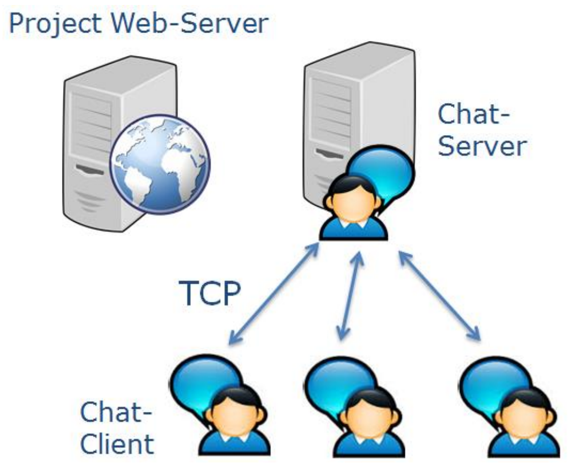
CA-1 Chat-Server/Client + Project Web Server

For this CA you must write a simple TCP-Chat-server and a corresponding client.

The CA must also provide a simple Web Site which should provide users with the ability to download the chat-client (as a jar-file) and include all required documentation for the project + a link to the Git-hub account that includes the source



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, so all implementations must obey a common application Layer Protocol running on top of TCP.

The protocol

We will design this protocol together Monday (September 5th). After that the protocol description will be available on Fronter in the folder CAs. Note: this description is THE MOST IMORTANT part of the requirements. If your server does not obey this protocol, it will not be accepted as a valid solution. Server and client <u>must not send anything else</u> than what is described in the protocol.

Requirements:

1. Your implementation must follow the rules given in the protocol designed in the step above.
2. You need to test your application (see notes)
3. The Server must include a log file where all Exceptions and relevant Events are logged
4. The system must include a reusable non-GUI client, offering a non-blocking way to listen for incoming messages.
5. The system must include a simple Swing-client which should use the client implemented in step-2.
6. The server must be deployed on, and made available from *DigitalOcean*
7. You must demonstrate your client up against a server from at least one other group
8. You must demonstrate your server up against at least two clients, from another group.

Test requirements:

You need to make automated tests for the protocol from server and client side.

In your Server project (if you have split server and client), spin up the server in a new thread, and test it through a dummy client. An example for setup and teardown for a test suite is listed below:

```
@BeforeClass
  public static void setUpClass() {
    new Thread(new Runnable(){
      @Override
      public void run() {
        Server.main(null);
      }
    }).start();
  }
  @AfterClass
  public static void tearDownClass() {
    Server.stopServer();
  }
```

The Web Server

The projects web server must be implemented as a simple Tomcat/JSP-based web application and must as a minimum, include the Following:

Requirements

- The server must include a main-page with a link to your client (as a jar) and a link to a second page, containing all documentation for the project (including a link to your source code on Github).
- The page containing the documentation and your link to Git-hub must be password protected, using either basic HTTP Authentication or Form Based Authentication.
- The server must be deployed on, and made available from *DigitalOcean*

How to spend your four days

You are supposed to develop the system over 2 consecutive two-day mini sprints as sketched below:

- Day-1+2: Before the end of day two you must demonstrate your server, serving at least two clients (use Telnet until you have implemented your Java Client)
- Day 3+4: Complete and deploy the system

Study-points for this Period and the CA

You can earn a maximum of **40 study points** in this period.

To be approved for the examination you must score at least 80% of all available points during the semester.

Points are given as sketched below:

For your participation in the class (one for each day)	11 points
For the two Friday Study Point Exercises	Up to 10 points
Each member in a group can earn additional 19 points for the **CA** as sketched below:	
For your contribution to the code and documentation (verified via Git-hub + your documentation must include a note on who did what)	Up to 5 points
For the demonstration of the server Day-2	Up to 3 points
The quality of your code and coverage of **test cases**	Up to 4 points
For the demonstration of the Final System (it must obey the protocol 100% to get any points for this part) and the quality of the documentation	Up to 4 points
For giving us constructive feedback about module-1	Up to 3 point

Bonus Points

While you develop this system you will acquire knowledge about many small details, which eventually might took you a lot of time to figure out.

You should try to help each other, so if you have a problem, ask out in the class if anyone can help you[1] (and respect if the reply is – wait we are doing something else right now).

You should generally use the Q&A site for questions. Contribution on the QA-site and answering questions in the class, can earn you up to **5 EXTRA study 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 one other team.
- The client must be demonstrated up against your own server, + a server from at least one other team.
- Your opinions about good and bad things about how we planned and taught module-1 (we need this to improve future versions and perhaps even the coming modules)
- The code must be made available via GIT-hub.
- A short description including:
  - A short design description of the chosen design.
  - A section stating who did what
  - An image of the test results from JUnit

- The Compiled Chat client (as a jar), all documentation and a link to Git-hub, must be available via a small web-site published on your web-server.
- Both Servers must be deployed on an DigitalOcean Virtual Machine (Droplet)

How to hand in: Upload a single link to Fronter containing ONLY the link for your projects web site.

When to hand in: No later than Sunday, September 11th 14.00.

[1] Obviously, teachers and/or the tutors will also be available