();  
 return siteList;  
 }  
}

**Scaner:**

import java.util.\*;  
public class Scaner {  
 static LinkedList<URLDepthPair> *site*;  
  
 public static void main(String[] args){  
 String startURL=args[0];  
 String fDepth=args[1];  
 String cThread=args[2];  
 //String startURL="http://www.google.com";  
 //String fDepth="3";  
 //String cThread="4";  
  
 URLDepthPair startUrl = new URLDepthPair(startURL,0);  
  
 if (fDepth.matches("\\D+") || !startUrl.isURL() || cThread.matches("\\D+")) {  
 System.*out*.println("java Crawler <" + startURL + "><" + fDepth + "><" + cThread + ">");  
 return;  
 }  
  
 int finalDepth=Integer.*parseInt*(fDepth);  
 int countThread=Integer.*parseInt*(cThread);  
  
  
 URLPool pool = new URLPool(startUrl,finalDepth);  
  
 LinkedList<Thread> threadList = new LinkedList<>();  
  
 for (int i=0;i<countThread;i++){  
 CrawlerTask crawlerTask = new CrawlerTask(pool);  
 threadList.add(new Thread(crawlerTask));  
 threadList.getLast().start();  
 }  
  
 while (pool.getWaitThreads()!=countThread){  
 System.*out*.print("");  
 }  
  
 *site*=pool.getSite();  
  
 for(URLDepthPair url: *site*){  
 System.*out*.println(url);  
 }  
  
 for(Thread thread: threadList){  
 thread.stop();  
 }  
 }  
}

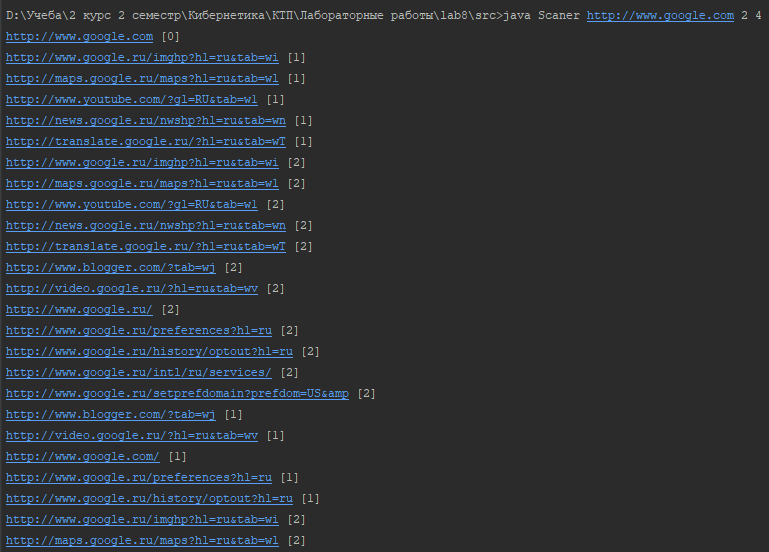
**URLPool:**

import java.util.\*;  
  
public class URLPool {  
 private LinkedList<URLDepthPair> untreated; //необработанные  
 private LinkedList<URLDepthPair> treated; //обработанные  
  
 private int finalDepth;  
  
 private int waitThreads = 0;  
  
 URLPool(URLDepthPair url,int depth){  
 finalDepth=depth;  
 untreated = new LinkedList<URLDepthPair>();  
 treated = new LinkedList<URLDepthPair>();  
 untreated.add(url);  
 }  
  
 public synchronized int getWaitThreads() {  
 return waitThreads;  
 }  
  
 public synchronized URLDepthPair getURL(){  
 if (untreated.size() == 0) {  
 try {  
 waitThreads++;  
 this.wait();  
 }  
 catch (InterruptedException e) {  
 System.*err*.println("MalformedURLException: " + e.getMessage());  
 return null;  
 }  
 }  
 treated.add(untreated.getFirst());  
 return untreated.removeFirst();  
 }  
  
 public synchronized void addListURL(LinkedList<URLDepthPair> URLs){  
  
 if (URLs.size()!=0){  
 if (URLs.getFirst().depth>=finalDepth){  
 treated.addAll(URLs);  
 }  
 else{  
 untreated.addAll(URLs);  
 for (int countSite=URLs.size(); countSite!=0 && waitThreads!=0;countSite-- , waitThreads--){  
 this.notify();  
 }  
 }  
 }  
 }  
  
 public LinkedList<URLDepthPair> getSite(){  
 return treated;  
 }  
}

**CrawlerTask:**

import java.io.IOException;  
  
public class CrawlerTask implements Runnable {  
  
 private URLPool urlPool;  
 private URLDepthPair urlDepthPair;  
 private Crawler crawler;  
  
 public CrawlerTask(URLPool pool) {  
 urlPool = pool;  
 crawler = new Crawler();  
 }  
  
 public void run() {  
 while (true) {  
 urlDepthPair = urlPool.getURL();  
 try {  
 urlPool.addListURL(crawler.work(urlDepthPair, 80));  
 } catch (IOException e) {  
 System.*out*.println(e);  
 }  
 }  
 }  
}

**Скриншот работы программы:**



Вывод: расширили сканер для использования поточной обработки Java так, чтобы несколько веб-страниц можно было сканировать параллельно.