**Федеральное государственное общеобразовательное учреждение высшего образования**

**Московский государственный технический университет**

**им. Н. Э. Баумана**

Факультет "Информатика и системы управления"

Кафедра "Информационные системы и телекоммуникации"

**Отчет по лабораторной работе №8**

**"Улучшенный поисковый робот"**

Студент: Задубин А.А.

Группа: ИУ3-42Б

Преподаватель: Руденкова Ю.С.

**Задание:**

Нам нужно было добавить новые возможности к коду написанному в прошлом задании:

1. Сделали класс с именем URLPool, который сохраняет список всех найденных URLs, вместе с "уровнем" каждого из этих URL. Первый URL с которого мы начали поиск имеет глубину поиска 0. Все URL найденные на этой странице имеют глубину 1, и так далее. Сохранили URL и их глубину вместе как экземпляр класса URLDepthPair. Для хранения элементов использовали LinkedList, так как он очень эффективно выполняет требуемые операции.

Нужно было чтобы пользователь класса URLPool мог извлекать пару URL-глубина из пула, и одновременно удалять ее из списка. Так же нужно было обеспечить возможность добавления пары URL-глубина к пулу. Обе эти операции рассчитаны на вызов из нескольких потоков, так как несколько потоков могут одновременно работать с URLPool.

2. Для того чтобы поиск можно было выполнять в нескольких потоках, создали класс CrawlerTask, который реализует интерфейс Runnable. Каждый экземпляр CrawlerTask имеет ссылку на единственный экземпляр класса URLPool. Поток поискового робота работает так:

1. Извлекает пару URL-глубина из пула (ждет если в данный момент ссылок нет).

2. Загружает веб страницу на которую ссылается URL.

3. Ищет на странице другие URL. Для каждого найденного на странице поток робота должен добавить новую пару URL-глубина в пул ссылок.

Новая пара имеет глубину на единицу большую, чем URL из которого эта ссылка извлечена на шаге 1.

4. Возвращается к шагу 1.

Эти операции продолжаются до тех пор, пока в пуле ссылок не останется ни одной пары (URL, глубина).

3. Так как робот запускает некоторое количество потоков, добавили в программу третий параметр командной строки, задающий число потоков робота. Наша функция main работает так:

1. Обрабатывает аргументы командной строки. Сообщает пользователю об ошибках, если они есть.

2. Создает экземпляр класса пула URL, и помещает указанный пользователем URL с глубиной 0 в пул.

3. Создает указанное пользователем число задач робота (и потоков для их запуска). Каждая задача робота должна иметь ссылку на пул URL, который только что создан.

4. Ожидает пока робот не закончит работу

5. Печатает в результате список найденных URL.

4. Синхронизируем доступ к объекту пула URL во всех важных местах, потому что доступ к этому объекту из потоков должен быть безопасным.

**Код программы:**

Crawler.java

**import** **java.io.\***;

**import** **java.net.MalformedURLException**;

**import** **java.net.Socket**;

**import** **java.net.UnknownHostException**;

**import** **java.util.Iterator**;

**import** **java.util.LinkedList**;

**import** **java.util.regex.Matcher**;

**import** **java.util.regex.Pattern**;

**public** **class** **Crawler** {

**private** **int** threads;

**private** URLPool p;

**public** **Crawler**(String url, **int** depth, **int** threads) {

**this**.threads = threads;

p = **new** URLPool(depth);

p.put(**new** URLDepthPair(url, **0**));

}

**private** **void** **runTask**() {

CrawlerTask task = **new** CrawlerTask(p);

**for** (**int** i = **0**; i < threads; i++) {

CrawlerTask crawler = **new** CrawlerTask(p);

Thread thread = **new** Thread(crawler);

thread.start();

}

**while** (p.getCounter() != threads) {

**try** {

Thread.sleep(**500**);

} **catch** (InterruptedException e) {

System.out.println("Игнорирование непредвиденного прерывания - " + e.getMessage());

}

}

p.printPair();

}

**public** **static** **void** **main**(String[] args) **throws** Exception, IOException {

**if** (args.length != **3**) {

**throw** **new** **Exception**("Correct format: java Crawler <URL> <depth> <threads>");

}

**int** maxSize = **0**;

**int** numOfThreads = **0**;

**try** {

maxSize = Integer.parseInt(args[**1**]);

numOfThreads = Integer.parseInt(args[**2**]);

} **catch** (NumberFormatException nfe) {

System.err.println("Sec argument must be an integer.");

System.exit(**1**);

}

**if** (!args[**0**].substring(**0**, **7**).equals("http://")) {

**throw** **new** **Exception**("First arg must be a link");

}

Crawler crawler = **new** Crawler(args[**0**], maxSize, numOfThreads);

crawler.runTask();

System.exit(**1**);

}

}

CrawlerTask.java

**import** **java.io.BufferedReader**;

**import** **java.io.IOException**;

**import** **java.io.InputStreamReader**;

**import** **java.io.PrintWriter**;

**import** **java.net.MalformedURLException**;

**import** **java.net.Socket**;

**import** **java.net.UnknownHostException**;

**import** **java.util.regex.Matcher**;

**import** **java.util.regex.Pattern**;

**public** **class** **CrawlerTask** **implements** Runnable {

**private** URLPool p;

**public** **CrawlerTask**(URLPool p) {

**this**.p = p;

}

**public** **static** String **patternSearch**(String line) {

Pattern patt = Pattern.compile("href=\"http[^\"]+\"");

