Aloha Chat System

It will consist of 3 modules:

Base Module

It will consist of these classes:-

1) Message class - It will contain the logic to separate the different parts of the message, like Prefix, data, etc.

Examples :-

a) JOIN (DM Message) :

"Arghya@98.19.67.45:34000 0 0 Avraneel@78.67.34.87:34000 : User Avraneel has joined!!!\n"

*) Arghya@98.19.67.45:34000 <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 0 stands for JOIN Numeric.
*) Second 0 stands for Direct Message(DM).
*) Rest is data.

b) JOIN (Group Message) :

"GroupName@ServerIP:Port 0 1 Avraneel@78.67.34.87:34000 : User Avraneel has joined!!!\n"

*) GroupName@ServerIP:Port <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 0 stands for JOIN Numeric.
*) Second 1 stands for Group Chat.
*) Rest is data

c) QUIT (DM Message) :

" Arghya@98.19.67.45:34000 1 0 Avraneel@78.67.34.87:34000 : User Avraneel has quit!!!\n"

*) Arghya@98.19.67.45:34000 <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 1 stands for QUIT Numeric.
*) Second 0 stands for Direct Message(DM).
*) Rest is data

d) QUIT (Group Message) :

"GroupName@ServerIP:Port 1 1 Avraneel@78.67.34.87:34000 : User Avraneel has quit!!!\n"

*) Arghya@98.19.67.45:34000 <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 1 stands for QUIT Numeric.
*) Second 0 stands for Group Chat.
*) Rest is data

e) PRIVMSG (DM Message) :
" Arghya@98.19.67.45:34000 2 0 Avraneel@78.67.34.87:34000 : Hey Avra!!!\n"
*) Arghya@98.19.67.45:34000 <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 2 stands for Private Message Numeric.
*) Second 0 stands for DM.
*) Rest is data

f) PRIVMSG (Group Message) :
"GroupName@ServerIP:Port 2 1 Avraneel@78.67.34.87:34000 : Hey guys!!!\n"
*) GroupName@ServerIP:Port <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 2 stands for Private Message Numeric.
*) Second 1 stands for Group Chat.
*) Rest is data

g) NICKCHANGE (DM Message) ;
" Arghya@98.19.67.45:34000 3 0 Avraneel@78.67.34.87:34000 : /nick Avra!!!\n"
*) Arghya@98.19.67.45:34000 <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 3 stands for Nick Change Numeric.
*) Second 0 stands for DM.
*) Rest is data
Return Message is:
" Arghya@98.19.67.45:34000 3 0 Avraneel@78.67.34.87:34000 : User Avraneel is now known as Avra!!!\n"

h) NICKCHANGE (Group Message) :
" GroupName@ServerIP:Port 3 1 Avraneel@78.67.34.87:34000 : /nick Avra!!!\n"
*) GroupName@ServerIP:Port <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 3 stands for Nick Change Numeric.
*) Second 1 stands for Group Chat.

*) Rest is data

Return Message is:
" GroupName@ServerIP:Port 3 0 Avraneel@78.67.34.87:34000 : User Avraneel is now known as Avra!!!\n"

e) PART (DM Message) :
" Arghya@98.19.67.45:34000 2 0 Avraneel@78.67.34.87:34000 : User Avraneel has parted!!!\n"
*) Arghya@98.19.67.45:34000 <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 2 stands for Private Message Numeric.
*) Second 0 stands for DM.
*) Rest is data

f) PRIVMSG (Group Message) :
"GroupName@ServerIP:Port 2 1 Avraneel@78.67.34.87:34000 : User Avraneel has parted!!!\n"
*) GroupName@ServerIP:Port <- Receiver's Address
*) Avraneel@78.67.34.87:34000 <- Sender's Address
*) First 2 stands for Private Message Numeric.
*) Second 1 stands for Group Chat.
*) Rest is data

The data part shouldn't end with '\n'.

