Description

In our IM client, a conversation is modeled after a chat room. It is identified by a conversation number, and consists of one or more users. A conversation is started by a user, and users are added to the conversation by any user that is already in the conversation. The original starter of the conversation does not “own” the conversation, in that the starter can exit or log out of the system without the conversation disappearing. The conversation is destroyed when all users have left the conversation through some form (closing the window, logging out, closing the IM client, etc.)

The Server contains a list of users and a list of rooms. Each time a client logs in, a User object is created, and that User object is added to the user list. If the client wants to create a room, the User object will tell the Server to create a Room. When a User wants to invite another User to the Room, a notification will be sent from one User to the other allowing him to accept or deny. If the User denies, the sender will be notified. Each time a client sends a Line to its corresponding User, the User sends this Line to the Room which adds it to its Tablet. Then, each room has a list of Listener objects subscribed to the room which updates what the client can see.

Extensions

In our IM client, we implemented the following extensions beyond the basic requirements

- Password authentication.
- Used SHA256 to hash password before stored
- Encryption for all messages across socket/client
- User/password registration and storage
- Server address prompt upon connection
- Emoticons, bold, strikethrough, and underline commands in the chatroom
- Window name changes to "New Message..." when room is out of focus

Testing Strategy, Concurrency/Threadsafe arguments are at the top of all relevant classes
Method specs are Javadoc'd at the beginning of methods

Client-Server Protocol

Server → client

Protocol ::= Message*

Message ::= NotifyContactOffline | NotifyContactOnline | NotifyRoomInvite | NotifyNewMsg |
NotifyRoomCreated | NotifyError | NotifyStatus | NotifyRoomUsers | NotifyEnter |
NotifyLeave | NotifyWelcome | NotifyTyping | NotifyEnteredText | NotifyIdle |
NotifyRegisterSuccess | NotifyOnline

NotifyContactOffline ::= ContactOffline Username Newline
NotifyContactOnline ::= ContactOnline Username Newline
NotifyRoomInvite ::= RoomInvite RoomNumber Username Newline
NotifyNewMsg ::= NewMsg Username RoomNumber Line Newline
NotifyRoomCreated ::= RoomCreated RoomNumber Newline
NotifyError ::= Error ErrorMsg Newline
NotifyStatus ::= Status OnlineStatus (Number)* Newline
NotifyRoomUsers ::= RoomUsers RoomNumber (Username TypeStatus)+ Newline
NotifyEnter ::= Enter Username RoomNumber Newline
NotifyLeave ::= Leave Username RoomNumber Newline
NotifyWelcome ::= Welcome Username Newline
NotifyTyping ::= Typing Username RoomNumber Newline
NotifyEnteredText ::= EnteredText Username RoomNumber Newline
NotifyIdle ::= Idle Username RoomNumber Newline
NotifyRegisterSuccess ::= RegisterSuccess Username Newline
NotifyOnline ::= Online (Username)+

RoomNumber ::= Number
ErrorMsg ::= error 0 malformed command |
error 1 command not found |
error 2 user not online |
error 3 room does not exist or you are not in it |
error 4 user with same name is already logged in |
error 5 bad username |
error 6 contact is not online |
error 7 user already online on same client |
error 8 user with same username already exists |
error 9 invalid login or password

ContactOffline ::= offline
ContactOnline ::= userOnline
RoomInvite ::= invite
NewMsg ::= message
RoomCreated ::= roomcreated
Error ::= error
Username ::= [A-Za-z0-9_-]+
OnlineStatus ::= (online | offline)
RoomUsers ::= roomUsers
Enter ::= enter
Leave ::= leave
Welcome ::= welcome
Typing ::= typing

EnteredText ::= enteredText
Idle ::= idle
RegisterSuccess ::= registerSuccess
Online ::= online
Number ::= (1-9)[0-9]*
Line ::= .*
Newline ::= \n

Client → server

Protocol ::= Message*
Message ::= NotifyLogin | NotifyCreateRoom | NotifyInvite | NotifyAccept | NotifyDecline | NotifySay |
NotifyLogout | NotifyCreateAcct | NotifyStatus | NotifyOnline | NotifyLeave | NotifyEnteredText |
NotifyIdle | NotifyTyping | NotifyRoomUsers

NotifyLogin ::= Login Username Newline
NotifyCreateRoom ::= CreateRoom Newline
NotifyInvite ::= Invite Username RoomNumber Newline
NotifyAccept ::= Accept RoomNumber Newline
NotifyDecline ::= Decline RoomNumber Newline
NotifySay ::= Say RoomNumber Line Newline
NotifyLogout ::= Logout Newline
NotifyCreateAcct ::= Register Username Line Newline
NotifyStatus ::= Status Newline
NotifyOnline ::= Online Newline
NotifyLeave ::= Leave RoomNumber Newline
NotifyEnteredText ::= EnteredText RoomNumber Newline
NotifyIdle ::= Idle RoomNumber Newline
NotifyTyping ::= Typing RoomNumber Newline
NotifyRoomUsers ::= RoomUsers RoomNumber Newline

