Socket client

Separate facade/model and user interface

(Remote facade)

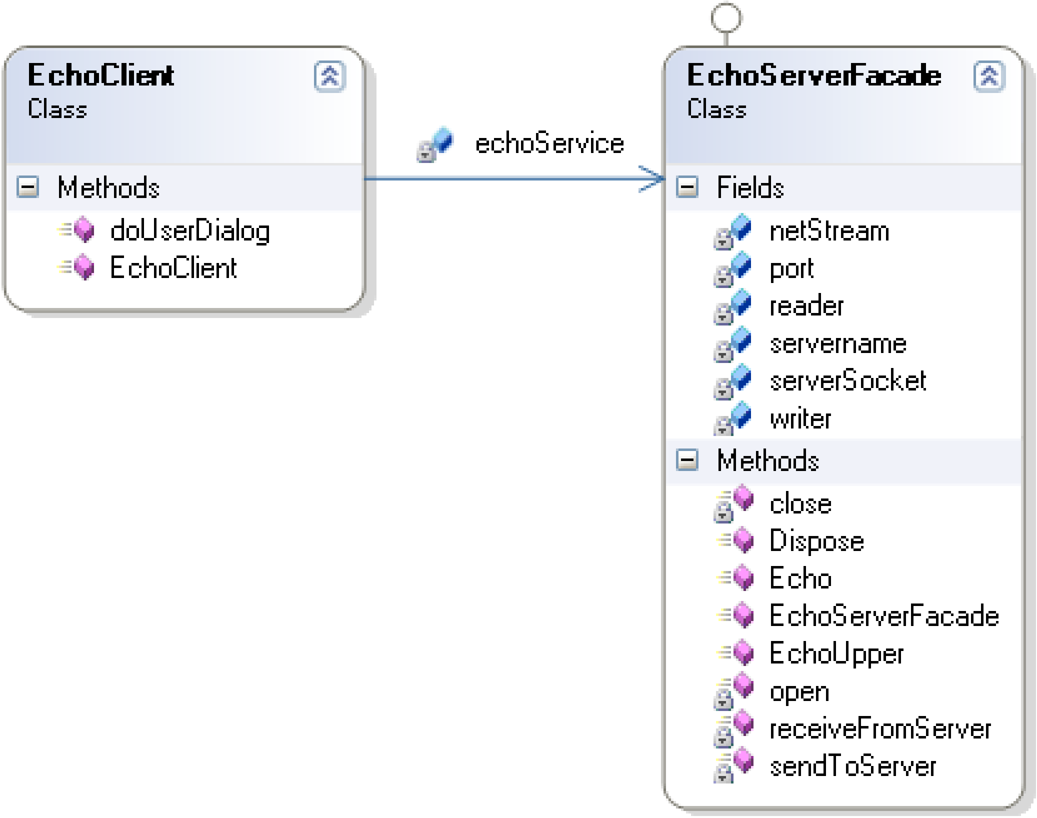
Partial of the class EchoServerFacade

```
private void sendToServer(string text)
{
    writer.WriteLine(text);
    writer.Flush();
}
private string receiveFromServer()
{
    try
    {
        return reader.ReadLine();
    }
    catch
    {
        return null;
    }
}

// Remote metoder
public string Echo (string text)
{
    #if TELNET_DIALOG       // Hvis compilerdirektiv er sat medtages koden
        string komandoklarbesked = receiveFromServer(); // vent på klarbesked
        Console.WriteLine("Komando-klarbesked fra server:" + komandoklarbesked);
    #endif

    sendToServer("echo");
    sendToServer(text);
    string response = receiveFromServer();
    return response;
}
```

SocketClientMedRemoteFasade_ConsolApplikation

- In this edition shows fundamentals how to divide a facade, which handles the communication with the server and user interface.
- This version uses a console interface.
- EDITION USING CONNECTED CONNECTION AND SEMILAR SERVER MUST SELECTED FOR RUNNING

