ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра МКиИТ

Лабораторная работа по технологиям программирования №7

«Многоядерный генератор фракталов»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Абросимов Данил

Вариант №1

Москва, 2018

Оглавление

[1. Цель и задачи. 3](#_Toc507964498)

[1.1. Доработать класс location, реализовав проверку точек на идентичность и реализовав метод кучи. 3](#_Toc507964499)

[1.2. Доработать класс AStarState, реализовав все нужные для работы программы функции.. 3](#_Toc507964500)

[2. Анализ предметной области и выбор инструментария. 3](#_Toc507964501)

[3. Код программы и объяснение функций. 3](#_Toc507964502)

[3.1. Класс Location. 3](#_Toc507964503)

[3.2. Класс AppStarState. 4](#_Toc507964504)

[4. Заключение. 6](#_Toc507964505)

1. Цель и задачи.

Написать программу, которая будет искать ссылке на заданной странице, до определенной глубины, и выводить их список.

1. Анализ предметной области и выбор инструментария.

Для написания данной программы использоваться IDE IntelliJ Idea, так как в данной и последующих задачах требуются средства для отладки программы.

Код программы и объяснение функций.

* 1. Класс WebCrawler.

import java.net.\*;

import java.util.\*;

import java.io.\*;

public class WebCrawler {

public static void main(String[] args) {

int depth = 0;

String URL = "", sDepth = "";

//Проверка введенных данных

if (args.length != 2) {

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

System.exit(1);

} else {

try {

depth = Integer.parseInt(sDepth);

} catch (NumberFormatException nfe) {

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

System.exit(1);

}

}

LinkedList<URL\_DP> checkURLs = new LinkedList<URL\_DP>();

LinkedList<URL\_DP> checkedURLs = new LinkedList<URL\_DP>();

LinkedList<URL\_DP> checkingURLs = new LinkedList<URL\_DP>();

URL\_DP usedDP = new URL\_DP(args[0], 0);

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

while(checkURLs.size() != 0) {

URL\_DP depthP = checkURLs.pop();

checkedURLs.add(depthP);

int myDepth = depthP.getDepth();

links = WebCrawler.getLinks(usedDP);

// TODO Проверить: если заявленная глубина не достигнута,

// TODO добавить сайт, который не был проверен

}

//Пройти через все собранные URL и вывести каждый

Iterator<URL\_DP> iterator = checkedURLs.iterator();

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

}

public static LinkedList<String> getLinks(URL\_DP \_URL){

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

String endURL = "\"";

String startURL = "a href=\"";

//Создание соккета

Socket socket;

try {

socket = new Socket(\_URL.getHost(), 80);

}

catch(IOException ex){return null;}

try {

InputStreamReader in = new InputStreamReader(socket.getInputStream());

}

catch(IOException ex){ return null;}

OutputStream ostr;

try{

ostr = socket.getOutputStream();

}

catch(IOException ex){return null; }

InputStream inStr;

try{

inStr = socket.getInputStream();

}

catch(IOException ex){return null;}

InputStreamReader readInput = new InputStreamReader(inStr);

BufferedReader readBuffer = new BufferedReader(readInput);

boolean tryRead = true;

while(tryRead){

String text = "";

if (text == null) { tryRead = false; };

int index = 0;

while(index != -1){

//Найти начало нужной строки

index = text.indexOf(startURL, index);

// TODO индексы начала и конца строки

URLs.add(text.substring(0,0));

}

}

String Path = \_URL.getPath();

String Host = \_URL.getHost();

return URLs ;

}

}

main() проверяет корректность введенных данных, если данные не верны, выводит сообщение об ошибке. Дает запросы на нахождение ссылок до заданной глубины. Под конец итераций мы получаем готовый список пройденных URL в checkedURLs. Далее, выводим каждый член списка с помощью итератора.

getLinks(URL\_DP \_URL) метод, который возвращает String LinkedList найденных ссылок на данной странице. Подключается к сайту через URL\_DP.  
Создаем сокет с хостом, который получили через аргументы и портом 80. Ставим таймаут на сокет. С помощью writer посылаем запрос серверу. Инициализируем InputStreamReader и пытаемся считать информацию с сайта. Если находим слово, которое начинается на "a href=\"" и, с помощью индексации начала и конца вложенной ссылки, выносим полученную ссылку в список.

* 1. Класс URL\_DP.

import java.net.\*;

public class URL\_DP {

private int \_Depth;

private String \_URL;

public URL\_DP(String URL, int depth) {

\_Depth = depth;

\_URL = URL;

}

public String getURL() {

return \_URL;

}

public int getDepth() {

return \_Depth;

}

public String ToString() {

String stringDepth = Integer.toString(\_Depth);

return stringDepth + '\t' + \_URL;

}

public String getPath() {

try {

URL url = new URL(\_URL);

return url.getPath();

}

catch (MalformedURLException e) {

return null;

}

}

public String getHost() {

try {

URL url = new URL(\_URL);

return url.getHost();

}

catch (MalformedURLException e) {

return null;

}

}

}

getURL() возвращает текстовый URL.  
getDepth() возвращает глубину.  
getPath() Возвращает текстовый пусть.

getHost() возвращает хост.

ToString() Вывод URL и глубины в одной строке.

1. Заключение.

Проделав всю работу, мы познакомились с java GUI, научились создавать кнопки, выпадающие списки, вставлять картинки, обрабатывать запросы и многое другое. Научились создавать фракталы.