**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

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

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе № 8

по дисциплине «Информационные технологии и программирование»

Выполнил: студент группы БФИ2102

Голубева Елена Александровна

Проверила:

Мосева Марина Сергеевна

Москва, 2022

Цель работы

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

Задание

1. Реализуйте класс с именем URLPool, который будет хранить список всех URL-адресов для поиска, а также относительный "уровень" каждого из этих URL-адресов.

2.Чтобы выполнить веб-сканирование в нескольких потоках, необходимо создать класс CrawlerTask, который реализует интерфейс Runnable. Каждый экземпляр CrawlerTask должна иметь ссылку на один экземпляр класса URLPool.

3. Изменить программу так, чтобы она принимала третий параметр через командную строку, который будет определять количество порождаемых потоков веб-сканера.

4.Синхронизировать объект пула URL-адресов во всех критических точках, так как теперь код должен быть ориентирован на многопоточность.

5. Веб-сканер не должен постоянно опрашивать пул URL-адресов в случае, если он пуст. Вместо этого пусть они ожидают в случае, когда нет доступных URL-адресов.

Код файла Crawlers.java

package com.company;  
  
import java.net.\*;  
import java.util.\*;  
import java.io.\*;  
  
//выводин список найденных адресов  
public class Crawlers {  
  
 public static void main(String[] args) {  
 args = new String[]{"http://go.com", "2", "5"

};  
  
 int depth = 0; //текущая глубина  
 int numThreads = 0; //число потоков  
  
 //проверяет на корректность длины входных данных (адрес глубина и потоки)  
 if (args.length != 3) {  
 System.*out*.println("usage: java Crawler <URL> <depth> <number of crawler threads>");  
 System.*exit*(1);  
 }  
  
 else {  
 try {  
 //значение глубиныи и потока из строки в целое значение  
 depth = Integer.*parseInt*(args[1]);  
 numThreads = Integer.*parseInt*(args[2]);  
 }  
 catch (NumberFormatException nfe) {  
 System.*out*.println("usage: java Crawler <URL> <depth> <number of crawler threads>");  
 System.*exit*(1);  
 }  
 }  
  
 // Пара URL-адресов с глубиной для представления веб-сайта, который пользователь ввел с глубиной 0  
 URLDepthPair currentDepthPair = new URLDepthPair(args[0], 0);  
  
 // пул и добавление в него введенного пользователем адреса  
 URLPool pool = new URLPool();  
 pool.put(currentDepthPair);  
  
  
 // общее число потоков и начальное их число  
 int totalThreads = 0;  
 int initialActive = Thread.*activeCount*();  
  
 // ожидающие потоки не равны запрошенному количеству потоков, если общее количество потоков меньше запрошенного количества  
 // потоков, создайте больше потоков  
 while (pool.getWaitThreads() != numThreads) {  
 if (Thread.*activeCount*() - initialActive < 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...");  
 }  
  
 }  
 }  
  
 // вывод обработанных ссылок с глубиной при всех потоках в процессе ожидания  
 Iterator<URLDepthPair> iter = pool.processedURLs.iterator();  
 while (iter.hasNext()) {  
 System.*out*.println(iter.next());  
 }  
 System.*exit*(0);  
  
  
  
 }  
  
 //метод который принимает пару адрес и глубина и возвращает список  
 public static LinkedList<String> getAllLinks(URLDepthPair myDepthPair) {  
  
 // инициализация списка  
 LinkedList<String> URLs = new LinkedList<String>();  
  
 // инициализация сокета  
 Socket sock;  
  
 // создание нового сокета с адресом, парой и портом 80  
 try {  
 sock = new Socket(myDepthPair.getWebHost(), 80);  
 }  
 catch (UnknownHostException e) {  
 System.*err*.println("UnknownHostException: " + e.getMessage());  
 return URLs;  
 }  
 // возвращает пустой список  
 catch (IOException ex) {  
 System.*err*.println("IOException: " + ex.getMessage());  
 return URLs;  
 }  
  
 // Установка времени ожидания сокета  
 try {  
 sock.setSoTimeout(3000);  
 }  
  
 catch (SocketException exc) {  
 System.*err*.println("SocketException: " + exc.getMessage());  
 return URLs;  
 }  
  
 // строки для пути адреса из пары и для хоста  
 String docPath = myDepthPair.getDocPath();  
 String webHost = myDepthPair.getWebHost();  
  
 // Инициализация OutputStream позволяет сокету отправлять данные на другую стороны соединения  
 OutputStream outStream;  
  
 try {  
 outStream = sock.getOutputStream();  
 }  
 catch (IOException exce) {  
 System.*err*.println("IOException: " + exce.getMessage());  
 return URLs;  
 }  
  
 // инициализация PrintWriter, сброс после каждого вывода  
 PrintWriter myWriter = new PrintWriter(outStream, true);  
  
 // Отправка запроса на сервер  
 myWriter.println("GET " + docPath + " HTTP/1.1");  
 myWriter.println("Host: " + webHost);  
 myWriter.println("Connection: close");  
 myWriter.println();  
  
 // Инициализация InputStream, позволяет получать данные с другой стороны  
 InputStream inStream;  
  
  
 try {  
 inStream = sock.getInputStream();  
 }  
  
 catch (IOException excep){  
 System.*err*.println("IOException: " + excep.getMessage());  
 return URLs;  
 }  
 // Создание новых InputStreamReader и BufferedReader для чтения строк с сервера  
 InputStreamReader inStreamReader = new InputStreamReader(inStream);  
 BufferedReader BuffReader = new BufferedReader(inStreamReader);  
  