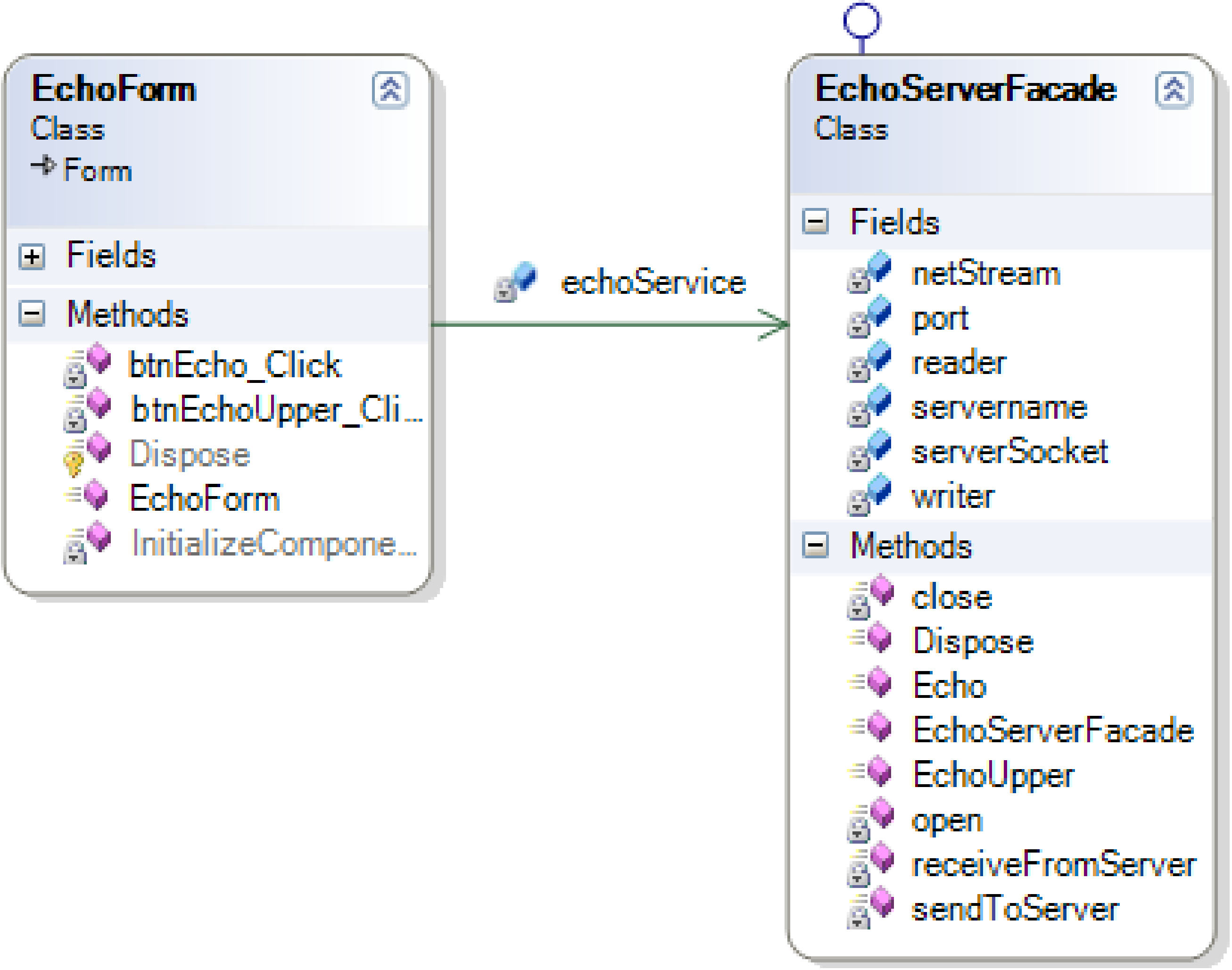
SocketClientMedRemoteFasade_ConsolApplikation



SocketClientMedRemoteFasade_WindowsApplication

- Same separation as in SocketClientMedRemoteFasade_ConsolApplikation, but with a windows user interface.
- EDITION USING CONNECTED CONNECTION AND SEMILAR SERVER MUST SELECTED FOR RUNNING