Login ::= login
CreateRoom ::= create
Invite ::= invite
Accept ::= accept
Decline ::= decline
Say ::= say
Logout ::= logout
Register ::= register
Status ::= status
Online ::= online
Leave ::= leave
EnteredText ::= enteredText
Idle ::= idle
Typing ::= typing
RoomUsers ::= roomUsers
Username ::= [A-Za-z0-9_-]+
RoomNumber ::= (1-9)[0-9]+
Line ::= .*
Newline ::= \n

Automated Testing:
User

CreateRoomTest:

- **Test:** Create a single room
  - **Result:** A room is created
- **Test:** Create multiple rooms
  - **Result:** Multiple rooms are created
- **Test:** Invalid Grammar
  - **Result:** Malformed error command

InviteTest

- **Test:** invite a single user to a room.
  - **Result:** the user receiving the invite message and joining the room.
- **Test** invite a single user and leave room.
  - **Result:** the user still being able to join.
- **Test** multiple invites, all accept.
  - **Result:** all can all join
- **Test** multiple invites, all decline.
  - **Result:** we do not see any users join.
- **Test** multiple invites, some accept.
  - **Result**: only some join.
- **Test** accepting an invite to a no longer existing room.
  - **Result:** Room no longer exists error message
- **Test** declining an invite to a no longer existing room.
  - **Result:** Makes sure there is no error message
- **Test** sending accept message to a nonexistent invite.
  - **Result:** Error message
- **Test** sending decline message to a nonexistent invite.
  - **Result:** Make sure there is no error message
- **Test** invite multiple times to the same room
  - **Result:** only receive one invite message.
- **Test** invite same user to multiple rooms
  - **Result:** receive all the messages.
- **Test** invite a nonexistent user
  - **Result:** test for error message
- **Test** invite, accept and decline messages that do not conform to the grammar.
  - **Result:** Test for malformed command error.

LeaveRoomTest

- **Test**: Single person in the room.
  - **Result:** Upon leaving the room, the room dies. Subsequent creation of rooms increases the room number.
- **Test:** Multiple people in the room.
  - **Result:** When one user (the one who started the room) leaves, the remaining users get the leave message from the server
- **Test:** Multiple people in the room. The creator of the room invites someone and leaves before the invite is received.

    - **Result:** Checks to make sure that the invite can still be accepted, and that the people left in the room receive both the leave messages and the new enter message from the newcomer.

LoginTest

- **Test:** A simple login of a single user
    - **Result**: Get welcome message back.
- **Test:** Other users present.
    - **Result:** Get welcome message back.
- **Test:** Login with already taken name.
    - **Result:** Get error message
- **Test:** Try to send login command while already logged in.
    - **Result:** Get error message
- **Test:** Login, logout, then login again.
    - **Result:** Get two welcome messages.

LogoutTest

- **Test:** Logout having done nothing to rooms.
    - **Result:** Checks status for offline.
- **Test:** Say to room after logout.
    - **Result:** Check for error messages.
- **Test:** Someone else say to room after logout.
    - **Result:** Checks that you do not receive it.
- **Test:** Invite a logged out user,
    - **Result:** Checks for appropriate error.
- **Test:** Accept an invite after logging out
    - **Result:** Checks for appropriate error
- **Test:** Create a room after logging out
    - **Result:** Checks for appropriate error
- **Test:** Attempt inviting others after logging out
    - **Result:** Checks for appropriate error.

SayTest

- **Test:** Say something in a room with no one else.
    - **Result:** Get the message back
- **Test:** Say something in a room with others.
    - **Result:** Everyone gets the message.
- **Test:** Say something in a room before and after an invite is accepted.
    - **Result:** The message is only received after the invite is accepted.
- **Test:** Say a message with a newline in the middle of the string (note that this is not possible in the GUI)
    - **Result:** malformed command error (the command will end at the newline, and the next command picks up from there)
- **Test:** Say a message to a room that you are not currently in.
    - **Result:** correct error message.
- **Test:** Say a message to a room that you've been invited to but not accepted.
    - **Result:** correct error message.
- **Test:** Say a message to a room that you've declined.
    - **Result:** correct error message

Server

ServerRoom

- **Test:** to make sure a room can be created with proper ID
- **Test:** simple invite
    - **Result:** Make sure guest is on guest list, uninvited guests are not
- **Test:** accept invite
    - **Result:** Make sure join is successful, guest is removed from guest list
- **Test:** join without invite
    - **Result:** Make sure join is unsuccessful
- **Test:** decline invite
    - **Result:** Make sure removed from guest list
- **Test:** decline then accept
    - **Result:** Make sure removed from guest list, and the join is unsuccessful