  
  
 //чтение строк  
 while (true) {  
 String line;  
 try {  
 line = BuffReader.readLine();  
 }  
  
 catch (IOException except) {  
 System.*err*.println("IOException: " + except.getMessage());  
 return URLs;  
 }  
 // строки закончились  
 if (line == null)  
 break;  
  
  
 // переменные начального, конечного и текущего индекса ссылки  
 int beginIndex = 0;  
 int endIndex = 0;  
 int index = 0;  
  
 while (true) {  
  
 //константа для строки указывающей на ссылку  
 String URL\_INDICATOR = "href=\"";  
  
 //строка указывающая конец хоста  
 String END\_URL = "\"";  
  
  
 // индекс начала ссылки  
 index = line.indexOf(URL\_INDICATOR, index);  
 if (index == -1)  
 break;  
  
 // изменение текущего индекса и задание начального индекса  
 index += URL\_INDICATOR.length();  
 beginIndex = index;  
  
 // нахождение конца хоста(веб-узла) и присвоение текущему индексу значение конечного  
 endIndex = line.indexOf(END\_URL, index);  
 index = endIndex;  
  
 // установка ссылки меду начальным и конечным индексом и добавление адреса в список  
 String newLink = line.substring(beginIndex, endIndex);  
 URLs.add(newLink);  
 }  
  
 }  
 // возвращение списка  
 return URLs;  
 }  
  
}

Код класса URLPool

package com.company;  
  
import java.util.\*;  
  
  
public class URLPool { //класс для хранения адресов для поиска с глубиной  
  
 //список ожидающих адресов  
 private LinkedList<URLDepthPair> pendingURLs;  
  
 // список обработанных адресов  
 public LinkedList<URLDepthPair> processedURLs;  
  
 // массив просмотренных адресов  
 private ArrayList<String> seenURLs = new ArrayList<String>();  
  
 // количество ожидающих потоков  
 public int waitingThreads;  
  
 // инициализация ожидающих потоков и списков с ожидающими и обработанными адресами  
 public URLPool() {  
 waitingThreads = 0;  
 pendingURLs = new LinkedList<URLDepthPair>();  
 processedURLs = new LinkedList<URLDepthPair>();  
 }  
  
 // получение ожидающих потоков  
 public synchronized int getWaitThreads() {  
 return waitingThreads;  
 }  
  
 // получение размера списка с ожидающими адресами  
 public synchronized int size() {  
 return pendingURLs.size();  
 }  
  
 // добавляет пару пару глубина - адрес  
 public synchronized boolean put(URLDepthPair depthPair) {  
  
 // проверка добавления пары  
 boolean added = false;  
  
 // добавление пары в список, если еще не достигнута максимальная глубина  
 if (depthPair.getDepth() < depthPair.getDepth()) {  
 pendingURLs.addLast(depthPair);  
 added = true;  
  
 // уменьшение ожидающих потоков  
 waitingThreads--;  
 this.notify();  
 }  
 // добавление пары в массив просмотренных  
 else {  
 seenURLs.add(depthPair.getURL());  
 }  
  
 return added;  
 }  
  
 // получение следующей пары пары  
 public synchronized URLDepthPair get() {  
  
 // установка значения для пары  
 URLDepthPair myDepthPair = null;  
  
 // ожидание при пустом списке ожидающих адресов, увеличение ожидающих потоков  
 if (pendingURLs.size() == 0) {  
 waitingThreads++;  
 try {  
 this.wait();  
 }  
 catch (InterruptedException e) {  
 System.*err*.println("MalformedURLException: " + e.getMessage());  
 return null;  
 }  
 }  
 // добавление пары в списки просмотренных и обработанных адресов  
 myDepthPair = pendingURLs.removeFirst();  
 seenURLs.add(myDepthPair.getURL());  
 processedURLs.add(myDepthPair);  
 return myDepthPair;  
 }  
 //получение массива с просмотренными адресами  
 public synchronized ArrayList<String> getSeenList() {  
 return seenURLs;  
 }  
}

Код класса CrawlerTask

package com.company;  
  
import java.util.\*;  
  
// реализует интерфейс Runnable  
public class CrawlerTask implements Runnable {  
  
 // поле для пары адрес-глубина  
 public URLDepthPair depthPair;  
  
 //поле для пула  
 public URLPool myPool;  
  
 //конструктор для получения пула  
 public CrawlerTask(URLPool pool) {  
 myPool = pool;  
 }  
  
 //метод для запуска задач класса  
 public void run() {  
  
 // получение пары из пула  
 depthPair = myPool.get();  
  
 // глубина пары  
 int myDepth = depthPair.getDepth();  
  
 // получение всех ссылок с сайта и внесение их в список  
 LinkedList<String> linksList = new LinkedList<String>();  
 linksList = Crawlers.*getAllLinks*(depthPair);  
  
 for (int i=0;i<linksList.size();i++) {  
 String newURL = linksList.get(i);  
  
 // формирование пар для всех найденных ссылок и добавление их в пул  
 URLDepthPair newDepthPair = new URLDepthPair(newURL, myDepth + 1);  
 myPool.put(newDepthPair);  
 }  
 }  
}

Вывод

В процессе выполнения данной лабораторной работы мы расширили сканер для использования поточной обработки Java так, чтобы несколько веб-страниц можно было сканировать параллельно.

Ссылка на Github: https://github.com/ElenaGolubeva/Laboratory-8-Java

Список использованной литературы

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа,

2006.

1. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.