SocketClientMedRemoteFasade_WindowsApplication



Socket server

Separation of facade/model and communication with client

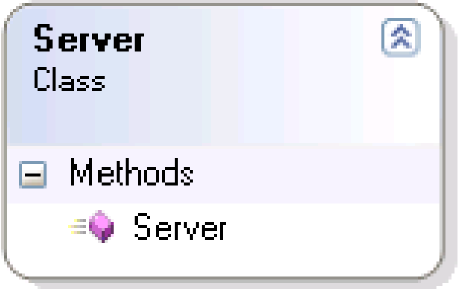
Socket server

Stateless services

SocketServerCmdBasedStatic

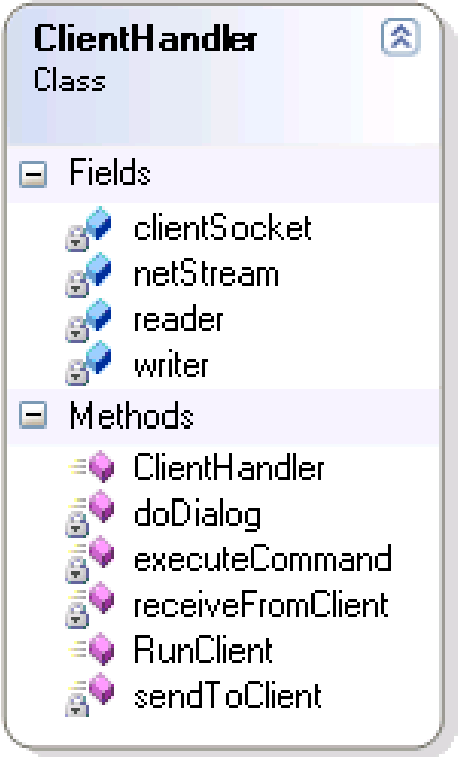
- This edition shows basic how to divide the communication with the client and processing of commands.
- Comand execution is separated in the class EchoService with static methods - no memory.
- Furthermore, the actual dialog is divided in the establishment and execution of the comands.

SocketServerCmdBasedStatic



Server
Class

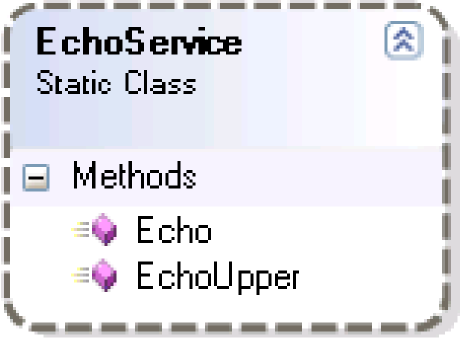
Methods
- Server



ClientHandler
Class

Fields
- clientSocket
- netStream
- reader
- writer

Methods
- ClientHandler
- doDialog
- executeCommand
- receiveFromClient
- RunClient
- sendToClient



