Лабораторная работа № 8

по дисциплине

“Объектно-ориентированное программирование”

Выполнил студент

группы БИБ1901

Бушеленков Роман

Москва 2021

**Цель работы:** расширите сканер для использования поточной обработки Java так, чтобы несколько веб-страниц можно было сканировать параллельно.

**Задачи на лабораторную работу:**

1. Реализовать класс с именем URLPool

2. Создать класс CrawlerTask

**Результат выполнения работы:**

package lab8;  
  
import java*.*util*.*\*;  
import java*.*io*.*\*;  
import java*.*net*.*\*;  
  
public class Crawler {  
 static int numThreads = 10;  
 static int timeOut = 300;  
 public static void main(String[] args) {  
 URLDepthPair currentDepthPair = new URLDepthPair("http://ege.edu.ru/ru/special/main/");  
 ArrayList<String> seenURLs = new ArrayList<String>();  
 seenURLs*.*add(currentDepthPair*.*getURL());  
  
 URLPool pool = new URLPool(2);  
 pool*.*put(currentDepthPair);  
  
 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("Caught unexpected: InterruptedException, ignoring...");  
 }  
  
 }  
 }  
 for(String s : pool*.*seenURLs) {  
 System*.*out*.*println(s + " depth");  
 }  
  
 System*.*out*.*println(pool*.*seenURLs*.*size());  
 System*.*exit(0);  
  
 }  
  
 public static LinkedList<String> getAllLinks(URLDepthPair myDepthPair) {  
  
 LinkedList<String> URLs = new LinkedList<String>();  
 Socket sock;  
  
 try {  
 sock = new Socket(myDepthPair*.*getWebHost(), 80);  
 }  
 catch (UnknownHostException e) {  
 System*.*err*.*println("UnknownHostException: " + e*.*getMessage());  
 return URLs;  
 }  
 catch (IOException ex) {  
 System*.*err*.*println("IOException: " + ex*.*getMessage());  
 return URLs;  
 }  
  
 try {  
 sock*.*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 = sock*.*getOutputStream();  
 }  
 catch (IOException e) {  
 System*.*err*.*println("IOException: " + e*.*getMessage());  
 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 = sock*.*getInputStream();  
 }  
 catch (IOException ioException){  
 System*.*err*.*println("IOException: " + ioException*.*getMessage());  
 return URLs;  
 }  
 InputStreamReader inStreamReader = new InputStreamReader(inStream);  
 BufferedReader BuffReader = new BufferedReader(inStreamReader);  
  
 while (true) {  
 String line;  
 try {  
 line = BuffReader*.*readLine();  
 }  
 catch (IOException e) {  
 System*.*err*.*println("IOException: " + e*.*getMessage());  
 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) {  
 System*.*err*.*println("error while substringing: "+beginIndex + " " + endIndex);  
 break;  
 }  
  
 }  
  
 }  
 return URLs;  
 }  
}

package lab8;  
  
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);  
 }  
 }  
}

package lab8;  
  
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;  
 }  
  
 public String getDocPath() {  
 try {  
 URL tempURL = new URL(url);  
 return tempURL*.*getPath();  
 }  
 catch (MalformedURLException malformedURLException) {  
 System*.*err*.*println("MalformedURLException in getDocPath(): " + malformedURLException*.*getMessage());  
 return null;  
 }  
 }  
  
 public String getWebHost() {  
 try {  
 URL tempURL = new URL(url);  
 return tempURL*.*getHost();  
 }  
 catch (MalformedURLException malformedURLException) {  
 System*.*err*.*println("MalformedURLException in getWebHost: " + malformedURLException*.*getMessage());  
 return null;  
 }  
 }  
}

package lab8;  
  
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;  
 }  
}

