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

1. Name: Kaijun He / Date: 01/10/2018

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

java version "1.8.0\_144"

3. Precise command-line compilation examples / instructions:

> javac InetServer.java

> Javac InetClient.java

4. Precise examples / instructions to run this program:

In separate shell windows:

> java InetServer

> java InetClient

5. List of files needed for running the program.

a. InetServer.java

b. InetClient.java

5. Notes:

because this code for InetServer and Inetclient is from professor's sample code, so it must be the same. and i add

up my own comments to those two files for grading. i test the client and server file to using my local internet ip address

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

import java.io.\*; // import library for input and output

import java.net.\*; // import librart for network

class Worker extends Thread { // define a worker class with a constructor with member socket

// where the constructor worker will read an input arguments s and assign it to local type socket sock

Socket sock;

Worker (Socket s) {

sock = s;

}

public void run(){

// In run function, intial the out and input stream and sign it to null;

PrintStream out = null;

BufferedReader in = null;

try {

//assign in and out value by calling sock with getInputStream() and getOutputStream()

in = new BufferedReader(new InputStreamReader(sock.getInputStream()));

out = new PrintStream(sock.getOutputStream());

try {

String name;

name = in.readLine (); // read the host name or ip address typing from the terminal window client

System.out.println("Looking up the name from input " + name);

printRemoteAddress(name, out);

} catch (IOException x) { // exception to throw message when could not read server correctly

System.out.println("Server read error");

x.printStackTrace ();

}

sock.close(); // close this sock connection

} catch (IOException ioe) {System.out.println(ioe);}

}

// printREmoteAddress is used to print out what the server find out and send to client window

static void printRemoteAddress (String name, PrintStream out) {

try {

out.println("Looking up " + name + "...");

InetAddress machine = InetAddress.getByName (name);

out.println("Host name : " + machine.getHostName ());

out.println("Host IP : " + toText (machine.getAddress ()));

} catch(UnknownHostException ex) {

out.println ("Failed in atempt to look up " + name);

}

}

// this part is not useful for simple server and client running process

static String toText (byte ip[]) {

StringBuffer result = new StringBuffer ();

for (int i = 0; i < ip.length; ++ i) {

if (i > 0)

result.append (".");

result.append (0xff & ip[i]);

}

return result.toString ();

}

}

// main function for InetServer java file, in this case we just make port number same as client java file.

// then it will be able to commnicate between server and client

public class InetServer {

public static void main(String a[]) throws IOException {

int q\_len = 6;

int port = 2000; // port number define by user

Socket sock;

ServerSocket servsock = new ServerSocket(port, q\_len);

System.out.printf("sample Inet server 1.8 starting up, listening at port %d.\n", port);

while (true) {

sock = servsock.accept(); // wait for client to do next lookup

new Worker(sock).start(); // and make a new work to handle the work.

}

}

}

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

1. Name: Kaijun He / Date: 01/10/2018

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

java version "1.8.0\_144"

3. Precise command-line compilation examples / instructions:

> javac InetServer.java

> Javac InetClient.java

4. Precise examples / instructions to run this program:

In separate shell windows:

> java InetServer

> java InetClient

5. List of files needed for running the program.

a. InetServer.java

b. InetClient.java

5. Notes:

because this code for InetServer and Inetclient is from professor's sample code, so it must be the same. and i add

up my own comments to those two files for grading. i test the client and server file to using my local internet ip address

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

import java.io.\*; // import library for input and output

import java.net.\*; // import librart for network

// define a client class

public class InetClient{

// main function to run client class

public static void main (String args[]) {

String serverName;

// if else condition to define the arguments whether is local host by determine its argument string length

if (args.length < 1)

serverName = "localhost";

else

serverName = args[0];

System.out.println("sample Inet Client, 1.8.\n");

System.out.println("Using server: " + serverName + ", Port: 2000"); // printout server name and port number

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

try {

String name;

do {

System.out.print("Enter a hostname or an IP address, (quit) to end: "); //print out message and

// ask user to type in the hostname and ip address which server will look up

System.out.flush ();

name = in.readLine ();

if (name.indexOf("quit") < 0) // when type in quit it will quit running client

getRemoteAddress(name, serverName);

} while (name.indexOf("quit") < 0);

System.out.println ("Cancelled by user request.");

} catch (IOException x) {x.printStackTrace ();}// just catch input and output exception in case.

}

static String toText (byte ip[]) { // not usefully to use toText function in this assignment.

StringBuffer result = new StringBuffer ();

for (int i = 0; i < ip.length; ++ i) {

if (i > 0)

result.append (".");

result.append (0xff & ip[i]);

}

return result.toString ();

}

static void getRemoteAddress (String name, String serverName){

// define socket varible, input and output stream

Socket sock;

BufferedReader fromServer;

PrintStream toServer;

String textFromServer;

try{

// open connection to the serever by typing user's own port number

sock = new Socket(serverName, 2000);

// assign value recieved from server and message will be send back to server by calling sock with getInputStream() and getOutputStream()

fromServer = new BufferedReader(new InputStreamReader(sock.getInputStream()));

toServer = new PrintStream(sock.getOutputStream());

// the hostname or ip adrress will be send to server

toServer.println(name);

toServer.flush();

// printout result send from server

for (int i = 1; i <=3; i++){

textFromServer = fromServer.readLine();

if (textFromServer != null)

System.out.println(textFromServer);

}

sock.close(); // close this socket connection

} catch (IOException x) {

System.out.println ("Socket error.");

x.printStackTrace ();

}

}

}