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)

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 it 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 2a – Make a Simple Framework

Now you are to build a simple Framework for more easily creating a TCP-server.

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

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 2b – Use the Simple Framework

Create a new .Net Core Console Application project.

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.

Assignment 3a – Improve the Simple Framework (extra)

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 3b – Soft Shutdown of the Simple Framework (extra)

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 server – you could make some check for validity – the call the method to change the running variable.
- BEFORE the while-loop, make a new Task where this ‘stop server’ is started.
- 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 3c – Logging Information of the Simple Framework (extra)

Add logging information to your TCP-server.

In Main method add Trace.Listeners.Add(.. add listener e.g. new ConsoleTraceListener()),

Then write Trace.TraceInformation(text), Trace.TraceWarning, Trace.Error to log information.

See more https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.trace?view=netcore-3.0

And more tracelisteners (to files, xml etc.) see: https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.tracelistener?view=netcore-3.0