You will develop a more realistic messaging system.

**Procedure**

Start with the lecture code for the messaging system.

You can choose either

the provided sample solution with quit, or

your own solution for quit, if it works properly,

your own implementation from scratch, if you are brave and confident, provided it exhibits the same behaviour.

The compilation procedure should be the same as for the quit exercise:

$ java Server

and

$ java Client <server machine>

That is, a user name is no longer provided.

Include a SOLUTION.md file explaining your approach.

If you choose to implement extra features, explain both their syntax (how to use) and their semantics (intended behaviour) in SOLUTION.md.

Include concise comments in your solution.

Use good code formatting (more information will be provided by Kelsey McKenna).

**Summary**

Extend the system with the following features:

1.Register.

2.Login.

3.Logout.

4.Keep all messages received by any user.

5.New syntax and semantics to send messages.

6.Allow user to move from current message to previous or next, or delete current message.

7.Self-chosen feature to get above 75%.

**Details**

**Register**

The user types



If Helen exists, and error message should be returned but the system should continue to operate normally.

If not, the server adds Helen to a suitable table with corresponding information.

No passwords are required.

You can choose whether Helen is logged in automatically or not after registering, but you should explain your choice in your SOLUTION.md file.

**Login**

The user types



Gives an error if Helen is not a register user. If successful, Helen gets to see the current message, if any.

Should the same user be allowed to login twice at the same time in different machines?

Decide what to do and explain in SOLUTION.md.

Allowing multiple simultaneous logins, properly implemented, gives you more marks.

But if you are not handling this properly, then you should not allow it, with an error message to the second login attempt.

**Logout**

The server stops sending messages to this client, but keeps storing

them in their list. The threads for serving this client should

terminate, without triggering errors.

**Keep all messages received by any user**

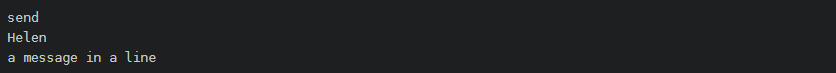
All messages for any user are kept in the server.

The server knows what the current message is for each user that has at least one message.

This is the message displayed in the client when the user logs in.

**New syntax and semantics to send messages**

This should be modified to use the following syntax:



Notice that these are three lines. In the original system, the line with the word send is not required.

If Helen is not logged in, but is a valid user, the message is stored in her list of messages, which she will be able to read when she logs in.

If she is logged in, she gets that message, which becomes her current message.

**Previous, next and delete**

previous moves to the previous message, if any, and shows it.

next moves to the next message, if any, and shows it.

delete deletes the current message. It is your choice to decide which message will become the current message (the previous, if any, or the next, if any) and you should state your choice in SOLUTION.md.

**Self-chosen feature**

This is to get above 75%. Possible examples include one of the following:

Passwords, with password and message encryption.

User groups like in WhatsApp.

User table and messages stored in a file so that a server can be stopped and restarted without loss of information.

A simple graphical interface replacing the command line.

If you are in doubt about whether a feature qualifies for full marks, or at least some marks above 75%, please ask in the Facebook group, in a lecture, or in the lecturer's office hour.