//start of file. Referanced docs.oracle.com for what some things were(ie. java.io, java.net, BufferedReader, Socket, PrintStream, and IOException) but put everthing into my own words strictly refeanced the site to see what it was.

//Some comments are the same as in InetServer as for they are the same code.

import java.io.\*; //imports all the the directories located in java.io so that when file is complied it classes. For example, in this code we use java.io.ioexception and import java.io.bufferedreader. By having the java.io.\* we do not need to import two things and we can just inport one allowing our code to be cleaner. The specific class io allows us to get a user input and output based on that input.

import java.net.\*; // Just like in the above input the net.\* imports all directories in the java.net so we dont have to add mutliple imports. The java.net specifically is a directory in java used to implament networking apps. It allows us to use things like Socket and ServerSocket in our code.

public class InetClient{ //new class InetClient

public static void main (String args[]) {

String serverName; //initialize serverName as a String varable.

if (args.length < 1) serverName = "localhost";//checks to see if args is less than one. If true, serverName is set equal to localhost.

else serverName = args[0];//otherwise serverName is set equal to args[0].

System.out.println("Jess Bender's Inet Client, 1.8.\n");//prints statement on terminal.

System.out.println("Using server: " + serverName + ", Port: 1581"); //prints Using server: plus whatever is saved in serverName plus the port.

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

try {//trys the following code, if fails jumps to catch.

String name;

do {

System.out.print("Enter a hostname or an IP address, (quit) to end: ");//prints statement on terminal.

System.out.flush ();

name = in.readLine ();//assigns the text from the BufferedReader in to name.

if (name.indexOf("quit") < 0)//checks to see if name = quit

getRemoteAddress(name, serverName);//calls function getRemoteAddress below and puts in name and serverName for the 2 string varables.

}//closes do

while (name.indexOf("quit") < 0);// keep doing the do above until name = quit

System.out.println ("Cancelled by user request.");//when name = quit print this statement.

}//closes try

catch (IOException x) {x.printStackTrace ();} //catches IOExeption when try fails and prints the error.

}//closes main

static String toText (byte ip[]) { //new method

StringBuffer result = new StringBuffer (); //new StringBuffer called result.

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

if (i > 0) result.append ("."); // add . to result if i is larger than 0.

result.append (0xff & ip[i]); // if not greater than 0 0xff & ip[i] gets added to result.

}//closes for

return result.toString ();// returns result as a string

}//closes toText

static void getRemoteAddress (String name, String serverName){ //// gets it name and serverName variables from above in the do statement

Socket sock; //makes Socket variable called sock.

BufferedReader fromServer; //makes BufferedReader variable called fromServer.

PrintStream toServer; //makes PrintStream variable called toServer.

String textFromServer; //makes String variable called textFromServer.

try{//trys the following code, if fails jumps to catch.

sock = new Socket(serverName, 1581); //assigns sock toa new Socket with serverName and the port 1581

fromServer = new BufferedReader(new InputStreamReader(sock.getInputStream())); //assigns fromServer to a new BufferedReader that gets the input from sock

toServer = new PrintStream(sock.getOutputStream()); //assigns toServer to a new PrintStream that writes the input from sock

toServer.println(name); toServer.flush();

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

textFromServer = fromServer.readLine();//assigns textFromServer to the text in fromServer.

if (textFromServer != null) System.out.println(textFromServer); //checks to make sure textFromServer is not null/ empty and id so prints whatever was in textFromServer.

}//closes for

sock.close();//closes sock

} //closes try

catch (IOException x) { //catches IOExeption when try fails.

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

x.printStackTrace ();//prints Socket error and printStackTrace witch prints details about the error including the line number where the error occurred.

}//closes catch

}//closes getRemoteAddress

}//closes class InetClient

//start of file. Referanced docs.oracle.com for what some things were(ie. java.io, java.net, BufferedReader, Socket, PrintStream, and IOException) but put everthing into my own words strictly refeanced the site to see what it was.

