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.

Client-Server Protocol

Server → client

Protocol ::= Message*
Message ::= NotifyContactOffline | NotifyContactOnline | NotifyRoomInvite | NotifyNewMsg |
    NotifyRoomCreated | NotifyError
NotifyContactOffline ::= ContactOffline Username Newline
NotifyContactOnline ::= ContactOnline Username Newline
NotifyRoomInvite ::= RoomInvite Username RoomNumber Newline
NotifyNewMsg ::= NewMsg Username RoomNumber Line Newline
NotifyRoomCreated ::= RoomCreated RoomNumber Newline
NotifyError ::= Error ErrorNo Line Newline
ContactOffline ::= <u>offline</u>
ContactOnline ::= <u>online</u>
RoomInvite ::= <u>invite</u>
NewMsg ::= <u>message</u>
RoomCreated ::= <u>roomcreated</u>
Error ::= <u>error</u>
Username ::= <u>[A-Za-z0-9_-]+</u>
RoomNumber ::= Number
ErrNo ::= Number
Number ::= <u>(1-9)[0-9]*</u>
Line ::= <u>.*</u>
Newline ::= <u>\n</u>

Client → server

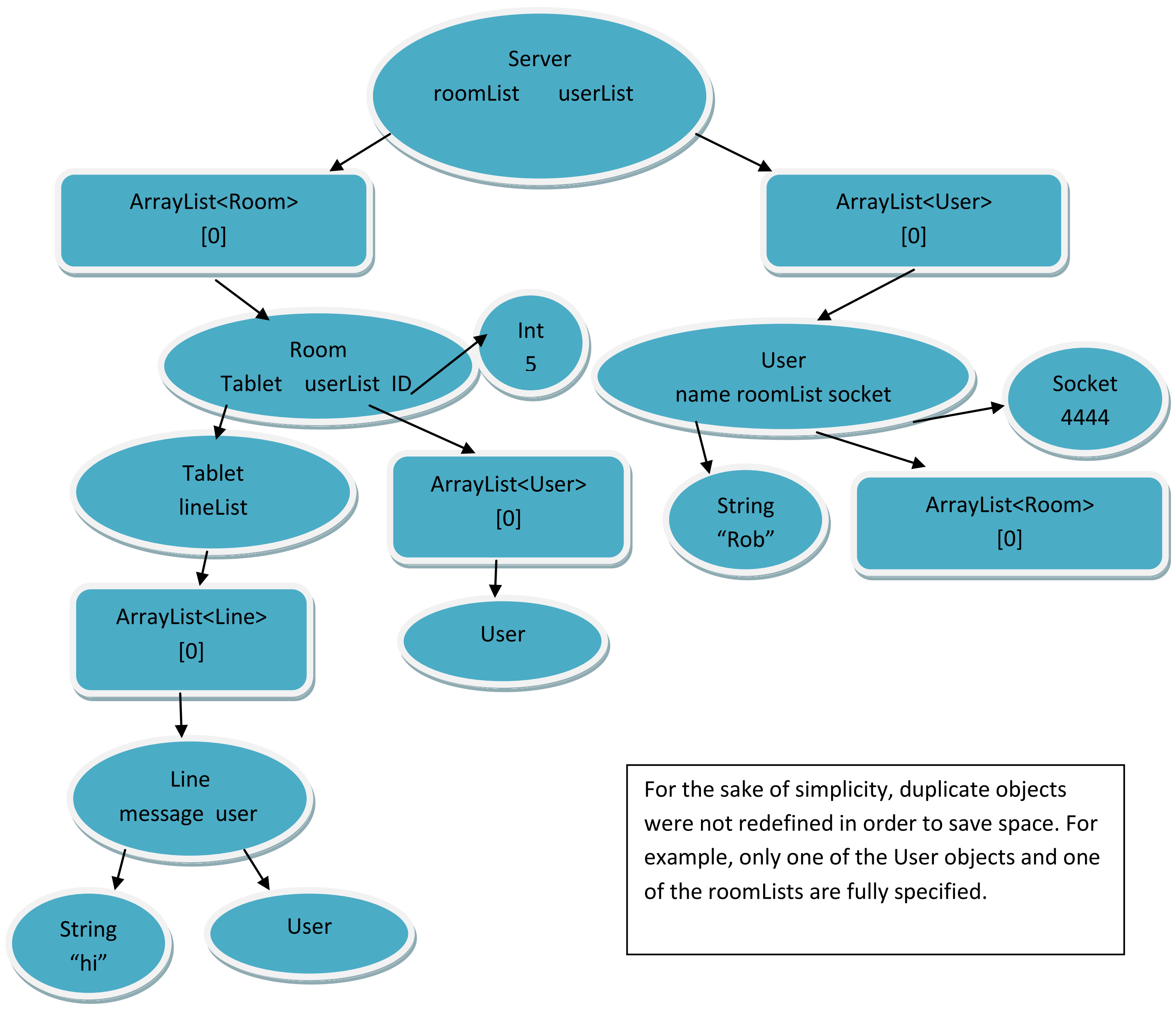
Protocol ::= Message*
Message ::= NotifyLogin | NotifyCreateRoom | NotifyInvite | NotifyAccept | NotifyDecline | NotifySay |
    NotifyLogout
NotifyLogin ::= Login Username Newline
NotifyCreateRoom ::= CreateRoom Username Newline
NotifyInvite ::= Invite From To RoomNumber Newline
NotifyAccept ::= Accept Username RoomNumber Newline
NotifyDecline ::= Decline Username RoomNumber Newline
NotifySay ::= Say Username RoomNumber Line Newline

```
NotifyLogout ::= Logout Username Newline
From ::= Username
To ::= Username
Login ::= login
CreateRoom ::= create
Invite ::= invite
Accept ::= accept
Decline ::= decline
Say ::= say
Logout ::= logout
Username ::= [A-Za-z0-9_-]+
RoomNumber ::= (1-9)[0-9]+
Line ::= .*
Newline ::= \n
```

