**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:

«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.

*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();  
 }  
}

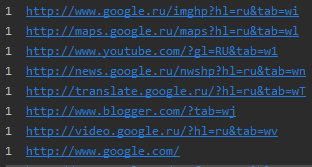
*Crawler:*

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;  
 }  
  
  
}

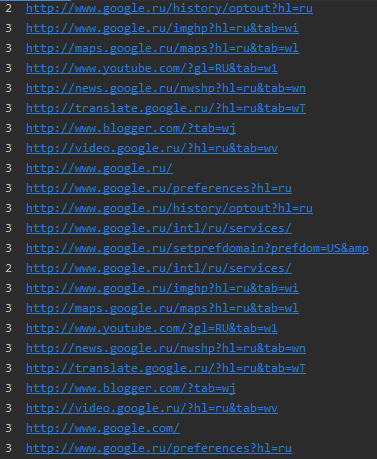
*URLDepthPair:*

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;  
 }  
 }  
}

*Результаты: Глубина равна 0.* Слева выводится глубина сайта, дальше гиперссылка на сам сайт.

**

*Глубина равна 2:*

**

Выводы: реализовал элементарный веб-сканер. Сканер автоматически загружает веб-страницы из интернета и ищет новые ссылки на этих страницах и повторяет их.