//Some comments are the same as in InetClient as for they are the same code.

import java.io.\*; //imports all the the directories located in java.io so that when file is complied it classes. For example, in this code we use java.io.ioexception and import java.io.bufferedreader. By having the java.io.\* we do not need to import two things and we can just inport one allowing our code to be cleaner. The specific class io allows us to get a user input and output based on that input.

import java.net.\*; // Just like in the above input the net.\* imports all directories in the java.net so we dont have to add mutliple imports. The java.net specifically is a directory in java used to implament networking apps. It allows us to use things like Socket and ServerSocket in our code.

class Worker extends Thread { //makes our worker.class then file is complied

Socket sock; //makes a new Socket called sock.

Worker (Socket s) {sock = s;}

public void run(){

PrintStream out = null; //assigns a PrintStream variable called out to null. PrintStream is a directory in the java.io package that writes output data.

BufferedReader in = null; //assigns a BufferedReader variable called in to null. BufferedReader is a directory in the java.io package that reads text from an input including sockets

try { //will try the things within the try otherwise will skip to catch

in = new BufferedReader // assigns in to a new BufferedReader

(new InputStreamReader(sock.getInputStream()));

out = new PrintStream(sock.getOutputStream()); // assigns out to a new PrintStream that gets its data from the Socket sock .getInputStream() allows us to get that output

try { //will try the things within the try otherwise will skip to catch

String name;//makes a new string called name

name = in.readLine ();//assigns the string variable name to what was read in the the varaible in.

System.out.println("Looking up " + name); //prints Lokking up and whatever was saved into the variable name to the terminal.

printRemoteAddress(name, out); //calls the static void printRemoteAddress a few lines down and sets the String to whatever the name vaible was assigned to and the PrintStream to whatever the out vaible was assigned to.

}//closes second try

catch (IOException x) { //catches IOExeption when try fails

System.out.println("Server read error"); x.printStackTrace (); //prints Server read error and printStackTrace witch prints details about the error including the line number where the error occurred.

} //closes catch

sock.close(); // closes the Socket called sock

} //closes first try

catch (IOException ioe) {System.out.println(ioe);} //catches IOExeption when try fails and prints the error

} // closes run()

static void printRemoteAddress (String name, PrintStream out) { // gets it name and out variables from above in the second try.

try { //will try the things within the try otherwise will skip to catch

out.println("Looking up " + name + "..."); //prints Looking up and whever is stored in the name vaiable and ...

InetAddress machine = InetAddress.getByName (name);

out.println("Host name : " + machine.getHostName ());//prints Host name and gets the host name from machine and print it.

out.println("Host IP : " + toText (machine.getAddress ())); //prints Host IP and IP address from machine in text format.

}//closes try

catch(UnknownHostException ex) { //catches UnknownHostException when try fails

out.println ("Failed in atempt to look up " + name); //prints Failed in atempt to look up and whatever was stored in the name variable.

}//closes catch

}//closes printRemoteAddress

static String toText (byte ip[]) {

StringBuffer result = new StringBuffer (); // makes a new StringBuffer called result

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

if (i > 0) result.append ("."); // add . to result if i is larger than 0.

result.append (0xff & ip[i]); // if not greater than 0 0xff & ip[i] gets added to result.

}//closes for

return result.toString (); //return the result in a string format.

}//closes toText

}//closes Worker class

public class InetServer { //new class InetServer

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

int q\_len = 6; //initialize q\_len as an int and assigns it to 6

int port = 1581;//initialize port as an int and assigns a port number(can be changed)

Socket sock; //makes a Socket called sock

ServerSocket servsock = new ServerSocket(port, q\_len); //makes a new ServerSocket called servsock and puts the port number and q\_len assigned above.

System.out.println

("Jess Bender's Inet server 1.8 starting up, listening at port 1581.\n");//prints that text

while (true) { //keeps running while its true

sock = servsock.accept(); //assignes sock to an accepted servsock.

new Worker(sock).start(); //calls on worker class with the sock assigned in it.

} //closes while

}//closes main

}// closes InetServer

//end of file