26- Maintaining a list of multiple clients

1. Hi, in this video I am going to show you how to handle multiple clients on your server.
2. It has been the most asked for topic since the launch of this course.
3. First of all I will open the server project which can be downloaded from Lecture 25 resources section.
4. The file “Form1.cs” is opened in design view.
5. I’ve added a list box “lbClients” to the form which is going to show the currently connected clients.
6. Let’s take a look inside the code.
7. I have created a class “ClientNode” to store information relevant to a client node
8. We are going to maintain a list of objects of this class.
9. When a new client will connect, we will add an item to our list.
10. When an existing client will disconnect, we will remove it from our list.
11. This class is deriving from IEquatable, class and if we scroll down we will see it contains definition of method “IEquatable.Equals”.
12. This method is going to be used in our code to locate a client node, later on.
13. The class has also defined its own version of ToString method as well.
14. Now I will switch to Form1.cs and add create a list object on class level.
15. I’ve instantiated this object in form constructor.
16. Next, we will go to the “onCompleteAccept” callback method.
17. The “TcpListener” definition is followed by a local “TcpClient” object tclient, this one was not present in last video.
18. Right after the TcpClient definition I have defined an object of ClientNode and assigned it a null value.
19. Inside the try block I will call EndAcceptTcpClient method which will return an instance of TCPClient.
20. We start listening for new clients in the next line.

21. Afterwards, you see a lock operation on “mlClientSocks” list object.
22. The lock is used to ensure thread mutual exclusion
23. You can see that in the next statement we create a new object of ClientNode , and store it in local variable “cnode”, we also add the object to our list of clients mlClientSocks.
24. If we go to the definition of this constructor, we will see that the first parameter is a TCPClient, second is a byte buffer for transmitting information, third a byte buffer for receiving information, and the last one is the unique string which will be used to identify the client later on.
25. Back in the “EndAcceptClient” method of our form code, I’m going to update the list view with new client’s information. I had to set the flag ***CheckForIllegalCrossThreadCalls*** to false in order to add a value to the list view.
26. Right after the lock block, I’m going to start reading data from the client which just got connected.
27. Now are are going to go in the callback onCompleteReadFromTCPClientStream method.
28. Here, I’ve added an object of ClientNode and called it cn.
29. First thing we’re going to do in this method is to lock our list of clients.
30. I’ve used Linq to find the object of client node class. in our list of clients. The “iEquatable.Equals” method we saw earlier in ClientNode class definition is going to be used by this operation internally.
31. When a client will be disconnected, we’re going to remove it from our list of clients and from our listbox as well.
32. Similar locking and list removal code is present in the catch block below.
33. In the next line of code, I am extracting data from the specific “ClientNode” object’s receive buffer RX and turning it into a string.
34. After printing the information, I’m refreshing the receive buffer.
35. Finally, you see that I am beginning the read operation on receive buffer of our specific “ClientNode” object.

36. Now let's talk about the send data operation.
37. We are going to use the currently selected item in the "Clients" listview.
38. On form design view, I will double click the "Send" button.
39. Inside, I am going to make sure that the listbox of clients contains some values and return if that's not true.
40. Check for empty payload is also in place.
41. As you would've expected, we've defined an object of ClientNode and locked the list of clients as well.
42. I'll use the currently selected item of client listbox to dig out relevant client node object.
43. First thing, we'll clean the transmission byte buffer of the client node object.
44. We put our payload in relevant transmission buffer and send it through relevant "TCPClient" object, of course we have a few sanity checks before we do so.
45. We are all set for a demo in next video!