6.005 Project 2: Conversation Design and Client-Server Protocol

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

TA: Joshua Oreman

Conversation Design

Definition of Conversation: Communication of text between two or more clients on the server.

Our Instant Messaging System will allow conversations between two people as well as conversations amongst multiple people. We will distinguish between conversations by giving each conversation a name. The name of conversations between 2 people will be their two usernames in alphabetical order (ex. if oremanj and leonid are chatting, they would be in the leonid-oremanj chat conversation). For conversations amongst more than 2 people, we will randomly assign a conversation name. For this initial design, we will not allow other users to join an already-existing chatroom.

Concurrency

Use a BlockingQueue in the Server to handle all incoming messages from clients.

Overview of Public Classes

Chat Package:

- <class> Server
    - Handles connections for and communication between multiple clients
    - Variables:
        - users: HashMap that keeps track of current instances of User that are connected to the server, key: String username, value: User that is associated with that username
    - Methods:
        - serve(): creates new instance of User class as a thread when a new user connects with a username
        - handleRequest(String input): make requested mutations to the Chat class if applicable, then return appropriate message to the user (See Server-to-Client protocol)

- <class> Client
  - Creates a thread when a new user begins the program on their computer
  - Choose an individual user alias to take for the chat time.
- <class> User
  - Static Methods:
    - isUser(String username) → true if this user has been previously created and is in our user database
  - Instance Methods:
    - addFriend(String username) → Adds friend to this user's friend list
    - isFriend(String username) → Checks whether a friend exists in the friends list
    - getFriendsList() → Returns HashMap of all friends of this user
    - isOnline() → true if a client currently connected to server is using username alias, false otherwise
    - setOnline(Boolean online) → set whether a client is using the username alias
    - isPassword(String password) → true if password corresponds to this User
    - sendMessage() → adds text to the Chat's BlockingQueue
    - getText() → returns the Chat's Conversation ArrayList
- <interface> Chat
  - used by User class to create dialogue
  - Uses a BlockingQueue to handle concurrency in sending messages to the specific instance of the class that extends Chat
  - Instance Variables:
    - private chatName → name of Chat. for DuoChat, the usernames of the participants in alphabetical order, separated by a dash. for GroupChat, a randomly generated chatName
    - private pendingText → BlockingQueue of text submitted by Users so far that have not been processed by Chat and displayed by the GUI
  - Methods:
    - updateConvo(): send next text from BlockingQueue to Server to update conversation
    - addLine(String text): add text to BlockingQueue
  - <class> DuoChat
    - Instance Variables:
      - private participants → list attribute that holds list of the two users in DuoChat
  - <class> GroupChat

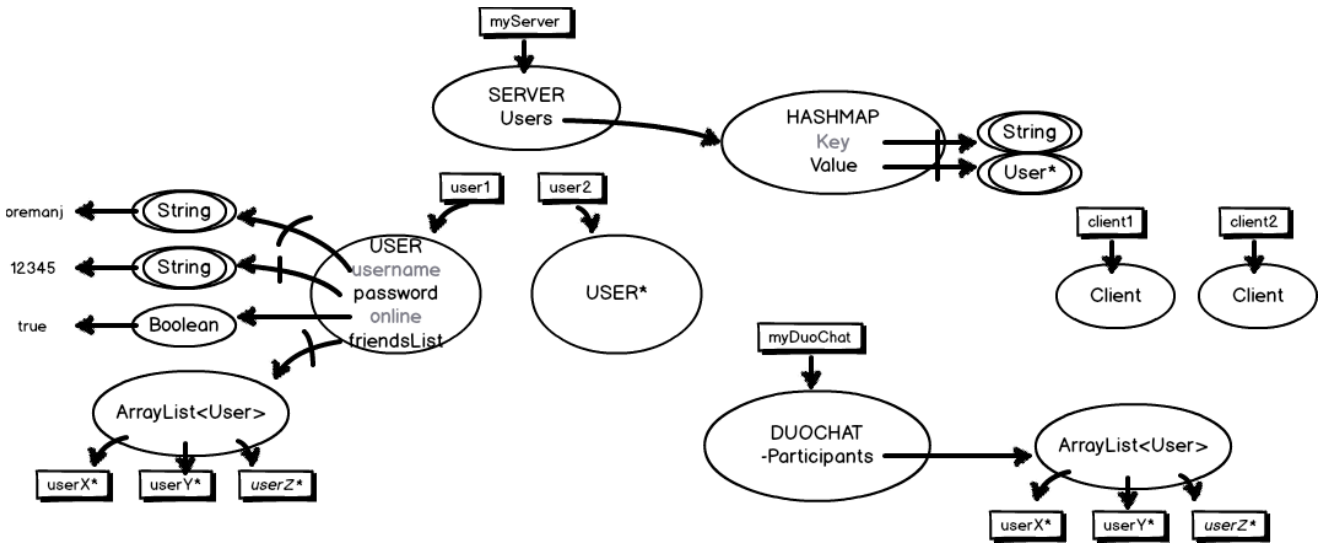
        - Instance Variables:
            - private participants → list attribute that holds list of all users in the GroupChat

GUI Package:

