A design for a chat room server

JINWEI GU 14306748

15/10/2014

Group 1

1 Introduction

According to the group discussion, we would like to implement a simple server now as the first stage. We plan to use the iterative development method and to implement all functions step by step.

The server we designed allows clients to register, create a tag (send private message or build a public chat room), join a tag, and request for client lists/ tag lists.

All messages clients send or receive will all show on the command line window and in this stage we only have one window which is used to display all information.

The system consists of 4 main classes (Server, Client, Message and Tag). In the next parts, I will introduce the architecture as well as the detailed design of the server.

2 Architecture

2.1 Class Diagram

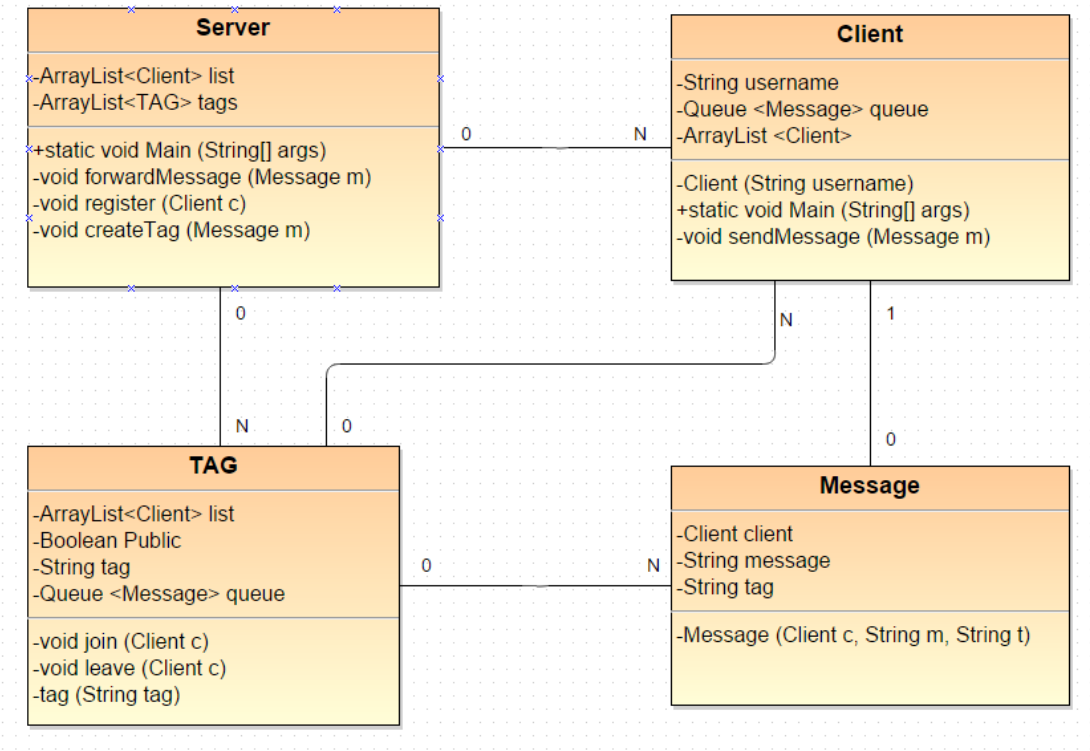

Figure 2.1 Class Diagram

3 Detailed design and process of each function

3.1 Group Messaging

In our design, clients could send group message to a public tag (like a chat room). The difference in a message for chat room and that of private person is the tag symbol. We use '@username' as a tag for a private message while we use '#chartroom's name' as a tag for a group message.

3.1.1 Chat room

For a user who want to send a message to a specific chat room, the first thing is to create a message which contains a tag with a symbol '#' and then server will check if the client exists in the client arraylist, if not, the user must register and then the server will add him/her into the list. At next step, server will check the tag in the message. If the tag does not exist in the tag arraylist, the server will create a new tag (a new chat room). But if there is already a tag same as the tag in the message, user will have two choices. First one is to join in this tag (chat room), and another one is to create a new one with a new chatroom name.

Upon a user join in one tag (chat room), server will forward the messages which are sent to this tag (chat room) to the user. User could send message to any chatroom only if he/she join in this public tag (chat room).

3.1.2 Client state

As the class diagram shows, server has the list for all registered clients and tag has the list for all joined clients in this chat room. In this stage, we do not care much about the client state, like whether the user is connecting to the server when new message posting. The server just forward the message to all clients joined in the same tag (their names will occur in the list). And unless the client is not connected at the time of message receipt, the user can receive all the chat room's messages.

3.1.3 Late comers

If a client is not connected with the server at the time of message receipt, user could ask server to print the message record for him/her when client connect with server again. The record is stored in queue in tag class. In our design, the number of record is limited because storing a lot record makes it slow for server to print latest messages. We plan to set the maximum record number as 25 or a little more than this. This record is continuously updated and always stores the latest 25 messages.

3.2 Private Messaging

As I mentioned in the group messaging, client could send a private message to another user. The first thing that the user must to do is to create a message in which the tag symbol is '@'. And then the server still need check if this user exist, if not, registration is necessary for the user to do. After this process, if the message receiver's name does not match any client in the client arraylist, the server will return an exception to client. On the other hand, if the receiver exists, server will store the message in the specific tag and forward the message to the receiver client. We set a variable of Boolean type in the tag class. This indicates this tag is a public tag (chat room) or a private tag (only one member in the tag-the message receiver). If the message is private, some methods like join and tag are both could not be invoked. Only the message could be stored in queue and only the message receiver could request a private message record from server. No user could join in a private tag and no other users could see the private message except the receiver.

3.3 Identify

In this stage, we just plan to identify users by their usernames, and this means every user's username must be unique. Their usernames are stored in the list when they register. Unregistered user cannot take any further actions until he/she finished registration. But in the further development, corresponding password and some other user information are necessary for identifying the user. However, we just focus on username in the simple design now.

4 Conclusion

Above is my group solution on the chat room server. We construct four main classes and plan to implement the basic functions like send group message, private message,

registration and request for message record.

But we still need add some more functions in the following development. The unique command line window is hard to read messages, if possible, I would like to design a model that separate chat has a dependent window and this will be clear for user to read messages.