# Lab 1: A simple chat program

**Prelude**

Before you start, make sure that you have read the general instructions and are following them on every lab.

**Overview**

You shall write a socket based chat server and client using the java classes `java.net.Socket` and `java.net.ServerSocket`. The client should connect to the server through the Internet and this must work by two clients running locally on same computer as the server and one client connect remotely running on other computer.

**Specific Requirements**

1. The implementation shall be written in Java.

2. The implementation must be able to handle multiple events in a non-blocking manner. Specifically, this means that incoming and outgoing requests on both the client and server must be able to occur simultaneously. You must use threading on both the server and client.

3. The server must be able to handle multiple clients simultaneously without risking losing a message.

4. You must make sure that the order of the messages is preserved for all messages.

5. To know who writes what, everyone needs a username/alias that will be printed when they send a message. It is a task of the server to associate the name of the Client who sends the message in every message.

6. It must be able for clients to send/receive private messages to/from each other.

**Tips:**

- On the server you will need to have a list of some sort (e.g. `Arraylist`) to save the sockets.
- To avoid having to use multiple physical machines, use SSH (secure shell), by for example SSH:ing to `myUserName@u-shell.csc.kth.se` and starting a client from there.

**References:**

For more information on how to create and manage threads:

http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

For more basic information on how basic IO works in java there is always:

http://docs.oracle.com/javase/tutorial/essential/io/index.html

**Questions:**

1. If you have n connected clients, how many instances of thread are needed by the:
    - Server?
    - Client?
2. What does the Java keyword *`synchronized`* do?
3. What is a runnable in Java?
4. Describe the four layers in the TCP/IP protocol stack.
5. What does the flags, ACK, SYN and SEQ mean and what protocol do they belong to?
6. What is the difference between TCP and UDP?

**Bonus assignment X1**

Provide your application with the feature that clients can send arbitrary binary files e.g. image files to each other. It is important that the file uploads directly by the client and also do not pass through the server. The procedure will be as following:

Assume that a client, A, likes to send a file to another client, B.

1. Client A sends a request to inform the client B about the file.
2. In this moment the client B can accept or reject the request.
3. In case of rejecting, the client B sends a rejecting response to the client A.
4. In case of accepting, the client B sends an accepting response to the client A.
5. The client A initiate a `ServerSocket` and invokes the `accept` method.
6. Client A sends address information of the awaited `ServerSocket` object to the client B.
7. In this moment the client B starts downloading the file ***directly*** (without involvement of the server) from the client A.

Questions:

What is the advantage of this procedure for transferring files compared with other procedure like there the client B starts a `ServerSocket` for aim of uploading files?