Aloha Chat System

It will consist of 3 modules:

1)Base Module

It will consist of these classes:-

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

This class will store messages of these formats :-

a) Join message (DM) - “Arghya@56.89.67.11:34000 0 0 Avraneel@78.89.64.56:34000 : User Avraneel has Joined”

*) Arghya@56.89.67.11:34000 <- Receiver’s Address
*) Avraneel@78.89.64.56:34000 <- Sender’s Address
*) First 0 stands for Numeric 0, which is JOIN.
*) Second 0 stands for Direct/Private Message.
*) “User Arghya has Joined” <- Data to be sent.

b) Join message (Group Chat) - “GroupName@ServerIP:Port 0 1 Arghya@56.89.67.11:34000 : User Arghya has Joined”

*) GroupName@ServerIP:Port <- Receiver’s Address
*) Arghya@56.89.67.11:34000 <- Sender’s Address
*) First 0 stands for Numeric 0, which is JOIN.
*) Second 1 stands for Group Message.
*) “User Arghya has Joined” <- Data to be send.

c) Quit message (DM) - “Arghya@56.89.67.11:34000 1 0 Avraneel@78.89.64.56:34000 : User Avraneel has Quit”

*) Arghya@56.89.67.11:34000 <- Receiver’s Address.
*) Avraneel@78.89.64.56:34000 <- Sender’s Address.
*) First 1 stands for Numeric 1, which is QUIT.
*) Second 0 stands for Direct Message.
*) “User Avraneel has Quit” <- Data

d) Quit message (Group Chat) - “GroupName@ServerIP:Port 1 1 Arghya@56.89.67.11:34000 : User Avraneel has Quit”

*) GroupName@ServerIP:Port <- Receiver’s Address.
*) Avraneel@78.89.64.56:34000 <- Sender’s Address.
*) First 1 stands for Numeric 1, which is QUIT.

*) Second 1 stands for Group Chat.
*) “User Avraneel has Quit” <- Data

e) Private Message (or DM) - “Arghya@56.89.67.11:34000 2 0 Avraneel@78.89.64.56:34000 : Hey Arghya !!!”

*) Arghya@56.89.67.11:34000 <- Receiver’s Address.
*) Avraneel@78.89.64.56:34000 <- Sender’s Address.
*) First 2 stands for Numeric 2, which is QUIT.
*) Second 0 stands for Direct Message.
*) “Hey Arghya !!!” <- Data

f) Private Message (in Group Chat) - “GroupName@ServerIP:Port 2 1 Arghya@56.89.67.11:34000 : User Avraneel has Quit”

*) GroupName@ServerIP:Port <- Receiver’s Address.
*) Avraneel@78.89.64.56:34000 <- Sender’s Address.
*) First 2 stands for Numeric 2, which is QUIT.
*) Second 1 stands for Direct Message.
*) “Hey Arghya !!!” <- Data

g) Nick Change (DM) - “Arghya@56.89.67.11:34000 3 0 Avraneel@78.89.64.56:34000 : Avraneel is now known as AvraNeel !!!”

*) Arghya@56.89.67.11:34000 <- Receiver’s Address.
*) Avraneel@78.89.64.56:34000 <- Sender’s Address.
*) First 3 stands for Numeric 3, which is Nick Change.
*) Second 0 stands for Direct Message.
*) “Avraneel is now known as AvraNeel !!!” <- Data

h) Nick Change (DM) - “ GroupName@ServerIP:Port 3 1 Avraneel@78.89.64.56:34000 : Avraneel is now known as AvraNeel !!!”

*) GroupName@ServerIP:Port<- Receiver’s Address.
*) Avraneel@78.89.64.56:34000 <- Sender’s Address.
*) First 3 stands for Numeric 3, which is Nick Change.
*) Second 1 stands for Group Chat.
*) “Avraneel is now known as AvraNeel !!!” <- Data

ii) Prefix class - It will contain 3 data members namely IP and Host, which will be public members, separated by ‘:’, and Nick, which will be separated from the rest by ‘@’. Example - “Arghya@113.43.32.190:34000”, “Neel@56.29.78.93:34000”. Nick must not have this 3 characters: ‘ ’ (space), ‘@’ (apostrophe), and ‘:’ (colon).

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 – It will be used in the Server and Client module, where each Channel class will have it's own "name", a list of Prefix objects which will be used to mix and match the Sender and Receivers, and a queue of ChatMsg objects which will store the messages received.

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

2)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.

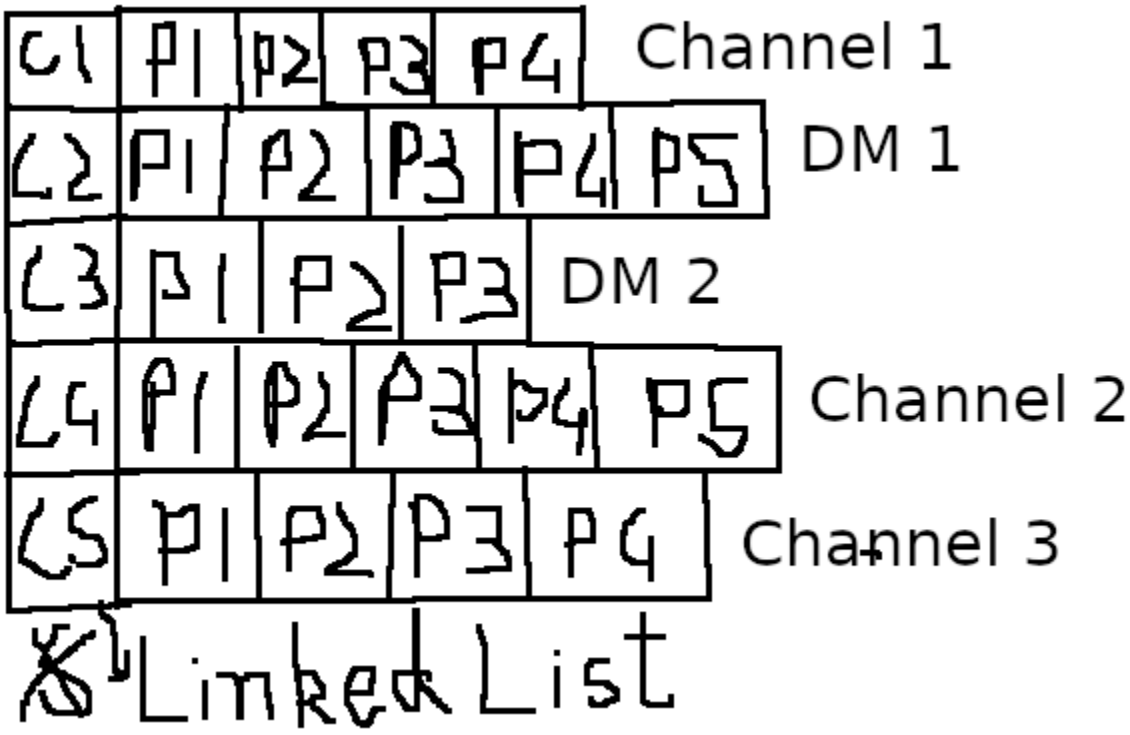
3)Server Module

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

i) ChannelList class, which is used to iterate over the Channel objects 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.



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.