**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 №7**

**"Поисковый робот"**

Студент: Задубин А.А.

Группа: ИУ3-42Б

Преподаватель: Руденкова Ю.С.

**Задание:**

В этой лабораторной работе, требовалось написать простейший поисковый робот. Робот должен был автоматически загружать веб страницы из сети Интернет, искать в них новые ссылки, и повторять эту операцию для каждой найденной ссылки.

Он просто просматривает новые URL (указывающее на расположение других веб страниц) на каждой странице, сохраняет эти ссылки и печатает их в конце работы программы. Программа должна была иметь два аргумента в командной строке:

1. Строка, содержащая URL страницы с которой начинается поиск

2. Положительное целое число – максимальная глубина поиска

Если программа получает некорректные аргументы, она немедленно прекращает работу и печатает сообщение, содержащее правила вызова, например:

Правила вызова: java Crawler <URL> <глубина поиска>

Программа должна была сохранять URL в виде строкиString вместе с глубиной (которая, в начале, равна 0). Для этого мы должны был создать класс, содержащий пару значений [URL, глубина].

Программа подключается к 80 порту заданного в URL сайта, используя класс Socket и запрашивает указанную веб страницу.

Программа должна была разбирать полученный в ответ текст (если он получен), строка за строкой отыскивая фрагменты строк такого формата:

<a href="[какой либо URL начинающийся с http://]">

Найденные URL сохраняются вместе с новым значением глубины ссылки в списке LinkedList объектов (URL, глубина). Новое значение глубины должно было быть на единицу больше глубины URL обрабатываемой страницы. Затем программа закрывает соединение с хостом.

Перед завершением программа выводит все найденные URL в пределах заданной глубины поиска.

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** {

**public** **static** **int** lim = **0**;

**public** **static** LinkedList<URLDepthPair> waitingList = **new** LinkedList<URLDepthPair>();

**public** **static** LinkedList<URLDepthPair> visitedList = **new** LinkedList<URLDepthPair>();

**public** **static** **void** **urlSearch**(URLDepthPair linkAndDepth)

**throws** UnknownHostException, MalformedURLException, IOException {

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

mySocket.setSoTimeout(**5000**);

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

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

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

out.println("Connection: close");

out.println();

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

String line;

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

**int** nextDepth = linkAndDepth.getDepth() + **1**;

String pattRez = patternSearch(line);

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

**if** (pattRez != **null** && nextDepth < lim) {

isContains = isContainsList(waitingList, pattRez);

**if** (!isContains) {

isContains = isContainsList(visitedList, pattRez);

}

**if** (!isContains) {

waitingList.add(**new** URLDepthPair(pattRez, nextDepth));

}

}

}

}

}

}

}

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

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

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

**return** **true**;

}

}

**return** **false**;

}

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

// href=\"(http[^\"]+)\"

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

Matcher match = patt.matcher(line);

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

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

} **else**

**return** **null**;

}

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

**if** (args.length != **2**) {

**throw** **new** **Exception**("Правила ввода: java Crawler <URL> <глубина поиска>");

}

// int lim = 0;

**try** {

// Parse the string argument into an integer value.

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

} **catch** (NumberFormatException nfe) {

// The first argument isn't a valid integer. Print

// an error message, then exit with an error code.

System.out.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");

}

// LinkedList<URLDepthPair> waitingList = new LinkedList<URLDepthPair>();

// LinkedList<URLDepthPair> visitedList = new LinkedList<URLDepthPair>();

waitingList.add(**new** URLDepthPair(args[**0**], **0**));

**while** (!waitingList.isEmpty()) {

URLDepthPair nextPair = waitingList.poll();

urlSearch(nextPair);

visitedList.add(nextPair);

}

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

System.out.println(i);

}

}

}

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 **bestContains**() {

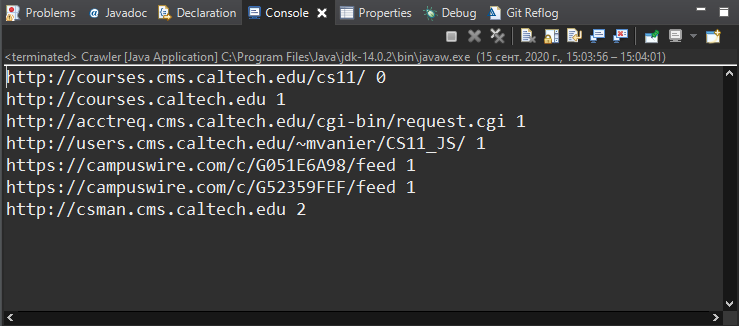
**return** (**this**.url);

}

}

**Вывод в консоль:**

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

****

**Вывод:** был написан простейший поисковый робот, который ищет ссылки до заданной глубины поиска и выводит найденные ссылки на экран.