2) Prefix class - It will contain 3 data members namely Nick & IP and Host, which will be public members, separated by '@' & ':'.
Example: Arghya@45.96.67.89:34000
Prefix class shouldn't have any other characters like '@', ':' except at the normal places.

3) Connection class – It's a abstract class. Used in connecting the client and server to the internet.
Functions :
a) Connect() - Helps in connecting the class to the internet
b) sendDataAsync(String) – Sends data String object asynchronously
c) sendData(String) – Sends data string object
d) receiveData(): String? – Receives a null or a String object
e) receiveUTF8Data(): String? - Receives a null or UTF8 string object

f) Disconnect() - Disconnects the object

Member variables :

a) connected: Boolean – Used as a flag
b) output: ByteWriteChannel – Used to write data to the socket
c) input: ByteReadChannel – Used to read data to the socket
d) socket: Socket – Used to open the socket for the connection
e) port: Int – Port to connect to
f) address: String – Address of the other client/server

4) BasicConnection class – It inherits the Connection abstract class and declares its member methods.

5) Channel class - This class contains a queue of the ChatMsg objects received by the channel(here channel can be public channel or DM). It will help in both server and client module.

Member variables :

a) messageList: Queue<ChatMsg> - List of messages for the channel
b) channelPrefix: Prefix – Prefix for the channel
c) channelName: String – Name of the channel
d) dmORgroup: Boolean? - Used to state if the Channel object is DMChannel(true) or GroupChannel(false)

Functions :

a) Channel(String,Int) – Construct a Channel object and

In my implementation the Channel class is a abstract class, which we use to inherit into DMChannel class(for Direct messages) and GroupChannel class(for group messages).

DMChannel Class -

Member variables :

a) receiverPrefix: Prefix – Prefix for the recipient

Functions :

a) DMChannel(Prefix,Prefix) – Construct the receiverPrefix and the channelPrefix

GroupChannel Class -

Member variables :

a) receiverGroupPPL: MutableList<Prefix> - List of the prefixes of the people ‘ in the channel

Functions :

a) addPPL(Prefix) – Prefix to be added
b) ifPplIsThere(Prefix): Boolean – Check if the Prefix is there or not
c) removePPL(Prefix) – Remove a particular prefix

For GroupChannel objects, its not possible to instantiate all the users at runtime. People may come and go at any instant.Also both DMChannel & GroupChannels will be present in the server, so they need to be mixed and matched from the incoming connections.

6) AlohaClient :

This is class which helps in client connection of the software users. It is also present in the Server module to help find the users for the server. READ: WIP

Member Variables :

a) channels : MutableList<Channel> - List of channels the client is joined in
b) conn : BasicConnection – Connection used to connect to the server.
c) serverIp : String – IP of the server
d) myIp – IP of the client
e) port – Port of the client and the server
f) prefix – Prefix of the client

Functions :

a) AlohaClient(String,String,String) - Constructs the object. The parameters are nick, ip and port. This 3 helps in initialising the myIp, prefix, serverIp, port and conn member variables.
b) sendMsg(ChatMsg) – Sends a chatmsg to the server
c) joinChannel(Channel) – Helps in sending a channel join message and adds the channel to the channels MutableList
d) partChannel(Channel) - Helps in sending a channel part message and parts the given channel.
e) quitServer() - Helps in sending a quit message to all the channels and disconnects the connection.
f) sendPrivMsg(String, Channel) – Helps in sending a private message (the String

object) to the channel.

g) receiveMsg(ChatMsg, Channel) – Receive a message from that channel into channels object

7) Helper functions :

a) Numerics – It is a enum. Samples – JOIN, QUIT, PRIVMSG, NICKCHANGE, PART.
b) retNumeric(int) : Numerics – Convert a int to a appropriate numeric.
c) MsgData – Data class to store the ChatMsg data members.
d) msgDataToStr(MsgData) : String – Return the String object of MsgData
e) matchPrefix(Prefix,Prefix) : Boolean – Match 2 prefixes
f) findPublicIP() : String – Find the public IP
g) parseStrToMsgData(String) - Convert a String object to MsgData
h) findLocalIP() - Find the local IP of the system.

