Group 7 Communications: API Specification Documentation

Server Class

Purpose: The main server class for the communication app.

1. Public static void main

Purpose:
The main driver for the server program.

Implementation:
Starts the server and listens for incoming connections, loads the list of users, spins threads to handle connections, and allows for messages to be handled

---

ClientHandlerClass

Purpose: Handles communication with each client.

1. public ObjectOutputStream getObjectOutputStream

Purpose: Establish the connection between the server and the client so messages can be passed

Response: ObjectOutputStream is created

2. private void authenticate

Purpose: Receive the login request from the client and check the credentials of the login and send data over

Request: Message object - containing user and password as a string

Implementation: takes the input and compares to database of users
Response: Message object - containing the user data if login is successful

3. private void sendSynchronousMessage

Purpose: Send a message to a user that is online on the client.

Request: ObjectOutputStream of client, Message Object containing message sent and user info such as senderID and ReceiverIDs

Implementation: Check each client and the user ID stored in the client compare to the ID that is in the message Object and send the message to the specified client.

Response: void, message is sent through output tunnel

4. void offlineMessage

Purpose: if the user is offline, store the sent message to that user in a database (text file)

Request: Message Object

Implementation: method called if a boolean is false, send the message to a file to store until the user comes online.

5. void loadAsyncMessages

Purpose: Initial load method to load messages that were not sent yet. Used at server start to load from file.

Implementation: Using a FileWriter and ObjectOutputStream to store message objects to a text file.

6. void sendAsynchronousMessage

Purpose: When the user comes online, send all the messages that were sent to the user while they were offline.

Implementation: check if a user comes online, send the message to them.

7. void logout

Purpose: logout the user from the client

Request: Message Object of MessageType.LOGOUT

Implementation: read the message and log the user out by setting the user instance to null elements

8. void getUserDirectory

Purpose: get the list of users and their status to the client for displaying

Request: Message Object

Implementation: take a request and get all the users from the user list

Response: A string used to parse the usernames and statuses for the GUI to Display.

Client Class

Purpose: the Application that has the functions to login to a server, send messages, logout

1. public Client()

Purpose: Constructor… create an instance of a client

Implementation: set the elements to the correct respective values

2. public Boolean login

Purpose: Send a login request to a server to authenticate

Request: a Username and a Password in string format

Implementation: takes the strings and sends it as a message object along with the message type = Message

Response: Receives a message that will allow for the user information to be passed to the client once authenticated.

3. public void sendMessage

Purpose: Used to send a message to another client, through the server.

Request: A message with a Message String, a sender, sender ID, ReceiverIDs, and message type

Implementation: Use a ObjectOutputStream to write the created message through to the server

4. public void receiveMessage

Purpose: Receive messages from the server

Request: A Message object with parameters such as message, sender, receiver, senderID, receiverID, MessageType.Type

Implementation: Use An ObjectInputStream to read the message sent from the server.

5. public String viewLog

Purpose: Used to send the list of logged messages for display on the GUI

Request: a String indicating how the log will be sorted to display on the GUI

Implementation: Take a string in, return a list by calling the designated log function

Response: return a String that contains all of the logs

6. public Boolean logout

Purpose: Sends a logout request to the server

Request: none

Implementation: use ObjectStream to send to server

Response: Sends a message to the server with MessageType.LOGIN

7. public String getUserDirectory

Purpose: Get the list of users and their status to the GUI

Request: None/ a button press by the USER

Implementation: send a Message indicated type of DIRECTORY

Response: return a String that contains the userDirectory of all users and their status.

User Class

Purpose: Used for creating instances of users for Client and Server to use.

1. public User

Purpose: default empty constructor… Create a User Instance that will set all values equal to null

Implementation: Java Constructor format, setting all values in the constructor to null

2. public User

Purpose: constructor using parameters…Constructor that will create the User instance with given information rom function parameters

Implementation: take in a userName, userRole, userPassword, user status, and login status and set the values in the Java constructor

3. private void SetID

Purpose: set the unique id that is assigned to the user.

4. void setPassword

Purpose: Set the password of a user

Request: A password in String format for the setter to set the value equal to.

5. void signOut

Purpose: set the boolean indicators to show that the user signed out

6. void signIn

Purpose: set the boolean indicators to show that the user signed in

7. void setRole

Purpose: set the specific role of the user to either REGULAR or IT roles.

Request: Takes in the role of a user

8. Boolean getLoginStatus

Purpose: Used to see if the user is logged in or not

Response: returns a boolean element of a user instance

9. Boolean getOnlineStatus

Purpose: Used to see if the user is online

Response: returns a boolean element of a user instance

10. Role getRole

Purpose: get the role of a specific user instance

Response: return the ROLE of a user

11. Integer getID

Purpose: get the ID of a specific user instance

Response: return a integer that is defined as the user instance's ID

12. String getPassword

Purpose: Get the password of a specific user instance

Response: return the string which contains the password of the user instance

13. String getUserName

Purpose: get the username of the user instance

Response: returns a string containing the username of the user instance

14. public String toString

Purpose: used to display all of the information of a user

Response: Returns a string that contains all of the information of a user instance

---

UserLoader Class

Purpose: Loads the users from a database (.txt) and creates user instances for those users. Used for sending and receiving messages.

