Design Milestone 1

Alvaro Morales, Ayesha Bose, John Holliman

{alvarom, aybose, holliman}@mit.edu

Our IM system will support multiple users joining a chat room. Users can choose to create a new chat room or join an existing one from a displayed list. If there are zero users in a chat room, the chat room is deleted.

Client-Server Protocol

We use Request and Response objects for server/client communication. More specifically, we convert Java objects to JSON, send the JSON objects between the client and server, and then convert the received JSON object back into a Java object. For instance, when a client wishes to communicate with the server a Request object of a specified type is created, converted to JSON using GSON, a Java library used to convert Java objects into their JSON representation, and sent to the server where it is converted back into a Java object. The same is done with Response objects for communication from the server to client.

A Request object is used in the following cases: login/logout (connect to server with specified nickname), join/leave a chat room, get users in a chat room, and send a message in a chat room.

A Response object is used in the following cases: login (notify of success/failure), join a chat room (notify of success/failure), send a message (notify of success/failure), leave a chat room (notify of success/failure), send a message (notify of success/failure and also distribute to other users in a chat room).

Note: We use method overloading to construct Request object of different types with different attributes.

class Request
The type of Request will dictate the content of the object and whether or not it has any of the optional parameters.

- Type (the types are those enumerated above)
- Message (optional)
- Content

class Response
The type of Response will dictate the content of the object and whether or not it has any of the

optional parameters.

- Type (the types are those enumerated above)
- Message (optional)
- Content

GSON (https://code.google.com/p/google-gson/)

“Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of. There are a few open-source projects that can convert Java objects to JSON. However, most of them require that you place Java annotations in your classes; something that you can not do if you do not have access to the source-code. Most also do not fully support the use of Java Generics. Gson considers both of these as very important design goals.”

Overall Functions

Server: accept connections
User: handle requests
Chat Room: broadcast messages to users

SERVER-SIDE

class Server implements Runnable

The server accepts sockets, and creates a new thread per incoming connection.
A server has the following attributes:

- Serversocket

Methods:

- public void run() // creates a new User thread per incoming connection

class User implements Runnable

A user will be allowed to enter the system by creating a username. They can then join chat rooms. A user listens to requests send from the client, and routes them to the appropriate chat room.
A user will have the following attributes:

- username
- Socket

class ChatRoom implements Runnable

We’ve defined our conversation to act as a chat room. Users can sign in and join a list of available chat rooms or create a new chat room. If a chat room has no users in it, the server removes that chat room. On joining a chat room, the user can message other users and receive messages from other users.
Each chat room will implement Runnable and have the following attributes:

- name (Each chat room will have to have a unique name)
- list of users connected

Methods:

- run() // routes responses to users
- hashcode (override Object method)
- void addUser(Socket soc) //add user to chat room
    - param: Socket soc that a user is using
    - return: Response to user that adding was successful
- void removeUser(Socket soc) //remove user from chat room
    - param: Socket soc that a user is using
    - return: Response to user that removing was successful

class Message

A message will have the following attributes:

- message (String representing message)
- timestamp
- username (String corresponding to user that wrote the message)
- destination (conversation tag of some sort)

Methods:

- hashcode (override Object method)
- toString (return message string)

CLIENT-SIDE

class ChatSession implements Runnable

The ChatSession class routes messages to the appropriate ChatWindow by name.
It will have the following attributes:

- chatWindows (list of chat windows)

Methods:

- void routeMessage(ChatWindow c, Message m)
    - param: ChatWindow c to display message
    - param: Message m to send to chatwindow

class ChatWindow

The ChatWindow is a client representation of the Server ChatRoom.
It will have the following attributes:

- name (Each chat room will have to have a unique name)

Methods:

- hashcode (override Object method)
- void sendMessage(Message m) //sends message to server
    - param: Message m that is what the client sends to the server
- void getListOfClients()

- void leaveRoom()
- void addMessage()