EchoService
Static Class

Methods
- Echo
- EchoUpper

Socket server

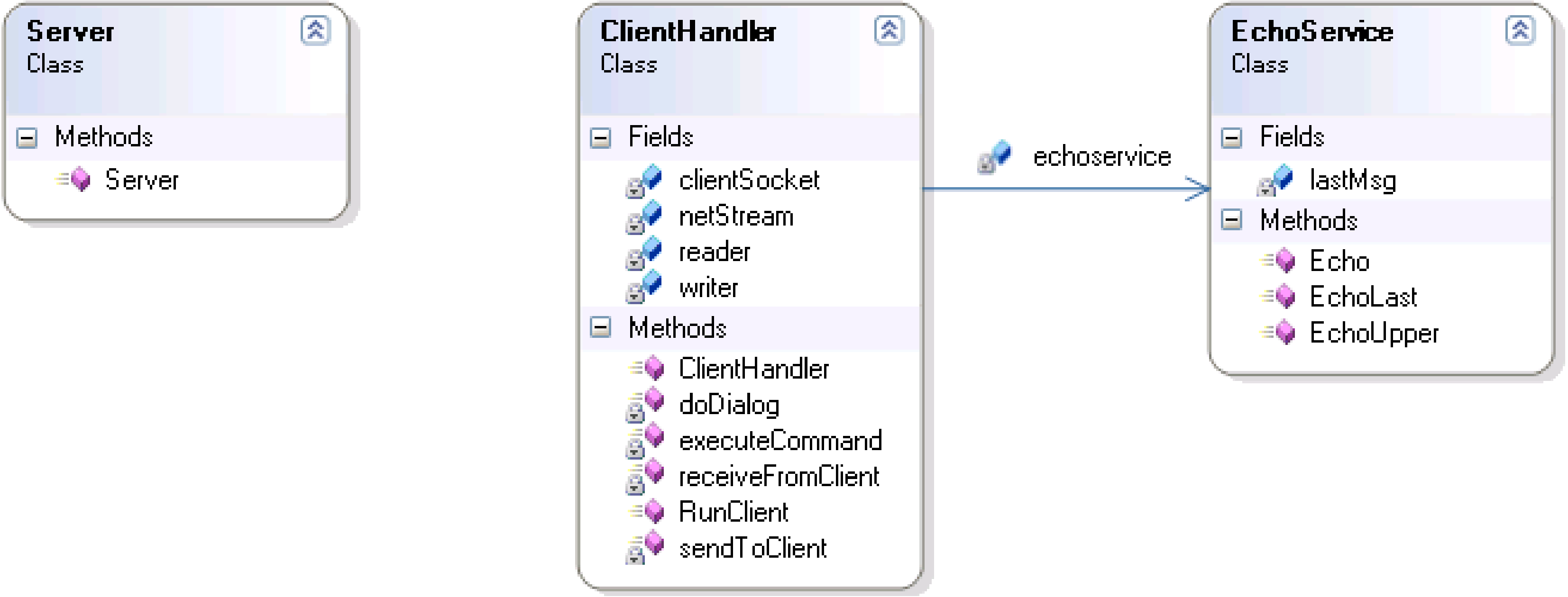
Statefull services

Using an full time open socket-connection

SocketServerCmdConnected

- In this edition the EchoService class changed to be an object instead witch now also has got memory
- During the dialogue the socket connection is keept open and service-object lives during the whole dialogue
- We now got a statefull dialog.

