|  |  |
| --- | --- |
| H:\LOGO-NXV\Hai__090908__02_1_den.jpg | Faculty of Information Technology  H A N O I U N I V E R S I T Y |

61FIT3NPR – Network Programming   
Tutorial week 6

Java TCP Socket and Thread

1. Exercise 1:

TCP socket and Create a thread by extending Thread at server to maintain connection from client

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.OutputStreamWriter;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**public** **class** ServerProgram {

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

ServerSocket listener = **null**;

System.***out***.println("Server is waiting to accept user...");

**int** clientNumber = 0;

**try** {

listener = **new** ServerSocket(7777);

} **catch** (IOException e) {

System.***out***.println(e);

System.*exit*(1);

}

**try** {

**while** (**true**) {

Socket socketOfServer = listener.accept();

**new** ServiceThread(socketOfServer, clientNumber++).start();

}

} **finally** {

listener.close();

}

}

**private** **static** **void** log(String message) {

System.***out***.println(message);

}

**private** **static** **class** ServiceThread **extends** Thread {

**private** **int** clientNumber;

**private** Socket socketOfServer;

**public** ServiceThread(Socket socketOfServer, **int** clientNumber) {

**this**.clientNumber = clientNumber;

**this**.socketOfServer = socketOfServer;

// Log

*log*("New connection with client# " + **this**.clientNumber + " at " + socketOfServer);

}

@Override

**public** **void** run() {

**try** {

BufferedReader is = **new** BufferedReader(**new** InputStreamReader(socketOfServer.getInputStream()));

BufferedWriter os = **new** BufferedWriter(**new** OutputStreamWriter(socketOfServer.getOutputStream()));

**while** (**true**) {

String line = is.readLine();

os.write(">> " + line);

os.newLine();

os.flush();

**if** (line.equals("QUIT")) {

*log*("Client # " + **this**.clientNumber + " quit!");

**break**;

}

}

} **catch** (IOException e) {

System.***out***.println(e);

e.printStackTrace();

}

}

}

}

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.Date;

**public** **class** ClientDemo {

**public** **static** **void** main(String[] args) {

**final** String serverHost = "localhost";

Socket socketOfClient = **null**;

BufferedWriter os = **null**;

BufferedReader is = **null**;

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

String s;

**try** {

socketOfClient = **new** Socket(serverHost, 7777);

os = **new** BufferedWriter(**new** OutputStreamWriter(socketOfClient.getOutputStream()));

is = **new** BufferedReader(**new** InputStreamReader(socketOfClient.getInputStream()));

} **catch** (UnknownHostException e) {

System.***err***.println("Don't know about host " + serverHost);

**return**;

} **catch** (IOException e) {

System.***err***.println("Couldn't get I/O for the connection to " + serverHost);

**return**;

}

**try** {

String responseLine;

os.write("HELLO! now is " + **new** Date());

os.newLine();

os.flush();

responseLine = is.readLine();

System.***out***.println("Server: " + responseLine);

os.write("I am Tom Cat");

os.newLine();

os.flush();

responseLine = is.readLine();

System.***out***.println("Server: " + responseLine);

**while** (**true**)

{

System.***out***.println("Please enter your message");

s = inFromUser.readLine();

os.write(s);

os.newLine();

os.flush();

responseLine = is.readLine();

System.***out***.println("Server: " + responseLine);

**if** (s.equals("QUIT") | (responseLine == **null**) ) {

**break**;

}

}

os.close();

is.close();

socketOfClient.close();

} **catch** (UnknownHostException e) {

System.***err***.println("Trying to connect to unknown host: " + e);

} **catch** (IOException e) {

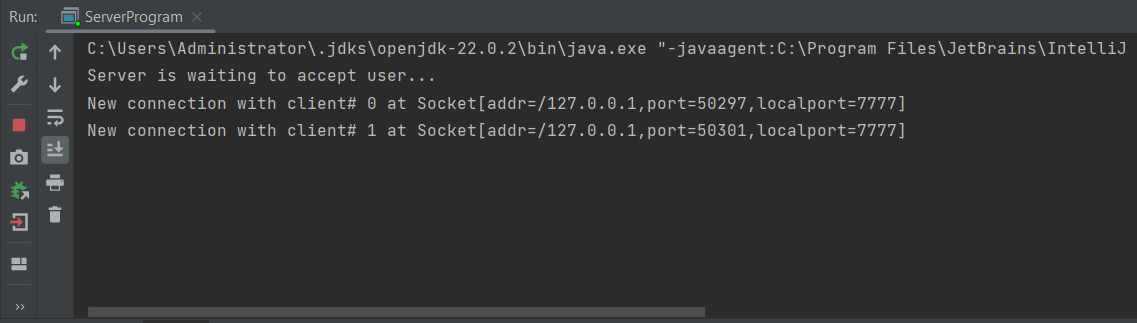
System.***err***.println("IOException: " + e);

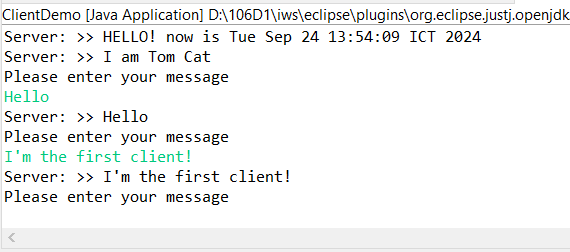
}

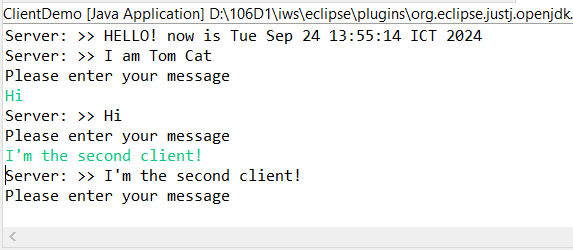
}

}

Run ServerProgram and some ClientDemo:







1. Redo the exercise 1 of the Tutorial Week 5 with multithread server (each thread maintains the connection with 1 client).
2. Redo the SquareServer, make it become a multithread server, so it can give service to multiple client.