1. Chat Server
   a) Constructor
      - ChatServer(int port)
        - constructs a new ChatServer that listens for connections on this port
   b) Public Methods
      - void notifyuserLoggedOut(User u)
        - notifies the server that the user u has logged out
        - requires that User u be non-null
      - void noitfyUserLoggedIn(User u)
        - notifies the server that the User u has logged out
        - requres that the User u be non-null
      - Room createRoom()
        - creates a new room and returns a pointer to that room
      - User getUser(String username)
        - returns a pointer to the User if he is online, null otherwise
      - Room getRoom(int num)
        - returns a pointer to the room with room number num if such a room exists, null otherwise

2. User
   a) Constructor
      - User(ChatServer server, Socket socket)
        - Constructs a new User object that belongs to the ChatServer server and listens to a client on the Socket socket
        - requires that ChatServer server and Socket socket be non-null
        - throws IOException
   b) Public Methods
      - void notifyContactEnteredRoom(User u, Room room)
        - notifies this user that the User u just entered the Room room
        - requires that User u and Room room be non-null
      - void notifyContactLeftRoom(User u, Room room)
        - notifies this user that the User u just left the room
        - requires that User u and Room room be non-null
      - void notifyContactOnline(User contact)
        - notifies this user that the this contact just came on line (ie logged into the server)
        - requires that User contact be non-null
      - void notifyContactOffline(User contact)
        - notifies this user that this contact just went off line (ie logged out of the server)
        - requres that User contact be non-null
      - void requestContact(String from)
        - notifies this user that the
        - requires that from be non-null
      - String username()
        - returns the username for this user

3. Room
    a) Constructor
        - Room()
            - constructs a new room object
    b) Public Methods
        - void addListener(RoomListener l)
            - adds the listener l to this room
            - requires that RoomListener l be non-null
        - void inviteUser(String from, String to)
            - notifies this room that the user whose username is from just invited the user whose username is to to this room
            - requires that from and to be non-null
        - int id()
            - returns the room number for this room



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.