**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**

по предмету «Кроссплатформенные технологии программирования»

на тему:

«Модифицированный веб-сканер»

Выполнил: студент группы

Митрохин Ярослав Игоревич

Руководитель:

Мосева Марина Сергеевна

Москва 2020

*Цель работы:* реализовать элементарный веб-сканер. Сканер будет автоматически загружать веб-страницы из интернета, искать новые ссылки на этих страницах и повторять их. Расширить сканер для использования поточной обработки Java так, чтобы несколько веб-страниц можно было сканировать параллельно.

*Ход работы:*

*Реализация классов: Socket*

public class Socket {  
 java.net.Socket socket;  
 Socket(String host, int port) throws IOException  
 {  
 socket = new java.net.Socket(host,port);  
 socket.setSoTimeout(4000);  
 }  
 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();  
 }  
}

*Crawler:*

public class Crawler {  
 LinkedList<URLDepthPair> curList;  
 PrintWriter printWriter;  
 BufferedReader bufferedReader;  
 Pattern regHTTP;  
 Socket socket;  
 public static void main(String[] args) throws IOException{  
 LinkedList<URLDepthPair> curSite=new LinkedList<>();  
 LinkedList<URLDepthPair> newSites=new LinkedList<>();  
 int depth = 2;  
 newSites.add(new URLDepthPair("http://www.google.com",0));  
 //newSites.add(curSite.getFirst());  
 while (newSites.size()!=0){  
 Crawler crawler = new Crawler(newSites.getFirst());  
 if (newSites.getFirst().getcDepth()!=depth) {  
 newSites.addAll(crawler.getUrl());  
 }  
 else  
 {  
 curSite.addAll(crawler.getUrl());  
 }  
 curSite.add(newSites.removeFirst());  
 }  
  
 for(URLDepthPair url: curSite)  
 {  
 System.*out*.println(url);  
 }  
 }  
  
 Crawler (URLDepthPair pair) throws IOException{  
 try {  
 socket = new Socket(pair.getSURL(), 80);  
 bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 printWriter = new PrintWriter(socket.getOutputStream(), true);  
 curList = new LinkedList<URLDepthPair>();  
 regHTTP = Pattern.*compile*("(http:\\/\\/[\\w\\-\\.!~?&=+\\\*'(),\\/\\#\\:]+)((?!\\<\\/\\w\\>))\*?");//true url or not  
 }  
 catch (Exception exc){  
 System.*out*.println(exc);  
 return;  
 }  
 printWriter.println("GET / HTTP/1.1");  
 printWriter.println("Host: "+pair.getSURL()+":80");  
 printWriter.println("Connection: Close");  
 printWriter.println();  
  
 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();  
 curList.add(new URLDepthPair(url,pair.getcDepth()+1));  
 }  
 line=line.replace(http,"");  
 }  
 }  
 }  
 catch (IOException except){  
 System.*out*.println(except);  
 }  
 socket.close();  
 }  
 LinkedList<URLDepthPair> getUrl()  
 {  
 return curList;  
 }  
  
  
}

*URLDepthPair:*

public class URLDepthPair {  
 private String SURL;  
 private int cDepth;  
 public String toString()  
 {  
 String strDe = Integer.*toString*(cDepth);  
 return strDe + " " +SURL;  
 }  
 public boolean isthisURL() throws MalformedURLException  
 {  
 try{  
 URL url = new URL(SURL);  
 return true;  
 }catch (Exception e)  
 {  
 System.*err*.println("MalformedURLException: "+e.getMessage());  
 return false;  
 }  
 }  
 public URLDepthPair(String URL, int depth)  
 {  
 this.SURL=URL;  
 this.cDepth=depth;  
 }  
  
 public int getcDepth() {  
 return cDepth;  
 }  
  
 public String getSURL()  
 {  
 try {  
 URL curl = new URL(SURL);  
 return curl.getHost();  
 } catch (MalformedURLException e)  
 {  
 System.*err*.println("MalformedURLException: "+e.getMessage());  
 return null;  
 }  
 }  
}

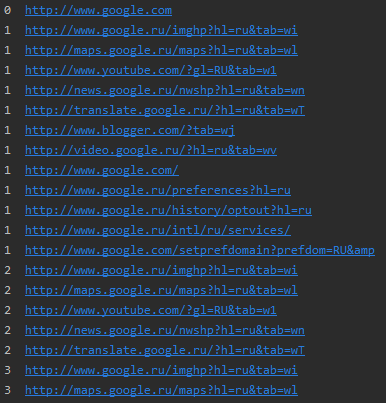
*CrawlerTask:*

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

*URLPool:*

public class URLPool {  
 private LinkedList<URLDepthPair> nopeList; //необработанные  
 private LinkedList<URLDepthPair> yesList; //обработанные  
 private int finalDepth;  
 private int waitThreads = 0;  
 public static void main(String[] args) throws IOException {  
 //String url = args[0];  
 //int finishDepth = Integer.parseInt(args[1]);  
 //int CountThre = Integer.parseInt(args[2]);  
 String url = "http://www.google.com";  
 int finishDepth = 3;  
 int CountThre = 4;  
 URLPool pool = new URLPool(new URLDepthPair(url,0),finishDepth);  
 LinkedList<Thread> threadList = new LinkedList<>();  
  
 for (int i=0;i<CountThre;i++){  
 CrawlerTask crawlerTask = new CrawlerTask(pool);  
 threadList.add(new Thread(crawlerTask));  
 threadList.getLast().start();  
 }  
  
 while (pool.getWaitThreads()!=CountThre){}  
  
 LinkedList<URLDepthPair> site=pool.getSite();  
  
 for(URLDepthPair iurl: site){  
 System.*out*.println(iurl);  
 }  
  
 for(Thread thread: threadList){  
 thread.stop();  
 }  
 }  
  
 URLPool(URLDepthPair url,int depth){  
 finalDepth=depth;  
 nopeList = new LinkedList<URLDepthPair>();  
 yesList = new LinkedList<URLDepthPair>();  
 nopeList.add(url);  
 }  
  
 public synchronized int getWaitThreads() {  
 return waitThreads;  
 }  
  
 public synchronized URLDepthPair getURL(){  
 if (nopeList.size() == 0) {  
 try {  
 waitThreads++;  
 this.wait();  
 }  
 catch (InterruptedException e) {  
 System.*err*.println("MalformedURLException: " + e.getMessage());  
 return null;  
 }  
 }  
 yesList.add(nopeList.getFirst());  
 return nopeList.removeFirst();  
 }  
  
 public synchronized void addListURL(LinkedList<URLDepthPair> URLs){  
  
 if (URLs.size()!=0){  
 if (URLs.getFirst().getcDepth()>=finalDepth){  
 yesList.addAll(URLs);  
 }  
 else{  
 nopeList.addAll(URLs);  
 for (int countSite=URLs.size(); countSite!=0 && waitThreads!=0;countSite-- , waitThreads--){  
 this.notify();  
 }  
 }  
 }  
 }  
  
 public LinkedList<URLDepthPair> getSite(){  
 return yesList;  
 }  
}

*Результаты: Глубина равна 3.* Слева выводится глубина сайта, дальше гиперссылка на сам сайт.

**

Выводы: реализовал элементарный веб-сканер. Сканер автоматически загружает веб-страницы из интернета и ищет новые ссылки на этих страницах и повторяет их. Расширил сканер для использования поточной обработки Java так, чтобы несколько веб-страниц можно было сканировать параллельно.