/\*--------------------------------------------------------

1. Name / Date: Jessica Bender / Version 2: Febuary 7th 2021 5:44pm

2. Java version used, if not the official version for the class:

My Java:

java version "9.0.4"

Java(TM) SE Runtime Environment (build 9.0.4+11)

Java HotSpot(TM) 64-Bit Server VM (build 9.0.4+11, mixed mode)

3. Precise command-line compilation examples / instructions:

> javac MiniWebserver.java

4. Precise examples / instructions to run this program:

>java MiniWebserver

enter name and 2 numbers into WebAdd.html hit submit

go to http://localhost:2540/abc.

5. List of files needed for running the program.

a. MiniWebserver.java

b. WebAdd.html

6. Notes:

----------------------------------------------------------\*/

/\*Explain how MIME-types are used to tell the browser what data is coming.

In order to know how MIME-types are used you must first know what MIME-types are. MIME stands for Multipurpose Internet Mail Extensions.

it is what detimins the format and type of a document. This includs things like .doc for Microsoft Word documents. .png or .jpeg for images. And much more.

by knowing the MIME-type websevers can determin how to display the incoming information. If a MIME-type cannot be detemined the sever may not display the content as desired

Along with checking the MIME-type the sever also looks for idetifation of the client and if it has access rights. This allows the URL to trnslate a local file.

\*/

/\*Explain how you would return the contents of requested files (web pages) of type HTML (text/html)

I would read the information in the html tags to deptemin how to display the content. The tags usually tell me for example what colors to display and what size to display the content at.

I would then read the actually content following the tag and desplay that as instuted.

I would continue this untill I got to the end of the html document.

\*/

/\*Explain how you would return the contents of requested files (web pages) of type TEXT (text/plain)

since this is a plain text I would read the contects and display them as is.

There are no specification on how to deplay the text. Thus all the text would be the same faunt size and color.

\*/

import java.io.\*; //imports all directories located in java.io

import java.net.\*; //imports all directories located in java.net

class ListenWorker extends Thread { //start of ListenWorker class

Socket sock; //new Socket called sock

ListenWorker (Socket s) {sock = s;}

public void run(){ //start of run

PrintStream out = null; // initialize a PrintStream called out and sets its value to null

BufferedReader in = null; //initialize a BufferedReader called in and sets its value to null

try {//Start try statement. If try doesnt work moved down to catch statement

out = new PrintStream(sock.getOutputStream()); //makes a new PrintStream and assigns it to out and gets the OutputStream from the Socket sock

in = new BufferedReader //makes a new BufferedReader and assigns it to in

(new InputStreamReader(sock.getInputStream())); //gets the InputStream from the Socket sock and makes a new InputStreamReader and puts in inside the BufferedReader in

System.out.println("Sending the HTML Reponse now: " +

Integer.toString(MiniWebserver.i) + "\n" );

String HTMLResponse = "<html> <h1> Hello Browser World! " +

Integer.toString(MiniWebserver.i++) + "</h1> <p><p> <hr> <p>";

out.println("HTTP/1.1 200 OK");

out.println("Connection: close");

out.println("Content-Length: 400");

out.println("Content-Type: text/html \r\n\r\n");

out.println(HTMLResponse); //prints HTMLResponse

String url = in.readLine(); //reads in and sets it to string variable url // local definition of socketData of type String

String equ = "="; // new string called equ and is set to =

String and = "&"; // new string called and and is set to &

url = url.replaceAll(and, equ); //replaces all & with = in url

String HTTP = " HTTP"; //new string called HTTP and is set to HTTP

url = url.replaceAll(HTTP, equ); //replaces all HTTP with = in url

String split[] = url.split("="); //splits url by the =

String name = split[1]; // gets the name from the split at index 1

String num1 = split[3]; // gets the first from the split at index 3

String num2 = split[5];// gets the second number from the split at index 5

int sum = Integer.parseInt(num1) + Integer.parseInt(num2);// add num1 and num2

//int x =Integer.parseInt(num1);

//int y =Integer.parseInt(num2);

//int summ = x + y;

//out.println(summ);

String hello = "<html> <h2> Hello " + name + " the sum of "+ num1 +" and " + num2 + " is " + sum +

"</h2> <p><p> <hr> <p>"; //prints the user

out.println(hello);

//out.println(hello);

for(int j=0; j<6; j++){

out.println(in.readLine() + "<br>\n");

} //closes for statement

out.println("</html>"); //prints message to PrintStream out

sock.close(); // close the connection to the Socket sock, but does not colose the connecton to the server;

} //closes try statement

catch (IOException x) { //start catch. Catches an IOException if the try fails.

System.out.println("Error: Connetion reset. Listening again...");//prints message if try fails

}//closes catch statement

}//closes run statement

}//closes ListenWorker class

public class MiniWebserver { //start of MiniWebserver class. This is a public class.

static int i = 0; //new int called i and sets it to 0

public static void main(String a[]) throws IOException { //start of main void method

int q\_len = 6; //sets q\_len to 6

int port = 2540; //sets port to 2540

Socket sock; //new Socket called sock

ServerSocket servsock = new ServerSocket(port, q\_len); //new ServerSocket called servsock. also sets it equal to the port and q\_len that we se above

System.out.println("Jess Bender's MiniWebserver running at " + port + ".");//Prints statement. Changed code to have my name and changed the statement so that the variable port would be printed and it was not hardcoded in the string statement.

System.out.println("Point Firefox browser to http://localhost:2540/abc.\n");//prints message

while (true) { //starts wile true statement to look for the next client connection

sock = servsock.accept();

new ListenWorker (sock).start();// starts new ListenWorker

}//clsoes while true

}//closes main void statement

}//closes miniwebserver class.