6.005 Project 2: Concurrency and Testing Strategy

Team Members: Rebecca Zhang (ryzhang), Katie Lee (katielee), Birkan Uzun (birkanu)

TA: Joshua Oreman

Concurrency Strategy

We need to account for concurrency for when...

- Multiple clients connect concurrently.
    - each client has its own thread.
    - broadcast method is not lost.
- Multiple clients try to connect with the same username.
    - Include a synchronized method checkUsername(String username) in **Client** which checks Server's list of existing **Client** usernames to see if this username already exists.
- Multiple messages are sent to the server concurrently.
    - Each message sent to the server is handled on its own thread.
- Multiple messages are sent to the client concurrently.
    - The Broadcast message method is synchronized, so only one thread of the server can broadcast a message to any particular client at the same time.
    - We do not have to worry about broadcasting messages at the same time as entering/leaving chatroom because the server thread for a particular client (aka clientThread) is individually synchronized, so only one of those method can be accessed at a given time.

Testing Strategy

Manual Testing:

We will do manual testing for GUI components.

1. LoginWindow pops up from Client class start
    - Submit button check for valid username entry
        - no username (i.e. whitespaces only)
        - not 6-16 characters
        - repeat username
    - LoginWindow closes if valid username is entered
    - MainWindow pops up when valid username is entered

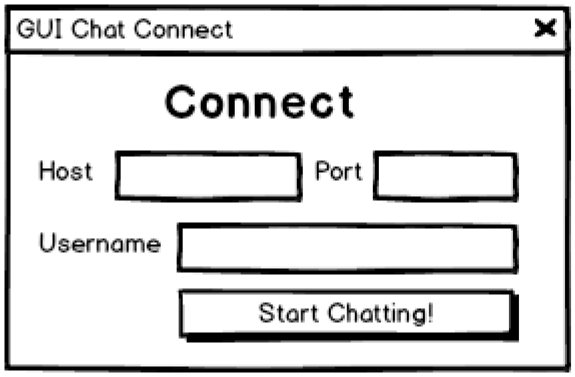
2. MainWindow:
    - Current online client list is correct
        - for other clients, new username added to their online client list
    - Message sent to other clients
        - check delay time so no lag in message delivery
    - Message received from other clients
        - no dropped messages
        - messages are in order in which they were sent
    - User leaves the chatroom
        - online client list for other clients has been updated
        - MainWindow closes if disconnected

JUnit Testing:
We will do JUnit testing on the non-GUI components (i.e. the Client class and Server class). Mainly, we will test the communication between the Client and Server classes by creating a socket and sending messages to the server directly.

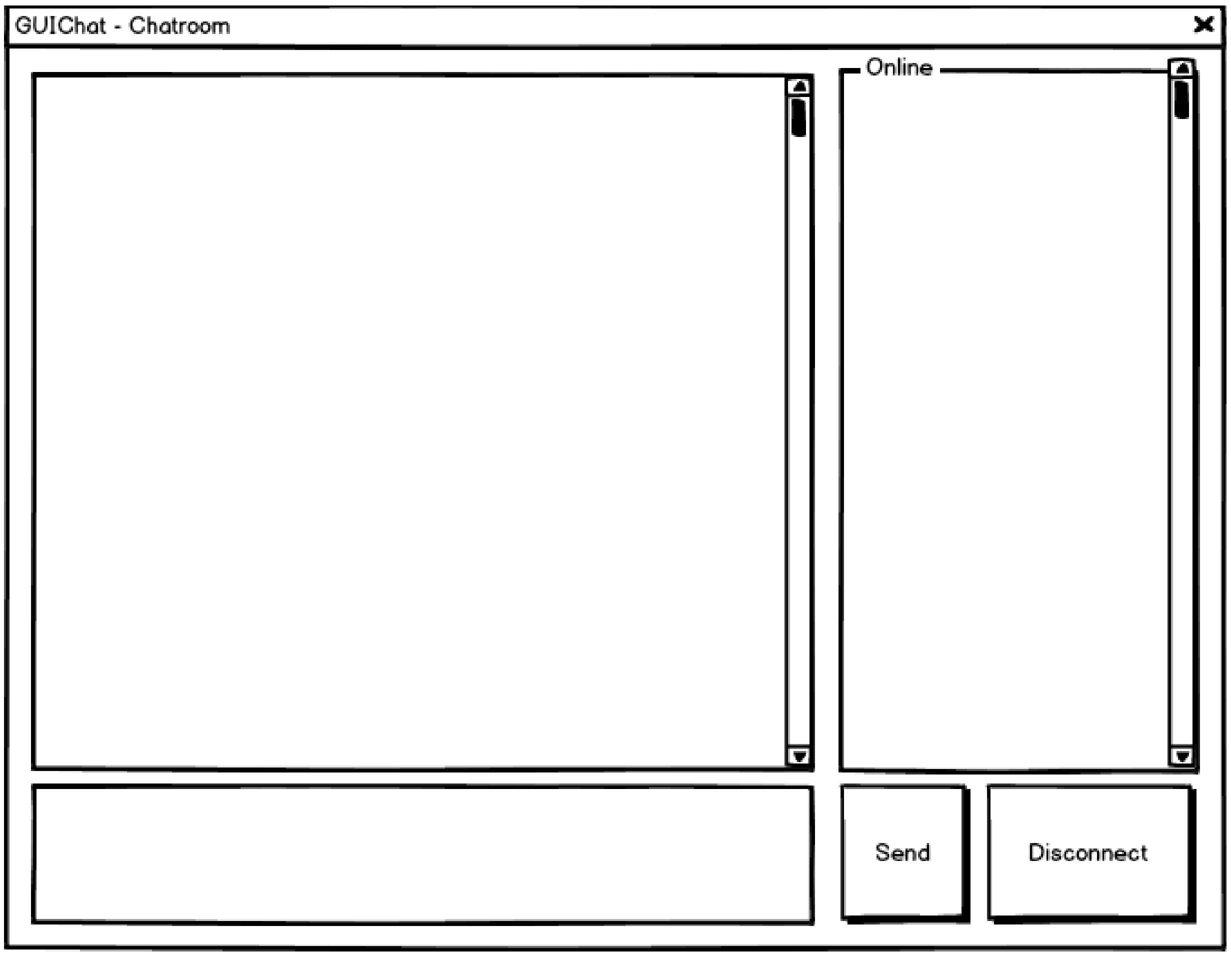
- Server receives Client message for...
    - Client connect
    - Client message
    - Client disconnect
- Server recognizes clientThreads list updates when...
    - new Client is added
    - Client is removed
- Client actively sees changes in clientThreads list when...
    - new Client is added
    - Client is removed
- Client receives Server broadcast for...
    - message
    - user login
    - user logout

UI Sketches

Initial connection window::



Main Chat Window::


GUIChat - Chatroom
Online
Send
Disconnect