Client Testing Strategy

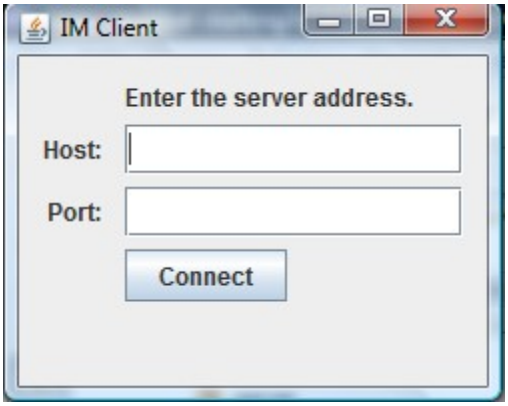
test all methods that send commands to the server and make sure they follow proper protocol, methods incude the following...

```
logout()
inviteClient(username)
changeTypingStatus(roomno,newstatus)
createRoom()
createAndInvite(username)
register(username,password)
say(roomno,msg)
acceptInvite(roomno)
declineInvite(roomno)
```

Hand Testing:

Connect Page

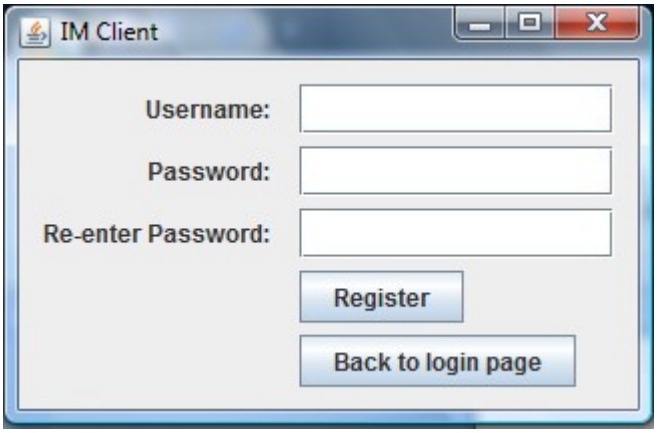
- **Test:** X button.
  - **Result**: Terminates the client.
- **Test:** Connect empty (host/port).
  - **Result**: Connection failure error.
- **Test:** Type large number into port (out of the range) and connect.
  - **Result**: Connection failure error.
- **Test:** Type proper (host/port).
  - **Result**: Connect Button should be disabled. Registration dialog. Switch to LoginView
- **Test:** Press enter in either text field instead of clicking button.
  - **Result**: Same as clicking connect button



Register page

- **Test:** Empty User/pass/re-enter.
  - **Result**: Username can only contain letters and numbers error.
- **Test:** Empty pass/re-enter.
  - **Result**: Password must be alphanumeric error.
- **Test:** Enter username non-alphanumeric(including space).
  - **Result**: Username can only contain letters and numbers error.
- **Test:** Enter password non-alphanumeric(including space).
  - **Result**: Password must be alphanumeric error.
- **Test:** Enter proper user/password; click register.
  - **Result**: Successful Registration.
- **Test:** Enter proper user/password; click back.
  - **Result**: Login page without registered username.
- **Test:** Enter mismatched password/re-enter; click register.
  - **Result**: Password must match error.
- **Test:** Click register, type in the fields, click back, click register again, and register.
  - **Result**: Successful Registration Dialog
- **Test:** Register a username that is already registered with different password.
  - **Result**: Username is already taken Dialog
- **Test:** Register a username that is already registered with the correct password.

    - **Result**: Username is already taken Dialog.
- **Test:** Press enter in any of the text fields.
    - **Result**: Same as pressing Register button



Login page

- **Test:** Click login without anything typed in.
    - **Result**: Username or password is incorrect error.
- **Test:** Click login with illegal characters(symbols, \n, \r, ^, & etc).
    - **Result**: Username can only contain letters and numbers.
- **Test:** X out login button
    - **Result**: Terminate client.
- **Test:** Login, logout, log back in.
    - **Result**: Successful login.
- **Test:** Login a user that is already logged in.
    - **Result**: User already logged in error.
- **Test:** Press enter instead of login button
    - **Result**: Same as pressing login

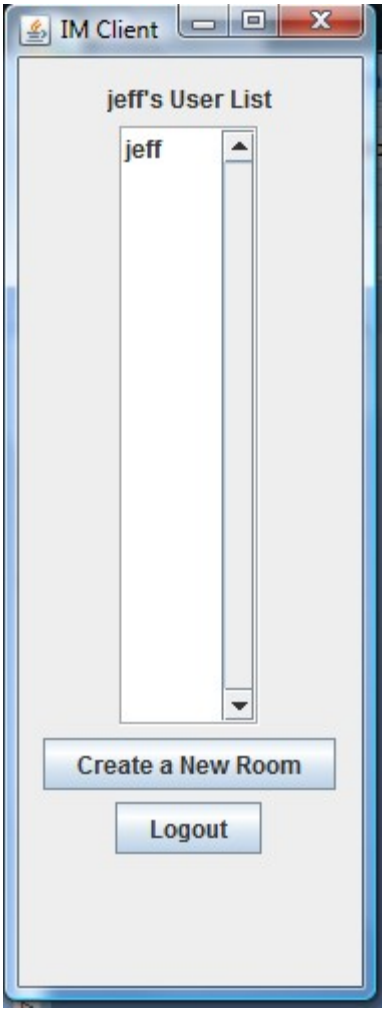
Userfile

- **Test:** Register a username/pass combination.
    - **Result**: userfile to make sure the password is hashed and cannot be hacked.

