6.005 elements of software construction
Final project – IM message system

by Eric Lu

1. Definition of a conversation:

User can start a personal conversation by clicking the other desired user who is in online list on the GUI interface; meanwhile, a conversation will be created with a special ID after the message being sent from the user. Each conversation will have a list of current user. Users can also invite others to join the current chat room. For users who have multiple invitations, they can join as many chat-room/ conversations as he/she want. As long as no client is in the conversation, the conversation object will be destroyed.

2. Class implantation

Server-Side:

Server - handling the multi-user/client connection and pass it to client handler

- Fields:
  - HashMap< hash(username), ClientHandler > OnlineUsers
  - ServerSocket : Socket
- Methods :
  - Main (port number): start a server with given hostname and port number.
  - Serve(): handling a new socket from client then throw it to a ClientHandler, new thread.

ClientHandler extends Thread – a sub-server that handles the multiple client sockets. At first, it will check whether the username of client is available, if not, the client socket will be closed. It will parse client's I/O to a Message ADT, and put a queue to BlockingQueue shared with Conversation.

Client handler will also receive an output queue from not only conversation but also server in conditions such as other user leave the conversation or message updating from other user.

- Fields :
  - Socket : clientSocket
  - ObjectInputStream : in
  - ObjetcOutputStream : out
  - HashMap<Interger , Conversation > Rooms – room of the current user
  - BlockingQueue <Message> outputQueue -- produced by different conversation or the server thread

- Methods :
  - ClientHandler(Socket) – constructor
  - Handle () – read input and send output to client
  - handleRequest (ObjectInputStream) – performing action correspond to input Object.
  - checkName (String username) -- check whether the username is valid. If it is valid , the username will be added to user-list of Server.
  - CreateConversation ()– create a conversation if there is no such a conversation with given Id.
  - LeaveConversation () – leave the conversation
  - updateMessage () – Aftering parsing the ObjectInputStream and send the messages to Conversation's queue.
  - Run() : sits in a loop calling take() on the outputQueue

Conversation extends Thread– Model of the MVC, a mutable data-type that stores the messages between clients. Conversation will wait and take the queue, putting a message queue onto correspond client-handlers outputQueue.

- Fields :
  - BlockingQueue <Message> InputQueue
  - HashMap< username, ClientHandler > Users of all client in the conversation.
- Methods :
  - Conversation (BlockingQueue<message> message, Integer Id) : initialize a conversation with given id
  - Run() : sits in a loop calling take() on the inputQueue
  - addClient () : add client to Users list and notify all the other client in this conversation by putting a message to their output queue.
  - RemoveCLient() : remove clienthandler in the conversation and notify other clients
  - updateMessage() : After consuming a queue from InputQueue, send the message back to correspond client handler.

Client-Side :

User – At First, a user, controller part of MVC, will invoke a MainBoard-GUI. Client has to make a socket connection to the server with specific address, port number. Afterward the User(client) has to type username to server to check if it's valid. If the username is not valid or unique client are asked to

type a new one. No conversation would start if the username is not valid.

- Fields :
  - Socket : clientSocket
  - ChatBoard : GUI
  - String : Username
  - ArrayList<String> OnlineUsers
  - ObjectInputStream : in
  - ObjectOutputStream : out
- Methods :
  - User(hostname, port, username): -- constructor
  - OpenChatBoard : Invoking the ChatBoard graphic user interface.

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

Messages transferring (ADT) between Client and Server :

UserInfo implements ToServerMessage – it is immutable and represents the user-Information object that client pass to server via socket. Containing the user information such as username, the id of a conversation that user join. An

- Fields
  - String username
  - ArrayList< String > ChatRoomId
- Methods :
  - UserInfo(username, ChatRoomId) – constructor
  - addRoom () – add a new chatroom that user joined.
  - deleteRoom () – delete room that the user just leave.

ChatMessage implements ToServerMessage, ToClientMessage – representing a new message and user-info of the conversation with id specified from client.

- Fields:
  - String ConversationId
  - String updatingMessage
  - String From -- username
- Methods :
  - ChatMessage(String username, String id, String text) – constructor
  - getUsername () : return the message owner.
  - getRoom () : return the id of conversation this message belongs to
  - getText (): return the content of the message.

UserOlineList implements ToClientMessage – it contains the list of current online user and their information from server. It will be sent to all online users while a new user is online or someone is offline

- Fields
  - ArrayList<String > OnlineUsers
- Methods :
  - getUserList() : return the online username

ToClietMessage interface – an interface that implemented by server to client and contains a method of converting a string to a ChatMessage object.

ToServerMessage interface – an interface that implemented by all clients to server message and contains a method that convert a string to a ChatMessage object

3. Protocol for communication between server and client

Client-to-server Message Protocol

PROTOCOL :: = (START | LEAVE | SENDMESSGAE| DISCONNECT| CONNECT)*

START :: = TO SPACE USERNAME+

LEAVE :: = EXIT SPACE CHATROOM

SENDMESSGAE :: = SEND SPACE MESSAGE SPACE CHATROOM

DISCONNECT ::= QUIT USERNAME

CONNECT ::= TELNET SPACE HOSTNAME PORT

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

TELNET :: = “telnet”

PORT ::= DIGIT+

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”

QUIT ::= “quit”

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

Protocol :: = (WELCOME | DISCONNECT| UPDATEMESSAGE | JOINMESSAGE) *

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

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

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

JOINMESSAGE ::= USERNAME SPACE “join” SPACE CHATROOM

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

CHARTROOM ::= “id:” DIGIT

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

General :

TEXT ::= [^(NEWLINE|SPACE)]+

DIGIT :: = [0-9]

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

NEWLINE ::= “\n”