**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

Кафедра «МКиИТ»

**Дисциплина «Объектно-ориентированное программирование»**

**Отчет по лабораторной работе №8**

Выполнил: студент группы БВТ1901

Перевозчиков С. В.

Руководитель:

Мосева М. С.

Москва 2020

Цель работы**:** модифицировать сканер с использованием поточной обработки Java так, чтобы несколько веб-страниц можно было сканировать параллельно.

Техническое задание:

1. Реализовать класс с именем URLPool, который будет хранить список всех URL-адресов для поиска, а также относительный "уровень" каждого из этих URL-адресов.
2. Создать класс CrawlerTask, который реализует интерфейс Runnable.
3. Изменить программу так, чтобы она принимала третий параметр через командную строку, который будет определять количество порождаемых потоков веб-сканера.
4. Реализуйте метод wait() внутри «get URL» в случае, если ни один URL-адрес в настоящее время недоступен.

Выполнение лабораторной работы.

1. Код для класса URLPool:

import java.util.\*;

public class URLPool

{

public static int max\_depth;

private LinkedList<URLDepthPair> pendingURLs;

public LinkedList<URLDepthPair> processedURLs;

public ArrayList<String> seenURLs = new ArrayList<String>();

public int waitingThreads;

public URLPool(int d)

{

waitingThreads = 0;

pendingURLs = new LinkedList<URLDepthPair>();

processedURLs = new LinkedList<URLDepthPair>();

max\_depth = d;

}

public synchronized int getWaitThreads()

{

return waitingThreads;

}

public synchronized int size()

{

return pendingURLs.size();

}

public synchronized void decrimentWaitingThreads()

{

waitingThreads--;

}

public synchronized boolean put(URLDepthPair depthPair)

{

boolean isAdded = false;

if (depthPair.getDepth() < max\_depth && !processedURLs.contains(depthPair.getURL()))

{

pendingURLs.addLast(depthPair);

isAdded = true;

this.notify();

}

else

{

addSeenURL(depthPair);

}

return isAdded;

}

public synchronized URLDepthPair get()

{

URLDepthPair depthPair = null;

if (pendingURLs.size() == 0)

{

waitingThreads++;

try

{

this.wait();

}

catch (InterruptedException interruptedException)

{

System.err.println("InterruptedException: " + interruptedException.getMessage());

return null;

}

}

if(waitingThreads>0)

waitingThreads--;

depthPair = pendingURLs.pop();

addSeenURL(depthPair);

processedURLs.add(depthPair);

return depthPair;

}

private synchronized void addSeenURL(URLDepthPair dp)

{

if(!seenURLs.contains(dp.toString()))

seenURLs.add(dp.toString());

}

public synchronized ArrayList<String> getSeenList()

{

return seenURLs;

}

}

1. Код для класса URLDepthPair:

import java.net.\*;

public class URLDepthPair

{

private int depth;

private String url = "";

URLDepthPair(String u, int d)

{

url = u;

depth = d;

}

URLDepthPair(String u)

{

url = u;

depth = 0;

}

public String getURL()

{

return url;

}

public int getDepth()

{

return depth;

}

public String toString()

{

return url + " depth = " + depth;

}

public String getDocPath()

{

try

{

URL tempURL = new URL(url);

return tempURL.getPath();

}

catch (MalformedURLException malformedURLExc)

{

System.err.println("MalformedURLException: " + malformedURLExc.getMessage());

return null;

}

}

public String getWebHost()

{

try

{

URL tempURL = new URL(url);

return tempURL.getHost();

}

catch (MalformedURLException malformedURLExc)

{

System.err.println("MalformedURLException: " + malformedURLExc.getMessage());

return null;

}

}

}

1. Код для класса CrawlerTask:

import java.util.\*;

public class CrawlerTask implements Runnable

{

public URLDepthPair depthPair;

public URLPool myPool;

public CrawlerTask(URLPool pool)

{

myPool = pool;

}

public void run()

{

depthPair = myPool.get();

int myDepth = depthPair.getDepth();

