/\*\*\* Assignment TCP Multiuser chat client \*\*\*/

import java.io.\*;

import java.net.\*;

public class multclient

{

public static void main(String argv[]) throws Exception {

String sentence;

String modifiedSentence;

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

Socket clientSocket = new Socket("localhost", 6789);

while (true)

{

DataOutputStream outToServer =new DataOutputStream(clientSocket.getOutputStream());

BufferedReader inFromServer =new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();

outToServer.writeBytes(sentence + '\n');

if (sentence.equals("EXIT")) {

break;

}

modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

}

clientSocket.close();

}

}

/\*\*\* Assignment TCP Multiuser chat server\*\*\*/

import java.io.\*;

import java.net.\*;

public class multserver

{

public static void main(String argv[]) throws Exception

{

ServerSocket welcomeSocket = new ServerSocket(6789);

Responder h = new Responder();

// server runs for infinite time and

// waits for clients to connect

while (true)

{

Socket connectionSocket = welcomeSocket.accept();

// on connection establishment start a new thread for each client

// each thread shares a common responder object

// which will be used to respond every client request

Thread t = new Thread(new MyServer(h, connectionSocket));

// start thread

t.start();

}

}

}

class MyServer implements Runnable

{

Responder h;

Socket connectionSocket;

public MyServer(Responder h, Socket connectionSocket)

{

this.h = h;

this.connectionSocket = connectionSocket;

}

//@Override

public void run()

{

while (h.responderMethod(connectionSocket))

{

try

{

// once an conversation with one client done,

// give chance to other threads

// so make this thread sleep

Thread.sleep(5000);

} catch (InterruptedException ex) { }

}

try {

connectionSocket.close();

} catch (IOException ex) { }

}

}

class Responder

{

String serverSentence;

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

// on client process termination or

// client sends EXIT then to return false to close connection

// else return true to keep connection alive

// and continue conversation

synchronized public boolean responderMethod(Socket connectionSocket)

{

try {

BufferedReader inFromClient =new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));

DataOutputStream outToClient =new DataOutputStream(connectionSocket.getOutputStream());

String clientSentence = inFromClient.readLine();

// if client process terminates it get null, so close connection

if (clientSentence == null || clientSentence.equals("EXIT")) {

return false;

}

if (clientSentence != null) {

System.out.println("client : " + clientSentence);

}

serverSentence = br.readLine() + "\n";

outToClient.writeBytes(serverSentence);

return true;

} catch (SocketException e) {

System.out.println("Disconnected");

return false;

} catch (Exception e) {

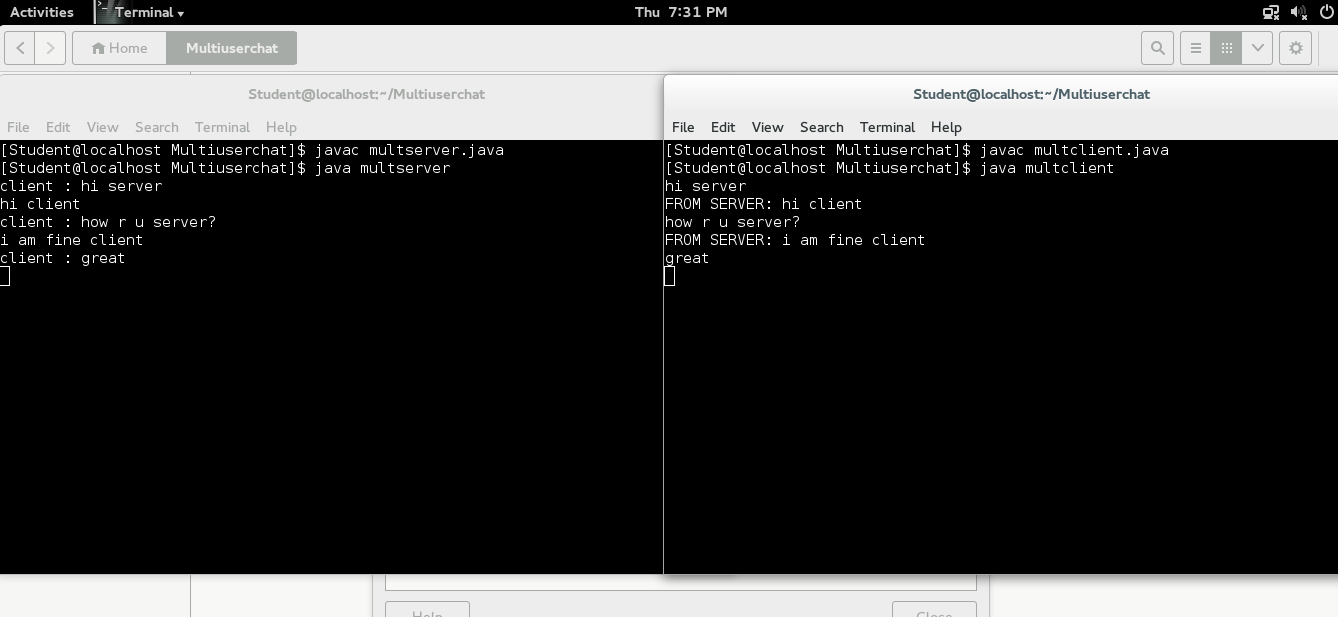
e.printStackTrace();

return false;}

}

}

\*\*\* OUTPUT of TCP Multiuser chat \*\*\*



/\*\*\* Assignment TCP Peer to peer chat client\*\*\*/

import java.io.\*;

import java.net.\*;

import java.net.Socket;

class client

{

public static void main(String argv[]) throws Exception

{

String sentence,s1;

String modifiedSentence;

BufferedReader inFromUser =

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

Socket clientSocket = new Socket("127.0.0.1", 6666);

DataOutputStream outToServer =

new DataOutputStream(clientSocket.getOutputStream());

BufferedReader inFromServer =

new BufferedReader(new

InputStreamReader(clientSocket.getInputStream()));

System.out.println("Enter your Message:");

sentence = inFromUser.readLine();

outToServer.writeBytes(sentence + '\n');

s1 = inFromServer.readLine();

System.out.println("FROM SERVER: " + s1);

clientSocket.close();

}

}

/\*\*\* Assignment TCP Peer to peer chat server\*\*\*/

import java.io.\*;

import java.net.\*;

import java.net.ServerSocket;

class server

{

public static void main(String argv[]) throws Exception

{

String clientSentence;

String sentence;

ServerSocket welcomeSocket = new ServerSocket(6666);

Socket connectionSocket = welcomeSocket.accept();

BufferedReader inFromClient =

new BufferedReader(new

InputStreamReader(connectionSocket.getInputStream()));

DataOutputStream outToClient =

new DataOutputStream(connectionSocket.getOutputStream());

BufferedReader inFromUser =

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

clientSentence = inFromClient.readLine();

System.out.println("FROM CLIENT: " +clientSentence );

System.out.println("Enter your Message:");

sentence = inFromUser.readLine();

outToClient.writeBytes(sentence + '\n');

}

}

\*\*\* OUTPUT of TCP Peer to Peer \*\*\*