Matcher match = patt.matcher(line);

**if** (match.find()) {

**return** (line.substring(match.start() + **6**, match.end() - **1**));

} **else**

**return** **null**;

}

**public** **void** **getContent**(URLDepthPair obj) **throws** Exception {

**try** (Socket mySocket = **new** Socket(obj.hostNameString(), **80**)) {

mySocket.setSoTimeout(**5000**);

**try** (PrintWriter out = **new** PrintWriter(mySocket.getOutputStream(), **true**)) {

out.println("GET " + obj.pathNameString() + " HTTP/1.1");

out.println("Host: " + obj.hostNameString());

out.println("Connection: close");

out.println();

**try** (BufferedReader in = **new** BufferedReader(**new** InputStreamReader(mySocket.getInputStream()))) {

String line;

**while** ((line = in.readLine()) != **null**) {

**int** newDepth = obj.getDepth() + **1**;

String findedUrl = patternSearch(line);

**if** (findedUrl != **null**) {

p.put(**new** URLDepthPair(findedUrl, newDepth));

}

}

}

}

}

}

**@Override**

**public** **void** **run**() {

**while** (**true**) {

URLDepthPair pair = p.get();

**try** {

getContent(pair);

} **catch** (Exception e) {

System.err.println(e);

}

}

}

}

URLDepthPair.java

**import** **java.net.MalformedURLException**;

**import** **java.net.URL**;

**public** **class** **URLDepthPair** {

**private** String url;

**private** **int** depth;

**public** **URLDepthPair**(String url, **int** depth) {

**this**.url = url;

**this**.depth = depth;

}

**@Override**

**public** String **toString**() {

**return** (url + " " + depth);

}

**public** String **hostNameString**() **throws** MalformedURLException {

**try** {

**return** **new** **URL**(url).getHost();

} **catch** (Exception e) {

**return** **null**;

}

}

**public** String **pathNameString**() **throws** MalformedURLException {

**try** {

**return** **new** **URL**(url).getPath();

} **catch** (Exception e) {

**return** **null**;

}

}

**public** **int** **getDepth**() {

**return** depth;

}

**public** String **getURL**() {

**return** url;

}

}

URLPool.java

**import** **java.util.LinkedList**;

**public** **class** **URLPool** {

**public** **int** maxSize;

**private** **int** counter;

**private** LinkedList<URLDepthPair> waitingList;

**private** LinkedList<URLDepthPair> visitedList;

**public** **URLPool**(**int** maxSize) {

**this**.counter = **0**;

**this**.maxSize = maxSize;

**this**.waitingList = **new** LinkedList<URLDepthPair>();

**this**.visitedList = **new** LinkedList<URLDepthPair>();

}

**public** **synchronized** **int** **getCounter**() {

**return** counter;

}

**public** **synchronized** **void** **printPair**() {

**for** (URLDepthPair i : visitedList) {

System.out.println(i.getURL() + " " + i.getDepth());

}

}

**public** **synchronized** **boolean** **isContainsList**(LinkedList<URLDepthPair> obj, String str) {

**for** (URLDepthPair i : obj) {

**if** (i.getURL().equals(str)) {

**return** **true**;

}

}

**return** **false**;

}

**public** **void** **put**(URLDepthPair obj) {

**synchronized** (waitingList) {

**boolean** isContains = **false**;

isContains = isContainsList(waitingList, obj.getURL());

**if** (!isContains) {

isContains = isContainsList(visitedList, obj.getURL());

}

**if** (!isContains) {

**if** (obj.getDepth() < maxSize) {

waitingList.addLast(obj);

waitingList.notify();

}

visitedList.addLast(obj);

}

}

}

**public** URLDepthPair **get**() {

URLDepthPair item = **null**;

**synchronized** (**this**) {

**while** (waitingList.size() == **0**) {

counter++;

**try** {

wait();

} **catch** (Exception e) {

System.err.println(e);

}

counter--;

}

item = waitingList.removeFirst();

}

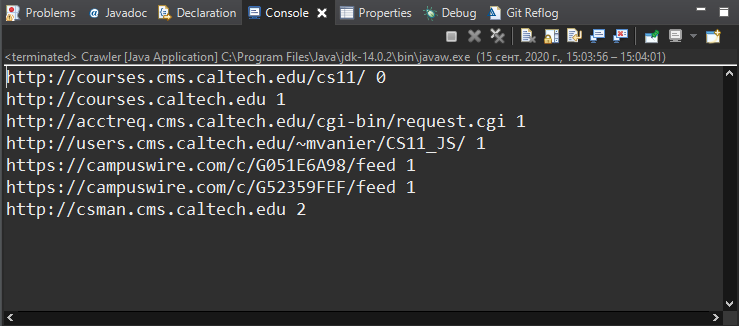
**return** item;

}

}

**Вывод в консоль:**

***Аргументы командной строки:***http://courses.cms.caltech.edu/cs11/ 10 2

****

**Вывод:** в ходе выполнения лабораторной работы мы улучшили поискового робота. Теперь он может использовать несколько потоков для поиска ссылок на сайте.