Buddy List Page

- **Test:** Click create new room with no users selected.
    - **Result**: Empty room created.

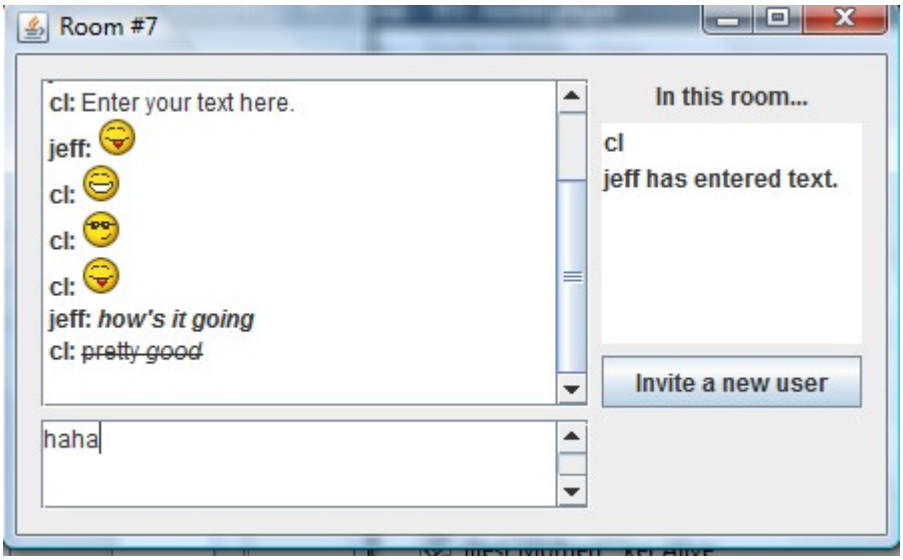
- **Test:** Click create with one person selected.
    - **Result**: Empty room created
- **Test:** Click create with multiple people selected.
    - **Result**: Empty room created.
- **Test:** Double click on contact.
    - **Result**: Create new room and invites them to it.
- **Test:** Double click with multiple contacts selected.
    - **Result**: Switches to single contact select and create/invites them to room.
- **Test:** Double click on own username.
    - **Result**: Nothing happens.
- **Test:** Logout with no rooms open.
    - **Result**: Reverts to login view with previous user/pass info
- **Test:** Logout with rooms open.
    - **Result**: Leaves and closes all open rooms and reverts to login view with previous user/pass info
- **Test:** X out buddy list.
    - **Result**: Leaves rooms, logout, and terminates client.



Room Window

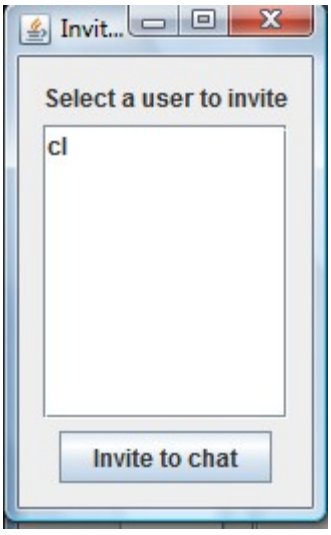
- **Test:** Everything sent between the client and server should be encrypted.
- **Test:** Press enter without any text
    - **Result**: Nothing happens.
- **Test:** Press enter with only spaces or tabs.
    - **Result**: Nothing happens.
- **Test:** Press enter with escape character (\n, \r etc.)
    - **Result**: “\n” or “\r” is printed literally not the escape character
- **Test:** Type text before and after a new user has joined
    - **Result**: The new user can only see the username: text typed after he joins
- **Test:** Type text with no other users in the room
    - **Result**: Username:text shows up in the user’s tablet.
- **Test:** Type text with multiple users in the room
    - **Result**: All the users in the room receive the username: text.
- **Test:** Type text after user is invited, before they decline
    - **Result**: The user does not see anything.
- **Test:** Type text after user is invited, before they accept
    - **Result**: The user does not see anything sent before his accept.
- **Test:** Type long text. Make sure scroll bar works.
    - The scroll bar scrolls to the bottom as the length of the text lengthens
- **Test:** Invite user with no one selected
    - **Result**: Nothing happens
- **Test:** Invite user with multiple users selected
    - **Result**: Cannot select multiple users while inviting.
- **Test:** Invite user with one user selected
    - **Result**: Invites that user.
- **Test:** Invite user already invited for a room
    - **Result**: That user is not listed.
- **Test:** Invite user before the list is refreshed after they have logged off
    - **Result**: User is no longer online dialog.
- **Test:** Invite user, leave the room before they accept.
    - **Result**: Room does not exist dialog.
- **Test:** Invite user already declined from a room.
    - **Result**: User gets invited again.
- **Test:** Invite user who already has invite for a different room.
    - **Result**: User gets invited to this specific room.
- **Test:** X out room windows, then get a message from that room.
    - **Result**: Leave room, no message received
- **Test:** Type commands {:), :(, 8), ;), :P, :D, :O } if its separated by spaces or end of line.
    - **Result**: Emoticons should replace the command.
- **Test:** Type command _text_ , -text-, *text* delimited by space or end of line
    - **Result**: The text will be underscore, strikethrough, and bold respectively. Combinations are applicable; some interweaving of different tags does not **Result** in bold/strikethrough/underline text.

