Abstract TCP Server – part 1

Mission

To build a framework for ease the implementation of TCP servers, this include concurrent server with a soft-closedown function.

Background

Straightforward TCP server programming from 3rd semester e.g. see

- Simple tcp server
- Concurrent tcp server

Template Design Pattern see

- C#Note chap. OOProg3 pp.1-17 + 43-47
- Dotfactory template in C#
- C# design patterns in general (extra)

Comments

- Comments of code -- Corey's Tutorial: How to Comment & Document Your Code (video)
- Insert XML Comments https://docs.microsoft.com/en-us/dotnet/csharp/codedoc
- How to use doxygen : http://www.doxygen.nl/manual/starting.html

Assignment 1 – A simple Server

Create a .Net Core Console application and create a simple Echo TCP server on port 7007 (for inspiration see https://github.com/rf18da2b3-3b/ClassDemoEchoServer/tree/master/ClassDemoEchoServer)

Ensure your server is concurrent by using a Task.Run to execute each client.

You are NOT asked to implement a Client-program.

Try your implementation with SocketTest:
Download and unzip SocketTest : https://sourceforge.net/projects/sockettest/

If SocketTest do not work, you possible need a Java Runtime Environment so download and install jre : http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

Assignment 2 – Make a Simple Framework for TCP Servers

Now you are to build a simple Framework supporting easy creation of a TCP-server.

Create a **.Net Core Class Library** and make a folder “TCPServer”.
In this folder, create an **abstract** class ‘AbstractTCPServer’, where you insert most of the code from the example from assignment 1.

The only exception is the part where you read from the StreamReader and write the string back to the StreamWriter.

This part should be extracted into a method ‘TcpServerWork’, with the reader- and writer-stream as parameters. This method should be abstract. (THE TEMPLATE METHOD)

Assignment 3 – Improve the Simple Framework

You can improve your AbstractTCPServer by:

- Pass the port number as a parameter to the AbstractTCPServer
- Pass a name to the Server e.g. “EchoServer”
- Printout state information like “started at port xxxx” etc.

Assignment 4 – Soft Shutdown of the Simple Framework

Instead of having, an infinite loop support a soft shutdown.

To support soft shutdown in the AbstractTCPServer do following steps.

- Introduce a bool variable running which is initial true.
- Use this variable in the while-loop.
- Implement a method to set the running variable to false
- Implement another method, which listen to the port number of the server plus one e.g. the server 7007, while the ‘stop server’ have 7008. When a client connect to this stop server – you could make some check for validity – then call the method to change the running variable.
- BEFORE the while-loop, make a new Task where to start this ‘stop server’.
- Do not call AcceptTCPClient direct. Make an if-statement if (listener.Pending()) -> then call AcceptTcpClient else wait XX sec (using Thread.Sleep(2*1000) = 2 sec)

Assignment 5 – Add comments to your Simple Framework

You must add (xml comments or /// comments) to your public methods in the ‘AbstractTCPServer’.

Make the comments be a good documentation for others to use i.e. describe the methods, its parameters as well as return values.

Use Doxygen to generate the html-help pages.

Assignment 6 – Use your Simple Framework

Create a new .Net Core Console Application project / or solution.

In this project make a folder ‘Server’, where you create a new class e.g. ‘MyServer’, that inherit from the abstract server from the Library. (I.e. add reference to your Library).

Implement the abstract method – again as echo, perhaps by capitalize the letters.

In the main, make an object of your Server and start it. – Try your server with SocketTest

I hope that you find it easier this time to create a server.