CS3524, CS4595 Programming
Assessment: Messenger

In this project, you will implement a simple messenger server that allows communication between two or more communication partners.

You can use and adapt code provided in lectures and practical exercises to construct your solution to the assessment. When you do, however, you must acknowledge the classes that your solution is based on using a comment in your source code. However, you are not allowed to use existing tools like JGroups Toolkit and/or JMS.

Please note that your solution has to be executed when marked. Therefore, provide information how to install / setup and run your solution from terminal windows.

Assessment Requirements

CGS D3-D1

In order to achieve a CGS D, develop a client – server application based on socket communication:

- ✓ Create a server based on socket communication that can handle the communication between a client and a server
- ✓ Create a client based on socket communication
- ✓ A user of the client application can type messages on the command line, which are displayed by the server.
  - Each message from the client creates a response by the server (you can echo the message back to the client, maybe with a text like "[Messenger]" in front)
  - The conversation ends when the client is aborted with CTRL-C

How to start your assessment

Your task is to create a simple Messenger server, where clients can exchange messages via a server. Start the assessment with a simple client – server application based on a socket communication, where the client sends messages to the server and the server is printing these messages (regard this as a kind of log of activities and as a debugging feature on the server).

CGS C3-C1

In order to achieve a CGS C, develop a client – server application based on socket communication that can handle messaging among multiple clients:

- ✓ Create a server based on socket communication that can handle multiple clients
- ✓ Create a client based on socket communication
- ✓ A user of the client application can type messages on the command line, which is sent by the server to all the clients

Create a solution where multiple clients can connect to a server, each identifying itself through a unique identifier (a name specified by a user at the start of the client).

Any message sent by one client is forwarded to all other clients by the server.

Develop a simple command language that allows clients to register with the server. The commands should follow a simple syntax, e.g. "REGISTER myname", where the first word in a message is checked whether it is a keyword of your command language (you can use java.util.Scanner or java.util.StringTokenizer to extract words from strings).

```
<command>    ::= <keyword> | <keyword><text>
<keyword>    ::= <register> | <unregister>
<register>   ::= REGISTER <name>
<unregister> ::= UNREGISTER
<text>       ::= any text . . .
```

If the first word is not a keyword recognised by the server then this message is forwarded to all other clients by the server.

When a user quits the execution of a client application, all created input and output streams, as well as sockets should be closed correctly. It should also unregister from the server. The client should see who the sender of a message is.

CGS B3-B1

In order to achieve a CGS B, develop a client – server application based on socket communication that can handle messaging among groups of multiple clients:

- ✓ Create a server based on socket communication that can handle multiple clients
- ✓ The server can handle one-to-one communication between two clients, and user groups, where clients can join and leave these groups
  - Create a set of commands that allows a user to CREATE a group, JOIN a group conversation, LEAVE a group, and REMOVE a group
- ✓ A user of the client application can engage in conversations with either all the other users, or can send messages to a group
  - Create a command SEND <name> <text>
    - If the name is a group name, the text is forwarded by the server to all clients in this group
    - If the name is the identifier of an individual client, the text is forwarded to this client

CGS A5–A1

In order to achieve a CGS A, develop a client – server application based on socket communication that can handle messaging among groups of multiple clients and forwards messages according to clients subscribing to particular keywords in messages that represent topics of interest:

- ✓ The server can handle users subscribing to topics of interest
- ✓ Users can subscribe to particular keywords that may occur in messages and whenever a message is sent by some client containing at least one of the subscribed keywords, it will be forwarded
- ✓ Add additional commands:
  - SUBSCRIBE <keyword>
  - UNSUBSCRIBE <keyword>
  - TOPIC <keyword> … creates a new topic (server registers this keyword)
  - TOPICS … lists the current set of keywords

You can also introduce a hashtag functionality, by simply setting a hashtag in front of words in your message. This requires the server to scan each message, extract all the words with a hashtag in front and perform the same action as required for the command ‘TOPIC’.