- **Test:** If the user has just pressed enter to text or at the start of him joining the room
    - **Result**: The user and all other users in the room will see him in the idle state, so only his name will appear in the “In this room…” list.
- **Test:** If the user is typing text
    - **Result**: The user and all other users in the room will see “[user] is typing…”
- **Test:** If the user has typed text but it has been 3 seconds since he last pressed a key
    - **Result**: The user and all other users in the room will see “[user] has entered text



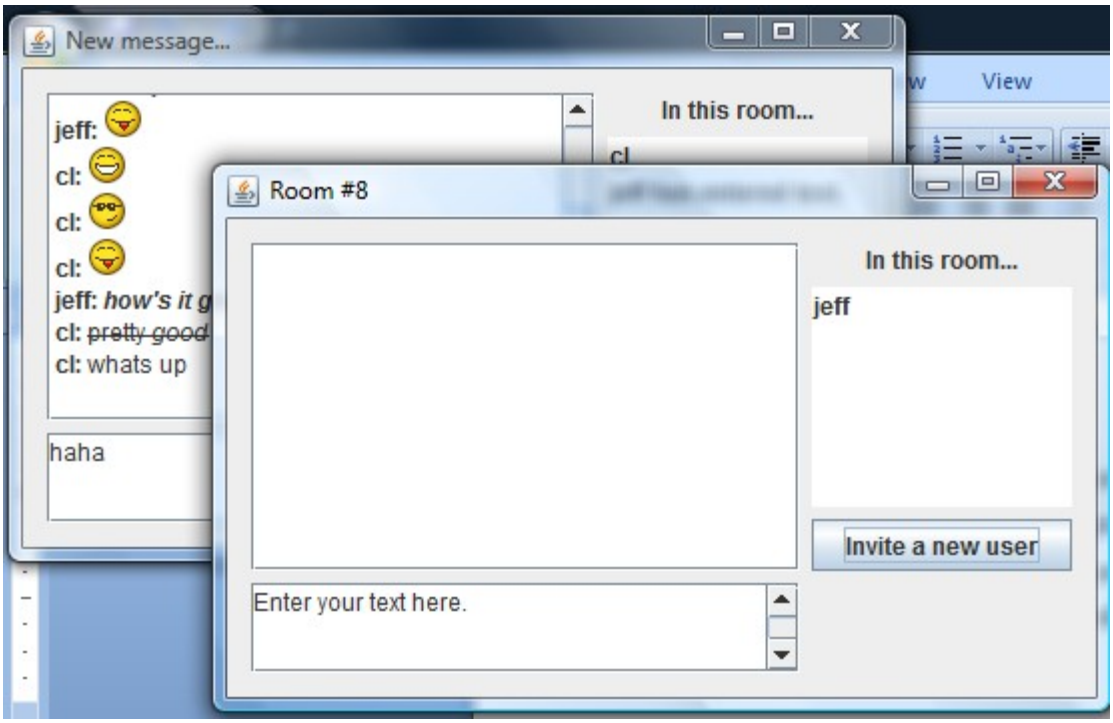
Invite Window

- **Test:** Click inviteUser to create invite window, X out room
    - **Result**: Both the invite window and the room window close.
- **Test:** Click inviteUser, X out List.
    - **Result**: No one gets invited. Room goes back into focus.
- **Test:** Click inviteUser.
    - **Result**: Invite window pops up.
- **Test:** Click inviteUser. Invite a contact.
    - **Result**: Invite window stays open. The contact is removed from the list.



