1. **Definition of a conversations :**

Client can start a personal conversation by clicking the other client who is online on the GUI interface. Additionally, there is no “create” method for a conversation. The conversations/ chat-room aren’t owned by the user who initiates the chat-room/conversation as well. User can join as many chat-room/ conversations as he/she likes

Each conversation/chat room will be assigned a chat-room-ID by the server.

Users can type message in the chat-room; then, people who are in the chat room are expected to receive the message-updating, text showing up in the room’s board.

For those who are not in the room already exist, existing conversation rooms are expected to show up in a list of all-conversations on the ChatBoard-GUI. If they want to start a group talk with others, they can join the room by clicking the room image and leave the room afterwards; Besides, they are able to see the message history by previous users before they’re joined. As long as no client is in the conversation, the conversation object will be destroyed.

1. **Protocol for communication between server and client**

**Client-to-server Message Protocol (client online state)**

PROTOCOL :: = (START | LEAVE｜SENDMESSGAE)\*

START :: = TO SPACE USERNAME+

LEAVE :: = EXIT SPACE CHATROOM

SENDMESSGAE :: = SEND SPACE MESSAGE SPACE CHATROOM

MESSAGE ::= (TEXT SPACE?)+

USERNAME ::= ^[a-zA-Z0-9] (TEXT)\* SPACE

// A username should start with a-z or A-Z or digits and can also contain [-] or [\_] or [.]

CHARTROOM ::= “id:” DIGIT

TO ::= “to”

EXIT ::= “exit”

SEND ::= “send”

**Client-to-server Message Protocol (client offline state)**

PROTOCOL ::= CONNECT

CONNECT ::= TELNET SPACE HOSTNAME PORT

HOSTNAME ::= [DIGIT (.?)]+[:] | TEXT+[:]

TELNET :: = “telnet”

PORT ::= DIGIT+

**Server-to-Off-Line-client Message Protocol:**

Protocol :: = WELCOME | DISCONNECT

WELCOME ::= “Hello !” SPACE USERNAME “you’re connected to ” HOSTNAME

DISCONNECT ::= “Sorry server ” HOSTNAME “ is currently unavailable. Please try again.”

HOSTNAME ::= [DIGIT (.?)]+[:] | TEXT+[:]

USERNAME ::= ^[a-zA-Z0-9] (TEXT)\* SPACE

**Server-to-On-line-Client Message Protocol:**

Protocol ::= (UPDATEMESSAGE | JOINMESSAGE) \*

UPDATEMESSAGE ::= USERNAME SPACE “say” SPACE (TEXT SPACE? NEWLINE?)+

JOINMESSAGE ::= USERNAME SPACE “join” SPACE CHATROOM

USERNAME ::= ^[a-zA-Z0-9] (TEXT)\*

CHARTROOM ::= “id:” DIGIT

**General :**

TEXT ::= [a-zA-Z.-\_]

DIGIT :: = [0-9]

SPACE ::= “ ” // an single character space

NEWLINE ::= “\n”

1. **Class implantation**

Server - handling the multi-user/client connection and stores the ADT of IM system ex. conversations between clients, connected clients list , or states of all users.

* It’s a mutable.
* Fields:
  + List< String > Users
  + HashMap : Conversations
  + ServerSocket : Socket
  + List< String> : state of user ex. On-line or Off-line
* Methods :
  + Main ( hostname, port number ): start a server with given hostname and port number.
  + Serve(): handling a new socket from client then throw it to a new thread,

User - a client that can connect/disconnect to the server and invoke a MainBoard GUI . It is also a controller part of MVC.

* Fields :
  + Socket : userSocket
  + ChatBoard : GUI
* Methods :
  + User( hostname, port ): try to connect a socket to the server with specific address. -- constructor
  + OpenChatBoard : invoke the ChatBoard

Conversation – Model of the MVC, a mutable datatype that stores the messages between clients.

* Fields :
  + List< StringBuilder > Message
  + List<String > StateOfUser
* Methods :
  + Conversation () : initialize a conversation with given username, constructor
  + getMessage (): return the message of the conversation, observor
  + updateMessage (): update the message, mutator
  + deleteMessage() : delete the message, mutator
  + getUserState() : get the online/offline state off all users

ChatBoard extends JFrame – View of the MVC , also a GUI that a user can see conversation with others and list of online users and communicate with other user. User can start a conversation with other users by triggering an event in the MainBoard ex. Clicking another user to start to chat.

* Fields:
  + JComponement : JTable, Jbutton, JLabel….
  + Conservation : conversation of the current user with others.
* Methods:
  + ChatBoard() : construct a GUI of char room
* ActionListeners :