SocketServerCmdConnected



Socket server

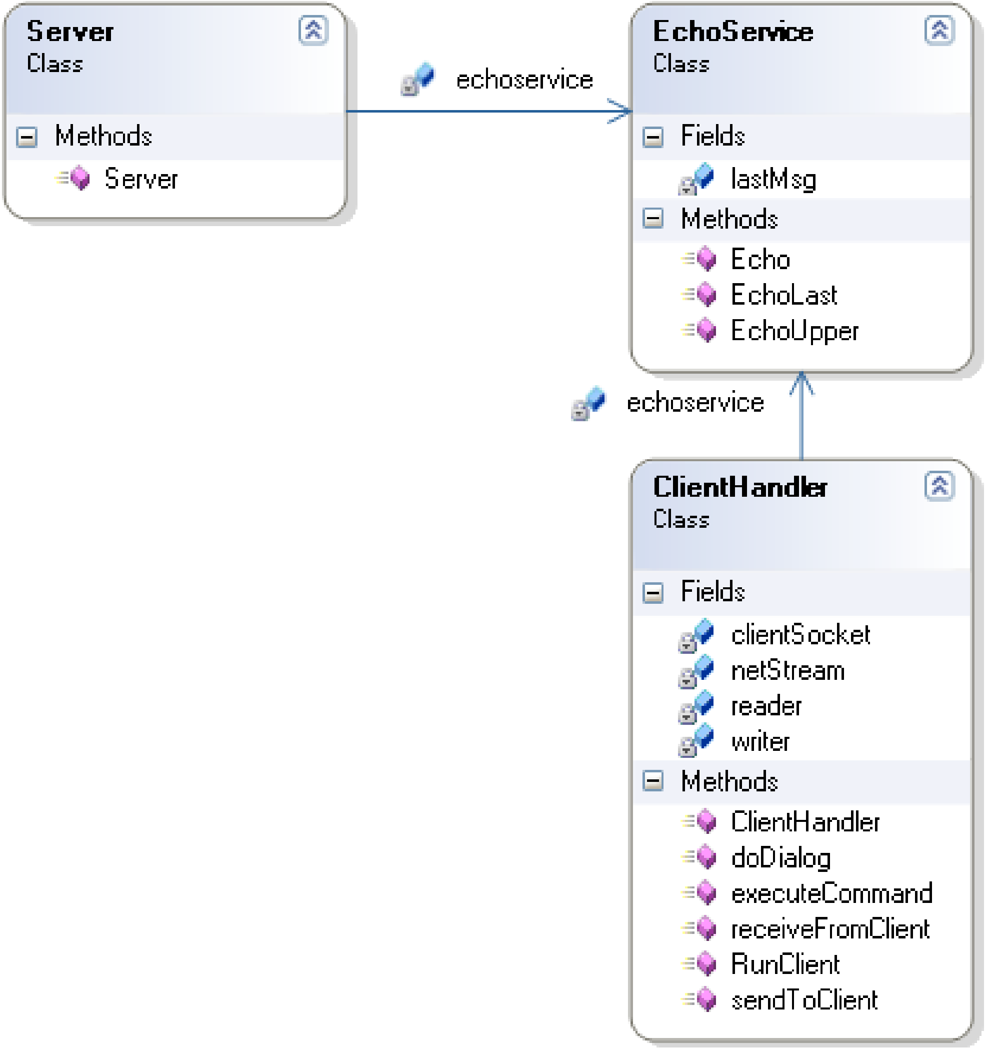
Statefull services

Shared service for more clients

SocketServerCmdDisconnected_V01

- In this edition the socketconnection is closed between each command.
- To get the service object to survive, it is moved to server-class.
- There exists in the edition only one service object that so now will be shared to all clients (connections).
- The version is taking care of secured synchronization in the service object.
- This version is only made to demonstrate that there can be a form of stateful communications.

SocketServerCmdDisconnected_V01



Socket server

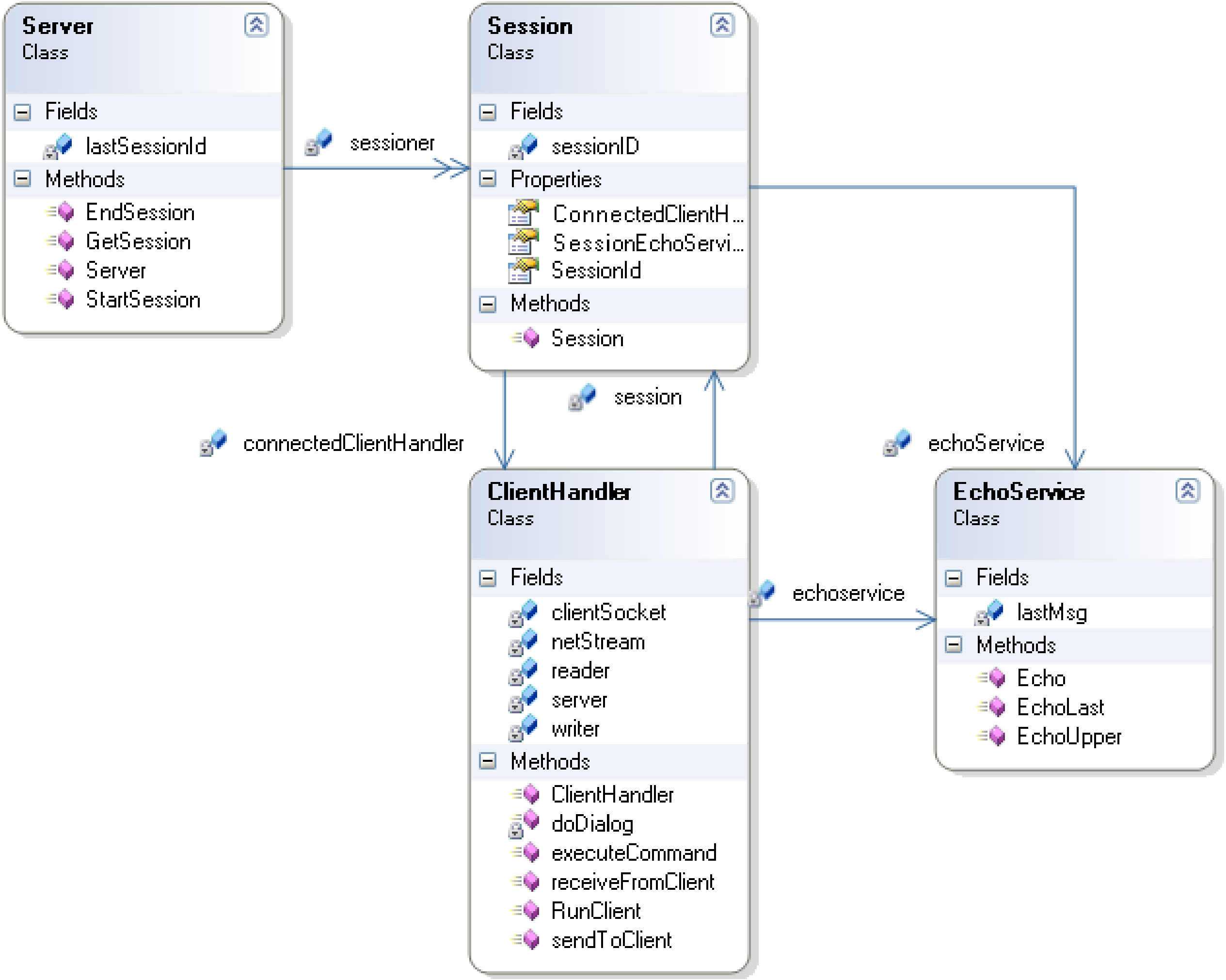
Statefull services for each client (session)