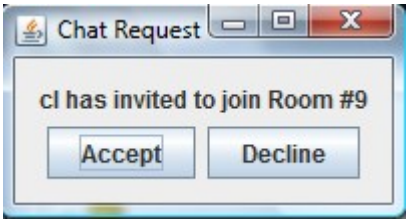
Multiple Room Window

- **Test:** Get a message from a room that is out of focus.
    - **Result**: Message appears in the tablet. “New message…” becomes the new window title.
- **Test:** Get a message from a room out of focus. Click on the room.
    - **Result**: The window title reverts back to the Room# format

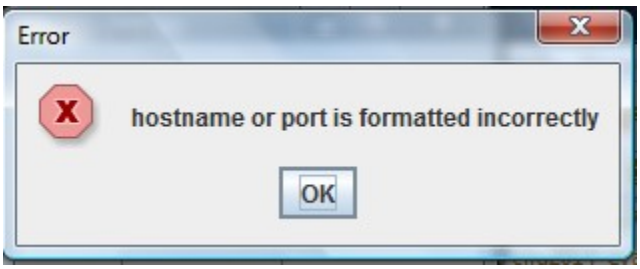


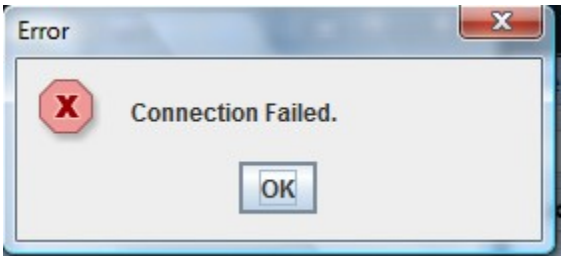
Accept/Decline User

- **Test:** Two invites to the same room, different users
    - **Result**: Should only get first invite.
- **Test:** Two invites to different rooms, same user, accept/decline combinations
    - **Result**: Gets both invites.
- **Test:** X out invite.
- **Test:** Get invite, X out buddyList
    - **Result**: Logged out all windows close.
- **Test:** X out invite, then get re-invited by same and different users
    - **Result**: Second invite is sent.
- **Test:** X out invite, then get different
    - **Result**: Second invite is sent.

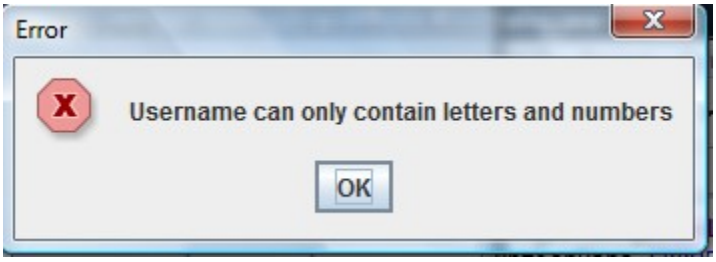


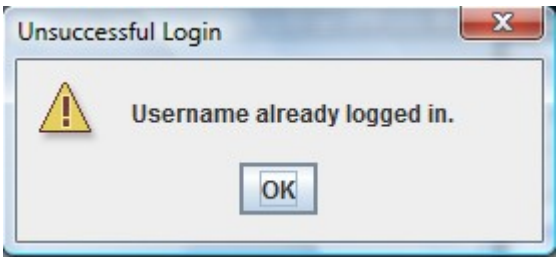
Dialog Boxes:

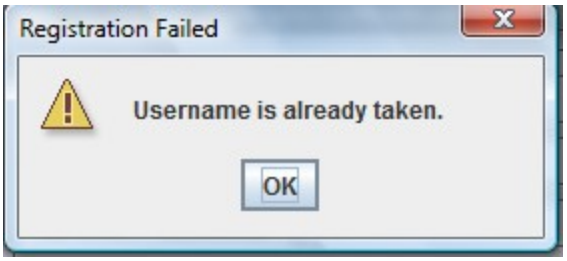

Error
hostname or port is formatted incorrectly
OK

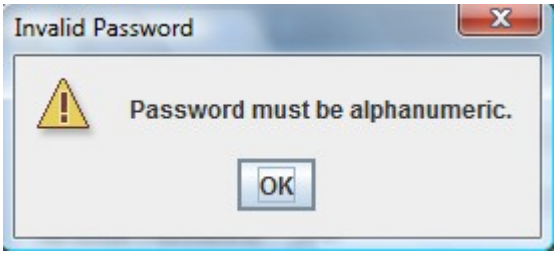

Error
Connection Failed.
OK


Invalid Login
Username or Password is incorrect.
OK


Error
Username can only contain letters and numbers
OK


Unsuccessful Login
Username already logged in.
OK


Registration Failed
Username is already taken.
OK


Invalid Password
Password must be alphanumeric.
OK

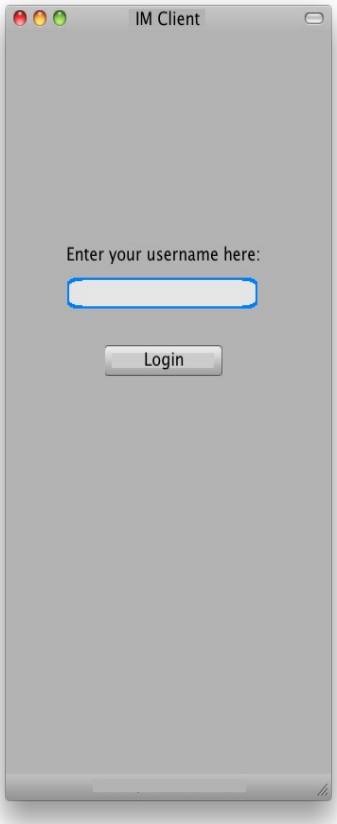

Invalid Password
Password does not match the confirm password.
OK

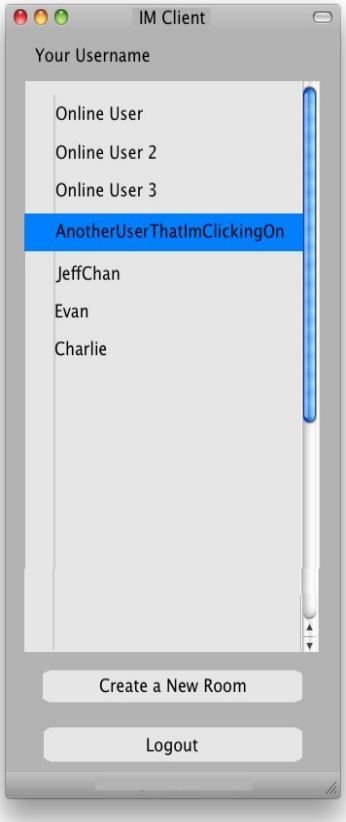

Error
Room does not exist.
OK

6.005 Project 2
Jeffrey Chan, Charles Liu, Evan Thomas

Client GUI Description

When the program is started, the user is presented with the login window (Figure 1) to type in their username. Hitting the enter key or clicking the login button will connect them to the server.

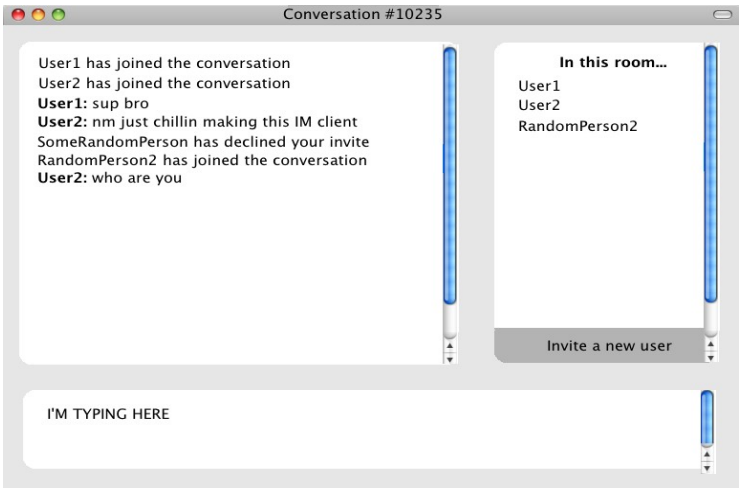




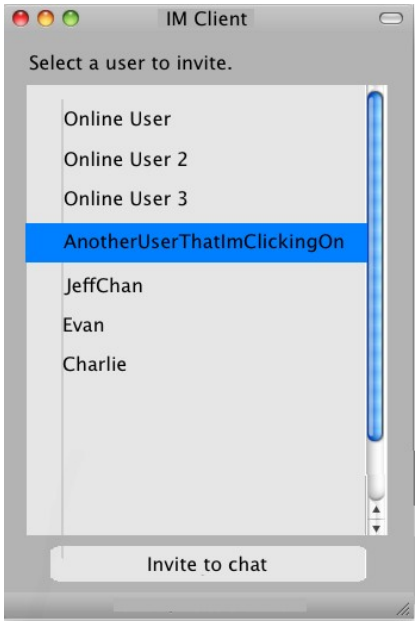
The next screen will be the “buddy list”, a list of all users that are online at the time. There are also two buttons on the bottom. The “Create a New Room” button will create a new chat room with only the current user in it. The “Logout” button will disconnect the user from the server, close all current chat windows, and display only the login window.