LinkedList<String> linksList = new LinkedList<String>();

linksList = Crawler.getAllLinks(depthPair);

for (String newURL : linksList)

{

URLDepthPair newDepthPair = new URLDepthPair(newURL, myDepth +1);

myPool.put(newDepthPair);

}

}

}

1. Код для класса Crawler:

import java.util.\*;

import java.io.\*;

import java.net.\*;

public class Crawler

{

static int numThreads;

static int timeOut = 1000;

public static void main(String[] args)

{

URLDepthPair currentDepthPair;

ArrayList<String> seenURLs = new ArrayList<String>();

URLPool pool;

try

{

currentDepthPair = new URLDepthPair(args[0]);

seenURLs.add(currentDepthPair.getURL());

pool = new URLPool(Integer.parseInt(args[1]));

pool.put(currentDepthPair);

numThreads = Integer.parseInt(args[2]);

}

catch(Exception exc)

{

System.out.println("usage: java Crawler <URL> <depth> <count of threads>");

return;

}

int activeAmount = Thread.activeCount();

while (pool.getWaitThreads() != numThreads)

{

if (Thread.activeCount() - activeAmount < numThreads)

{

CrawlerTask crawler = new CrawlerTask(pool);

new Thread(crawler).start();

}

else {

try

{

Thread.sleep(100);

}

catch (InterruptedException ie)

{

System.out.println("InterruptedException");

}

}

}

for(String s : pool.seenURLs)

{

System.out.println(s);

}

System.out.println(pool.seenURLs.size());

System.exit(0);

}

public static LinkedList<String> getAllLinks(URLDepthPair myDepthPair)

{

LinkedList<String> URLs = new LinkedList<String>();

Socket socket1;

try

{

socket1 = new Socket(myDepthPair.getWebHost(), 80);

}

catch (UnknownHostException e)

{

System.err.println("UnknownHostException: " + e.getMessage());

return URLs;

}

catch (IOException ex)

{

return URLs;

}

try

{

socket1.setSoTimeout(timeOut);

}

catch (SocketException exc)

{

System.err.println("SocketException: " + exc.getMessage());

return URLs;

}

String docPath = myDepthPair.getDocPath();

String webHost = myDepthPair.getWebHost();

OutputStream outStream;

try

{

outStream = socket1.getOutputStream();

}

catch (IOException e)

{

return URLs;

}

PrintWriter printWriter = new PrintWriter(outStream, true);

printWriter.println("GET " + docPath + " HTTP/1.1");

printWriter.println("Host: " + webHost);

printWriter.println("Connection: close");

printWriter.println();

InputStream inStream;

try

{

inStream = socket1.getInputStream();

}

catch (IOException ioExc)

{

System.err.println("IOException: " + ioExc.getMessage());

return URLs;

}

InputStreamReader inStreamReader = new InputStreamReader(inStream);

BufferedReader BuffReader = new BufferedReader(inStreamReader);

while (true)

{

String line;

try

{

line = BuffReader.readLine();

}

catch (IOException e)

{

return URLs;

}

if (line == null)

break;

int beginIndex = 0;

int endIndex = 0;

int index = 0;

while (true)

{

String START\_URL = "a href=\"";

String END\_URL = "\"";

index = line.indexOf(START\_URL, index);

if (index == -1)

break;

index += START\_URL.length();

beginIndex = index;

endIndex = line.indexOf(END\_URL, index);

index = endIndex;

try

{

String newLink = line.substring(beginIndex, endIndex);

if(URLs.contains(newLink))

continue;

if(newLink.startsWith("http")) {

URLs.add(newLink);

}

else if(!newLink.startsWith("tel"))

{

if(newLink.startsWith("/"))

URLs.add("http://"+webHost+""+newLink);

else

URLs.add("http://"+webHost+"/"+newLink);

}

}

catch(Exception exception)

{

break;

}

}

}

return URLs;

}

}

Результат работы программы:



Вывод:модифицировал веб-сканер с использованием поточной обработки Java так, чтобы несколько веб-страниц можно было сканировать параллельно.