Socket-connection is closed after each command

SocketServerCmdDisconnectedSession

- In this edition is added a new class, which must contain information for each client.
- Socket server contains a collection of these objects and gives each client access to:
  - create an object, access its object and maintain it.
    This allows each client now get its own object and be stateful, without the influence of other clients.
  - The socket-server ”keeps" the collection, in this manner it will ”live" in the full lifetime of the socket-server.
  - Connection is closed after execution of every client command.
  - This version have not included a secured synchronization access to the service-objektet.

SocketServerCmdDisconnectedSession



Socket server

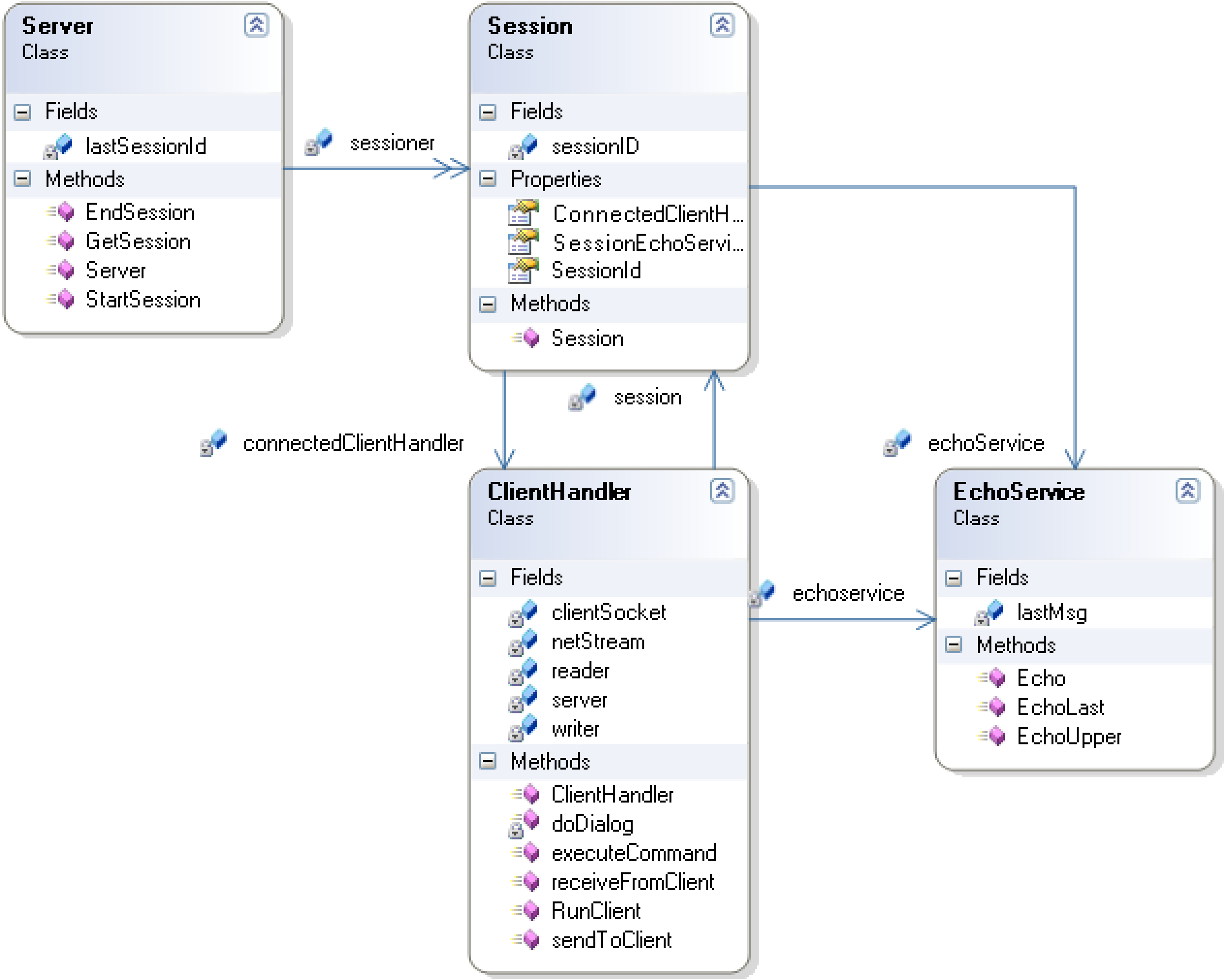
Statefull services for each client (session)