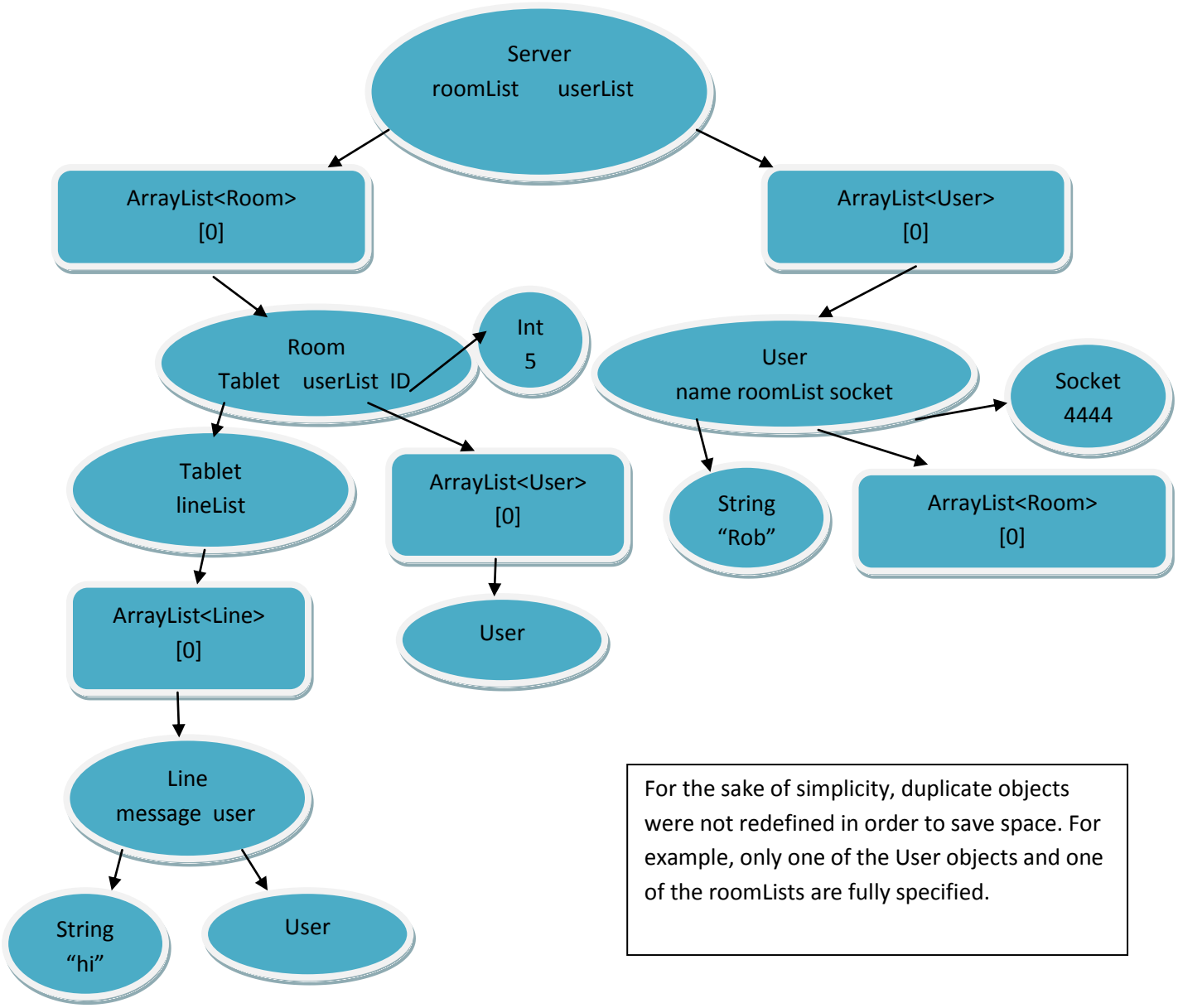
Single clicking a user will highlight the user in the buddy list. Double clicking a user will create a new room and invite that user to the chat. If the user already has a chat open in which the only other participant is the person that was double clicked, then that chat window will come into focus.

The conversation window has three main components. A log of the conversation appears in the top left, the people currently in the chat room are in the top right. The bottom text box is where a user types to input a message. Hitting enter will send the message to all users currently in the room.



To invite additional people to the room, the user clicks on the "Invite a new user" button, which brings up a window with all online users not in the current room. From there the user can invite other online users to the chat, at which point the invite dialog box will pop up for the invitees, where they can subsequently accept the invitation, and be added to the chat room, or decline the invitation.





For the sake of simplicity, duplicate objects were not redefined in order to save space. For example, only one of the User objects and one of the roomLists are fully specified.

Thread Saftey Arguments

ChatServer

The chat server is threadsafe because the only fields that are ever accesed by multiple threads are synchronized hashmaps (which are threadsafe data objects). Also, there are no race conditions because any pair of methods can be called concurrently. This is because almost all methods just have one line which returns or does some operation on a synchronized hashmap, so there are no operations to interleave. The only method that has multiple lines that could be interleaved is createRoom. If a removeRoom call were interleaved between the roomList.put line and the return room line the roomList would not have all the rooms it should, which would violate the spec of the ChatServer. This will not happen though because the removeRoom is only called is by a room object on itself, and it is impossible for a room object to remove itself before it is given to a User (which only happens after createRoom finishes).

User

User objects only have threadsafe fields. Furthermore User objects have methods that can be divided into three basic categories.

1.) Methods that a User object calls on itself (these are methods that the client “calls”, ie if the client sends a createRoom message the user object calls its own createRoom method). These methods can never inteleave with each other.
2.) Methods that other Users/Rooms/ChatServer call on this user. These methods can all be called concurrently because the only thing they do is call the sendToUser method, whose entire body is wrapped in a synchronized(out){}, so its as if it is never called concurrently.

So methods of type 1 never interleave with each other, methods of type 2 can interleave with anything, but thats ok because it just sends the user some message and then exists (doesn't affect any fields of this User object).

Room

Room objects are threadsafe because each method mutates completely different fields. The only methods that mutate the same field are join and leave, but these objects will never be called concurrently because if a user object is joining a room it cannot be leaving a room at the same time.