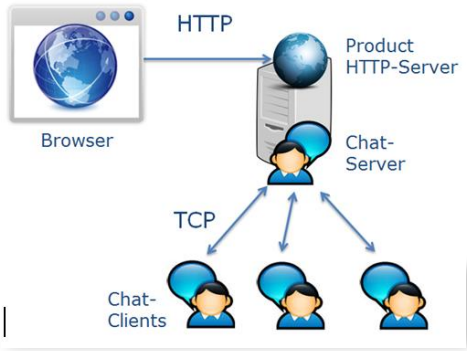
CA-1 Chat-Server/Client + supplementary Web Server

For this CA you must write a simple Chat-server and a corresponding client. The system must also provide a simple Web server which should provide users with the ability to download the chat-client, provide administrator information about the Chat system, and hold links to all required documentation for the project (see http://plaul3.cloudapp.net:9090/ for an example)



The Chat Server + Client

The server and the client must be written in Java.

It is expected that other companies will start to use this chat system and write their own clients (and servers) in languages/platforms other than Java[1], so all implementations must obey the protocol given below.

Commands from clients to the server		
Command	**Content**	**Example(s)**
CONNECT	Name of the client (it is initially assumed that all users have a unique name)	CONNECT#LarsM
	The CONNECT can only be followed by an ONLINE command from the server	
SEND	Receiver followed by the message. (receiver must be a name received via an ONLINE command)	SEND#Peter#Hello Peter SEND#Peter,Hans#Hello Peter and Hans SEND#*#Hello everybody
	After a SEND command, the server can send an ONLINE, MESSAGE or a CLOSE command	
CLOSE	Nothing	CLOSE#
	After having sent a CLOSE command the client should discard all messages received until the server responds with a corresponding CLOSE command on which the client should close the connection.	

Commands from Server to client(s)		
Command	**Content**	**Example**
ONLINE	Name of all clients, currently online. • The server should send this message to all clients, each time a client has connected or disconnected) • The list should include the name of the specific client that receives this message	ONLINE#LarsM,Peter,Hans
	The client can use this command to build a list of all On-line chatters. This command is sent each time a new client has connected or a client has left the chat.	
MESSAGE	Sender followed by the message	MESSAGE#Peter#This is the message being sent
	This Command is sent to a client whenever a message is received for the client (the server has received a SEND command with this name or "*")	
CLOSE	Nothing	CLOSE#
	Having sent a CLOSE command the server should close the connection and release all resources attached to that client.	

Every message is sent as a string, including, both the command and content, so all the examples above illustrates real messages which could be monitored via Wireshark.

[1] You will actually have to do this later this semester

Requirements:

- The protocol must follow the rules given above and be 100% unit tested.
- The server must include a log-file to hold important run-time information
- The system must include a simple Swing-client

The Web Server

The projects web server must be implemented in Java as a simple HTTP file server that can serve mostly static files. Is must however also be able to create some simple dynamic pages, i.e. a page that can show the current number of on-line users and a page that can show the log file for the chat server (see http://plaul3.cloudapp.net:9090/online.html for an example).

Requirements

- The server must be able to serve static pages (.html, *.pdf and java jar-files)
- The server must be able to create simple dynamic pages (pages built on the server before they are returned) as sketched below:
  - A page that show the current number of online users
  - A page that shows the current chat log information.
- The server must include a main-page, two pages for the dynamic content, a link to your client (as a jar), a link to your documentation and a link to your source code on Github.
- The server must include a log-file to hold important run-time information

Study-points for this Period and the CA

You can earn a maximum of **30 study points** in this period.

To be approved for the examination you must score at least 75% of all available points during the semester.

Points are given as sketched below:

For your participation in the class (one for each day)	15 points
Each member in a team can earn additional 15 points for the CA as sketched below:	
For your contribution to the code and documentation (verified via Git + your documentation must include a note on who did what)	Up to 5 points
The quality of your design, test strategy/coverage and documentation	Up to 5 points
How far you came with the actual exercise	Up to 5 points

Demonstration and what to hand in for the Chat System

- The server must be demonstrated up against your own client + a client from at least of the other teams
- The client must be demonstrated up against your own server, + a server from at least one other team.
- The code must be made available via GIThub.
- A short description including:
  - A short design description of the chosen design.
  - A section stating who did what
  - A short description of how you have implemented the state behaviour
  - A Wireshark sample for a complete TCP scenario between the server and a single client (from start to end) and a description which (as a minimum) must include:
    1) A flow Graph diagram showing both the initial three way handshake and the close down handshake.
    2) A description of the first 6 messages involved in the flow diagram above
- The Compiled Chat client, all documentation and the required admin info must be available via a small web-site published on your web-server.
- The project must be made available to everyone via your group account on Azure

Note: The Code for this CA will form the background for one or more examination questions