Socket-connection could be keept open or closed after each command

SocketServerCmdDisconnectedOnOff

- This version extends the previous one in that the connection can be kept open for more commands,
- but also can be interrupted (extra command "pause").
- The changes are minimal and show that it is now possibly to keep a state full communication with both a maintained socket connection and by an interrupted.
- The solution therefore has greater possibilities than the solution where the socket connection should be kept open during the communication with the client.
- This version have not included a secured synchronization access to the service-objektet.

SocketServerCmdDisconnectedOnOff



SocketServerCmdDisconnectedOnOff

Session klassen

```
public class Session
   {
       private string sessionID;
       private EchoService echoService;
       private ClientHandler connectedClientHandler;

       public Session(string sessionID)
       {
           this.sessionID = sessionID;
       }
       public string SessionId
       {
           get { return this.sessionID; }
       }
       public EchoService SessionEchoService   // ikke trådsikker
       {
           get { return this.echoService; }
           set { this.echoService = value; }
       }
       public ClientHandler ConnectedClientHandler
       {
           get { return this.connectedClientHandler; }
           set { this.connectedClientHandler = value; }
       }
   }
```

SocketServerCmdDisconnectedOnOff

data og metoder fra Server klassen – NB ikke trådsikrede i udgaven

```
private Dictionary<String, Session> sessioner = new Dictionary<string, Session>();
private int lastSessionId = 0;

public Session StartSession()   // ikke synkroniseret
{
    ++lastSessionId;
    Session nySession = new Session(lastSessionId.ToString());
    sessioner.Add(nySession.SessionId, nySession);

    #if DEBUG       // Hvis compilerdirektiv er sat medtages koden
    Console.WriteLine("SessionId: " + nySession.SessionId + " er nu startet");
    #endif

    return nySession;
}
public void EndSession(Session session)   // ikke synkroniseret
{
    sessioner.Remove(session.SessionId);

    #if DEBUG       // Hvis compilerdirektiv er sat medtages koden
    Console.WriteLine("SessionId: " + session.SessionId + " er nu afsluttet");
    #endif
}
public Session GetSession(string sessionId)   // ikke synkroniseret
{
    if (sessioner.ContainsKey(sessionId))
        return sessioner[sessionId];
    return null;
}
```

SocketServerCmdDisconnectedOnOff

doDialog metoden fra ClientHandler klassen

```
private void doDialog()
{
    #if TELNET_DIALOG       // Hvis compilerdirektiv er sat medtages koden
        sendToClient("Server klar"); // opstarts besked til klient
        sendToClient("Indtast SessionId eller 0 for ny"); // opstarts besked til klient
    #endif

    // Skaf session objekt
    string oldSessionId = receiveFromClient();
    session = server.GetSession(oldSessionId);      // hent evt. session-objekt
    if (session == null)
    {                                               // fantes ikke
        session = server.StartSession();            // start ny session
        session.SessionEchoService = new EchoService();
    }
    echoservice = session.SessionEchoService;       // hent stateful modelobjekt / facade

    #if TELNET_DIALOG       // Hvis compilerdirektiv er sat medtages koden
        sendToClient("SessionId:");
    #endif

    sendToClient(session.SessionId);                // Send anvendt sessionid til klient

    #if DEBUG       // Hvis compilerdirektiv er sat medtages koden
        Console.WriteLine("SessionId: " + session.SessionId + " er nu aktiv");
    #endif

    //            executeCommand(); den disconnectede tog kun en komando
    while (executeCommand());       // I denne udgave kan der modtages flere
}
```

