Aloha Chat System

It will consist of 3 modules:

Base Module

It will consist of these classes:-

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

Examples :-

a) "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) "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) " 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) "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) " Avraneel@78.67.34.87:34000 2 0 Arghya@98.19.67.45:34000 : 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) "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) " GroupName@ServerIP:Port 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) " 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"

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

ii) 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.

iii) Connection class - It will be a base class which is used to inherit the BasicConnection class from.

iv) BasicConnection class - It will be the class that implements the TCP plain connection to be used both by the server and client.

v) 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.

The Channel class should have these following members:-

i) Prefix[ ] receiver : If it is a DM, then it will only have one element, else if group chat then multiple.

ii) Prefix sender : Only one prefix is accepted. The channel class should get instantiated with this.

Channel Class should have this constructor:-

i) public Channel(Prefix sender)

In our 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).

i) GroupChannel class has the constructor : public GroupChannel(Prefix sender)

ii) DMChannel class has the constructor : public DMChannel(Prefix sender, Prefix receive)

For GroupChannel objects, its not possible to instantiate all the users at runtime. People may come and go at any instant. Thus we will add and remove, plus find if a user is there using these 3 functions :-

i) addPPL(Prefix prefix)

ii) ifPplIsThere(Prefix prefix)

iii) removePPL(Prefix prefix)

Also GroupChannels will be present in the server, so they need to be mixed and matched from the incoming connections.

Channel class should be able to run on threads, because in the ChatServer class, we will fire threads over its ChannelList member variable and run them asynchronously.

vi) Helper functions like to find if 2 IP are same, to find the correct message type sent by

the client,etc

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.