package a;

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

public class ChattingServer {

private Socket socket;

private String username;

//Server socket that we use to accept the new connection from the client

private ServerSocket serverSocket;

//Using to store the users (connection) to the server

private HashMap <Socket,DataOutputStream> users\_list = new HashMap();

//the constructor of server

public ChattingServer(int port) {

listen(port);

}

public void listen(int port) {

try {

serverSocket = new ServerSocket(port);

//ready informed message

System.out.println("Ready to connect..." + serverSocket);

while (true) {

//take the connection to the server socket

socket = serverSocket.accept();

//inform connecting

System.out.println("Connecting from" + socket);

//create DataOutputStream

DataOutputStream outputs = new DataOutputStream(socket.getOutputStream());

//save the information into the HashMap

users\_list.put(socket, outputs);

//create new thread

new ServerThread(this, socket);

}

} catch (IOException ex) {

Logger.getLogger(ChattingServer.class.getName()).log(Level.SEVERE, null, ex);

}

}

void sendToAll(String mess) {

//use Synchronized to avoid screwing up by different sub-process like remove the client

synchronized(users\_list) {

//loop through the hashmap

for (Socket user: users\_list.keySet()) {

try {

users\_list.get(user).writeUTF(mess);

} catch (IOException ie) {

System.out.println(ie);

}

}

}

}

//Remove the connection

void removeConnection(Socket s) {

//informing

System.out.println("Connection removing" + s + "...");

//remove it from hashmap

users\_list.remove(s);

//ensuring the connection is closed

try {

s.close();

}

catch (IOException ie) {

System.out.println("Cannot Close" + s);

}

}

}