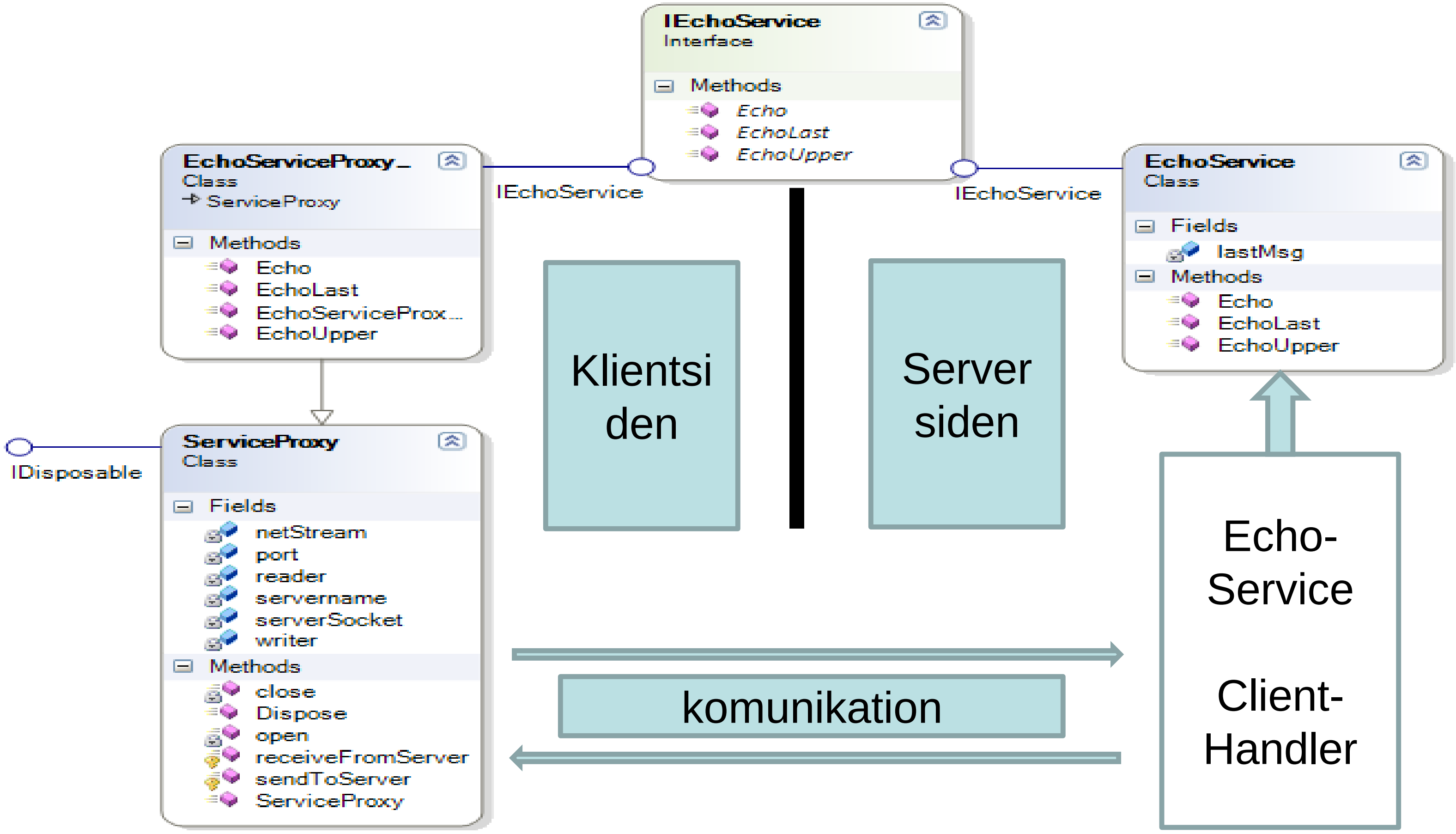
SocketServerCmdDisconnectedOnOff

uddrag fra metoden executeCommand i ClientHandler klassen

```
      public bool executeCommand()  //returner false hvis null eller bye
      {
          string command;
          command = receiveFromClient();
          if (command == null)
              return false;       // ikke mere input

          switch (command.Trim().ToLower())
          {
              case "echo":
                  {   // local scope så varable ikke ses uden for
                      string inputMsg = receiveFromClient();          // hent supplerende parametre fra klient
                      string response = echoservice.Echo(inputMsg);   // udfør metode
                      sendToClient(response);                         // send returværdi til klient
                  }
                  break;
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
              case "bye":
                  server.EndSession(session);         // Session nedlægges
                  return false;
              case "pause":                           // Her afbrydes uden at session nedlægges
                  return false;
              default:
                  sendToClient("Ukendt komando");
                  break;
          }
          return true;
      }
```

SocketEchoServiceLibrary

Klasser for service og remote tilgang tilrettet til proxy-mønster



Klientsiden

Serversiden

komunikation

Echo-Service

Client-Handler