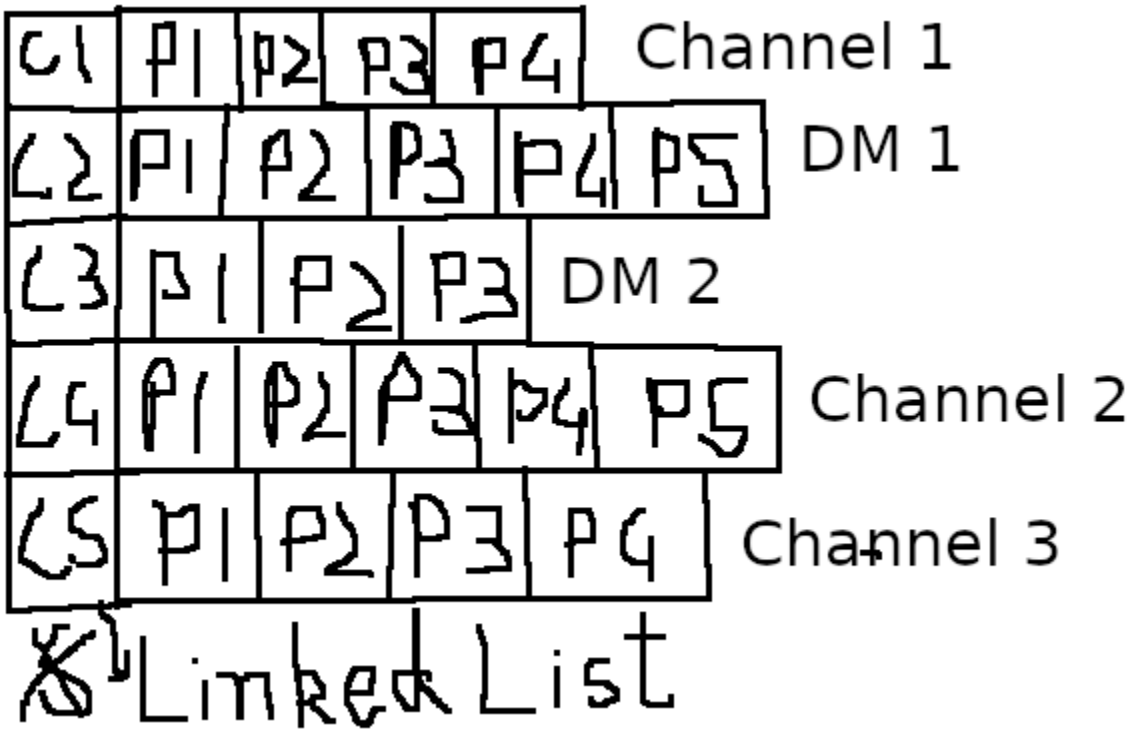
Client Module

This application contains two java classes. One is for server and other is for client. Java socket is used to connect them together. To run the application first we need to run the server and then client. GUI enabled window will be shown.

Server Module

The chat server is implemented by these classes.
i) ChatServer class - The ChatServer class starts the server, listening on a specific port. This will be the main running class of the server. It also contains a ChannelList class, which is used to iterate over the Channel classes present in the messages, and print the required output to the receiver.

Since each connection is processed in a separate thread, the server is able to handle multiple clients at the same time. When a client sends a message to the server, the server receives it from the client and checks if the client is present or not through its Prefix. If it is present, then it finds the Channel for the client through iterating over ChannelList class. If not then it adds a new Channel class with the ip of its receiver.


C1
P1
P2
P3
P4
Channel 1
C2
P1
P2
P3
P4
P5
DM 1
C3
P1
P2
P3
DM 2
C4
P1
P2
P3
P4
P5
Channel 2
C5
P1
P2
P3
P4
Channel 3
Linked List