1. public static ArrayList<User> loadUsersFromFile

Purpose: Takes a file and loads the contents to the server

Request: requests a File name in format "name.txt"

Implementation: parse a file using BufferedReader and create user instances to store in an arrayList to send to server

Response: returns an ArrayList of users that were instantiated

---

Role Class

Purpose: Used to enumerate the different types of roles a user could be assigned to

---

Message Class

Purpose: Used as the main type sent through from server and client streams

1. public Message(String messageString, String messageSender, String messageReceiver, MessageType messageType)

Purpose: Constructor… Create a message that is used for logging in and logging out

Request: Requests information in format of java Constructor to set respective elements to those values such as the message contents itself, the sender, the receiver, and the type of message

Implementation: standard Java constructor

Response: none, message object values are updated or set

2. public Message(String messageString, String messageSender, MessageType messageType, Integer senderUID, ArrayList<Integer> receiverUID)

Purpose: Constructor… Create a message that is used for creating the messages to send

Request: Requests information in format of java Constructor to set respective elements to those values such as the message contents itself, the sender, the receiver, and the type of message

Implementation: standard Java Constructor for class

Response: message is created

3. private void SetID

Purpose: Set the ID of the message

4. void setMessageDate

Purpose: set the date of when the message was sent

Request: a Date object

5. void updateIsSent

Purpose: indicates if a message is sent

6. private void updateMessageType

Purpose: change the message type of the message object

Request: a MessageType

7. String getMessageString

Purpose: get the contents that is stored in the message

Response: returns a string with the contents of the message

8. Date getMessageDate

Purpose: get the date of when a message is sent

Response: a Date object

9. String getMessageSender

Purpose: get the sender’s username

Response: a String of the sender username

10. MessageType getMessageType

Purpose: get the type of the message that was sent

Response: a MessageType of the message Object

11. Integer getMessageID

Purpose: get the ID of the message object

Response: a Integer that indicated the ID

12. Integer getMessageSenderUID

Purpose: get the UID of the sender of the message object

Response: a integer representing the Unique ID of the sender

13. ArrayList<Integer> getReceiverUID

Purpose: get the list of Receiver Unique IDs of the the Message Object

Response: and arraylist of Integers that represents the UIDs of receivers

14. Boolean getIsSent

Purpose: get the indicator of which the message Object was sent or not.

Response: a Boolean

15. public String toString

Purpose: display the contents of the message object

Response: a String that contains all of the information of the Message object

---

MessageType Class

Purpose: Used to enumerate the types of messages that could exist

---

Chat Class

Purpose: Used to create instances of chats for users and messages to exist in

1. public Chat

Purpose: Constructor… Used to create a chat instance

Implementation: standard Java Constructor

2. private void SetID

Purpose: set the UID of the Chat instance

3. ArrayList<Integer> getParticipantsUID

Purpose: get the UID of participants in a chat instance

Response: a Arraylist of Integers that represent the UIDs of the participants

4. String getParticipants

Purpose: get the participants of a chat instance

Response: a String containing all of the participants

5. String StringChatroomID

Purpose: get the chatroomID of the chat instance

Response: a String containing the chatroom ID

6. Integer ChatroomID

Purpose: get the chatroom ID if a chat instance

Response: a integer representing the ChatroomID of a chat instance

---

Log Class

Purpose: functions used to store messages sent to a log file for IT viewing

1. public boolean isIT

Purpose: checks if a user is of role IT

Request: a User object

Response: a boolean

2. public void updateLoggedMessageArray

Purpose: update the current array of logged messages to use for viewing

Implementation: add to a Java ArrayList of Messages

3. public String getAllLogs

Purpose: get log of all messages sent

Implementation: iterate through array of messages and add it to a string the send it over to designated area

Response: a String containing all of the logs separated by a new line

4. public String filterLogsBySender

Purpose: get the log of messages sent by a specific sender

Request: a string of a sender’s username

Implementation: iterate through the arraylist and find messages sent by the sender sent to this method.

Response: a String containing all of the messages sent by provided user

5. public String filterLogsByDate

Purpose: get the log of messages sent on specific date

Request: a Date in the format MM.DD.YY

Implementation: iterate through the arraylist of messages and find messages of certain date and store it to a String

Response: a String containing all of the messages of a certain date separated by a new line

6. public void addMessageToFile

Purpose: logging the message to designated file

Request: a Message Object

Implementation: change the format of the date to MM.DD.YY and then store the toString of Message to a specified File

---

DataBase

The type of database used may vary depending on the implementor. In our case we used log files. Objects are stored to the file asyncMessages.txt, Strings are store to commlogs.txt, and users.txt contains Strings with the user information.