- <class> LoginWindow
    - Created when the program begins
    - Instance of Client class is also initiated when the LogIn Window is created.
    - Client must login to specific instance of User with valid username and password to enter ChatWindow state
    - Components:
        - JLabels to label “GUIChat”, “Username:”, “Password:”
        - username (JTextField) → enter String username
        - password (JPasswordField) → enter String password
        - login (JButton) → if login information is correct, LoginWindow closes and FriendWindow opens. else, an error message pops up, and LoginWindow fields are cleared
            - we will make this login JButton threadsafe so that two people trying to log in to the same User cannot both do so.
        - signup (JButton) → takes client to SignupWindow state
- <class> SignupWindow
    - Pops up if signup button in LoginWindow class is created
    - Client can create a new User alias
    - Components:
        - JLabels to label “Sign Up!”, “New Username:”, “New Password:”
        - username (JTextField) → enter String username
        - password (JPasswordField) → enter String password
        - signup (JButton) → creates new user with corresponding password if username is not already taken
            - we will make this signup JButton threadsafe so that two people trying to create an instance of User with the same username cannot both do so.
        - backToLogin (JButton) → returns to LoginWindow GUI state
- <class> FriendWindow
    - Default state for client who has assumed an alias
    - Components:
        - friendsOnline (JList) → All friends who are online

            - Double click on username to create ChatWindow to communicate with given friend
        - friendsOffline (JList) → All friends who are offline
            - Currently cannot open ChatWindows for offline friends (no communication allowed)
        - groupChatFriends (JTextField) → Enter usernames to group chat with, each username separated by commas
        - startGroupChat (JButton) → Creates GroupChatWindow to communicate with friends listed in groupChatFriends
        - newFriend (JTextField) → Enter username for new friend to add
        - addFriend (JButton) → Adds user whose username is text defined in newFriend
        - logout (JButton) →
    - For future iterations of our GUIChat, we will consider using JMenu instead of multiple JButtons
- <class> DuoChatWindow
    - Window for a conversation between two users
    - There exists only one chat window for each chat name, but each user can have multiple ChatWindows
    - Components:
        - newText (JTextPane) → enter text to be added to conversation
        - send (JButton) → send newText to conversation
        - conversationText (JEditorPane) → displays text exchanged in specific instance of GroupChatWindow's conversation
- <class> GroupChatWindow
    - Window for a conversation between more than two users
    - Instantiated with a random chatName
    - A client can have multiple GroupChatWindows open at the same time
    - Components:
        - newText (JTextPane) → enter text to be added to conversation
        - send (JButton) → send newText to conversation
        - conversationText (JEditorPane) → displays text exchanged in specific instance of GroupChatWindow's conversation
        - participants (JList) → list of usernames of Users participating in the group chat

Snapshot Diagram



* signifies a recursion

Details:

- Server creates instances of User.
- Each instance of Client accesses the server
- Based on the information that the client supplies, the client adopts a User instance.
- Instances of Chats are created using 2+ users

Client-Server Protocol

The Client and Server classes need to communicate in order to log into a specific User and to send messages within a Chat. We have specified protocols to allow for the following:

- a client logging in to a specific user account
- a client logging out from his/her account
- a client exitting a chat
- a user adding a friend
- establishing a DuoChat or GroupChat between multiple users
- exchanging messages between users

Client-to-Server Protocol

MESSAGE :== (SIGNUP | LOGIN | LOGOUT | ADDFRIEND | STARTDUOCHAT | STARTGROUPCHAT |
SENDTEXT | EXITCHAT) NEWLINE

SIGNUP :== “signup” SPACE USERNAME SPACE PASSWORD

LOGIN :== “login” SPACE USERNAME SPACE PASSWORD

LOGOUT :== “logout” SPACE USERNAME

ADDFRIEND :== “add” SPACE USERNAME SPACE USERNAME

STARTGROUPCHAT :== STARTDUOCHAT (SPACE USERNAME)+

STARTDUOCHAT :== “chat” SPACE USERNAME SPACE USERNAME

SENDTEXT :== “text” SPACE USERNAME SPACE CHATNAME SPACE IM

EXITCHAT :== “exit” SPACE USERNAME SPACE CHATNAME

CHATNAME :== (USERNAME “-” USERNAME) | (“chat” INT)

USERNAME :== [a-zA-Z0-9]{6,16}

PASSWORD :== [a-zA-Z0-9]{6,16}

IM :== .+

SPACE :== “ “

NEWLINE :== "\r?\n"

INT :== [0-9]+

Server-to-Client Protocol

MESSAGE :== (SIGNEDUP | NOTSIGNEDUP | LOGGEDIN | NOTLOGGEDIN | LOGGEDOUT |
FRIENDADDED | FRIENDNOTADDED| DUOCHATSTARTED | GROUPCHATSTARTED |
TEXTSENT| EXITEDCHAT) NEWLINE

SIGNEDUP :== “signedup” SPACE USERNAME SPACE PASSWORD

NOTSIGNEDUP :== “notSignedUp” SPACE USERNAME SPACE ERRORMSG

LOGGEDIN :== “loggedIn” SPACE USERNAME

NOTLOGGEDIN :== “notLoggedIn” SPACE USERNAME SPACE ERRORMSG

LOGGEDOUT :== “loggedout” SPACE USERNAME

FRIENDADDED :== “friendAdded” SPACE USERNAME SPACE USERNAME

FRIENDNOTADDED :== “friendNotAdded” SPACE USERNAME SPACE USERNAME SPACE ERRORMSG

DUOCHATSTARTED :== “DuoChatStart” SPACE CHATNAME

GROUPCHATSTARTED :== “GroupChatStart” SPACE CHATNAME

TEXTSENT :== “textSent” SPACE USERNAME SPACE CHATNAME SPACE IM

EXITEDCHAT :== “exitedChat” SPACE USERNAME SPACE CHATNAME

CHATNAME :== (USERNAME “-” USERNAME) | (“chat” INT)

USERNAME :== [a-zA-Z0-9]{6,16}

PASSWORD :== [a-zA-Z0-9]{6,16}

ERRORMSG :== .+

IM :== .+

SPACE :== “ “

NEWLINE :== "\r?\n"

INT :== [0-9]+