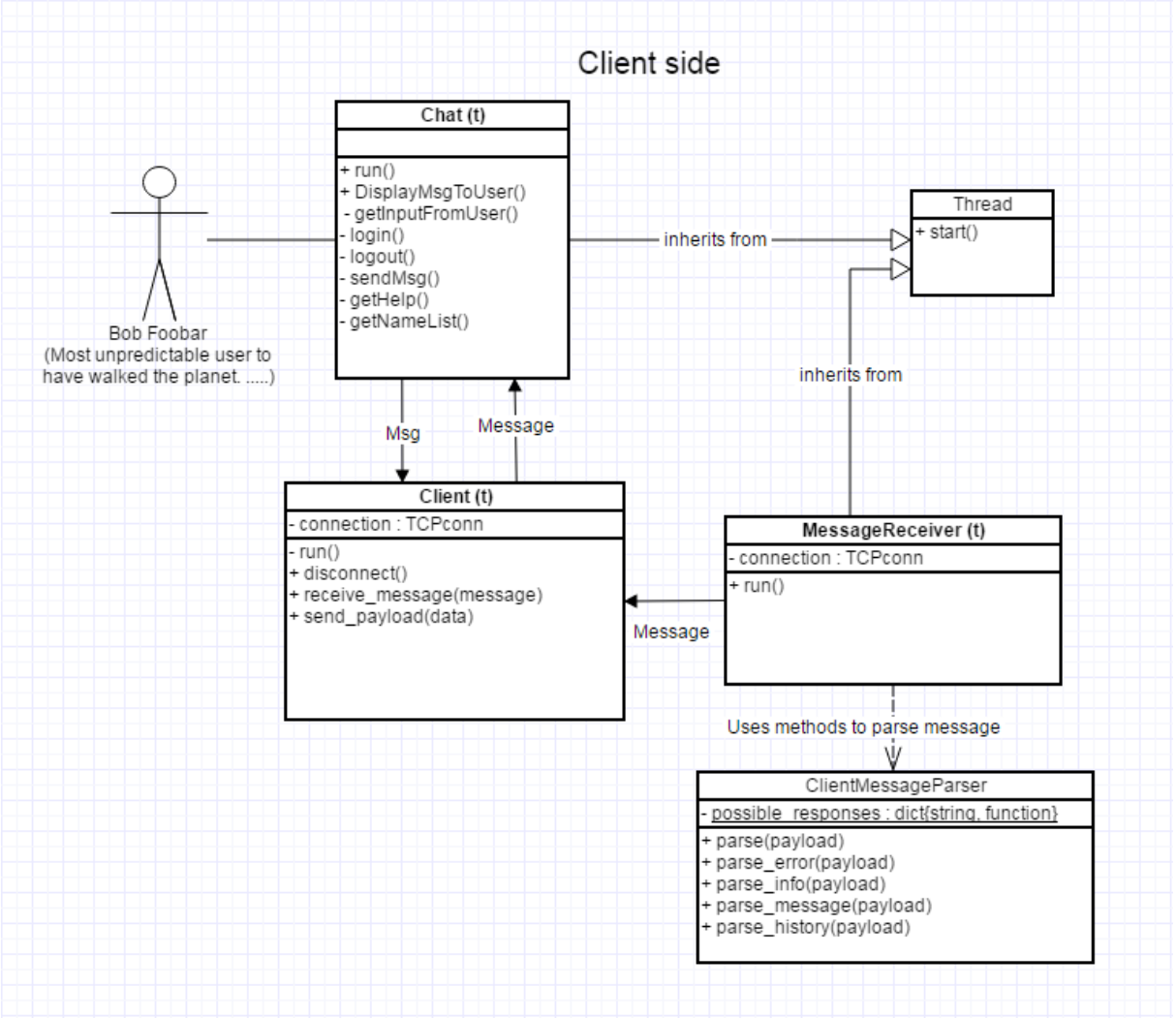
KTN1 – Design



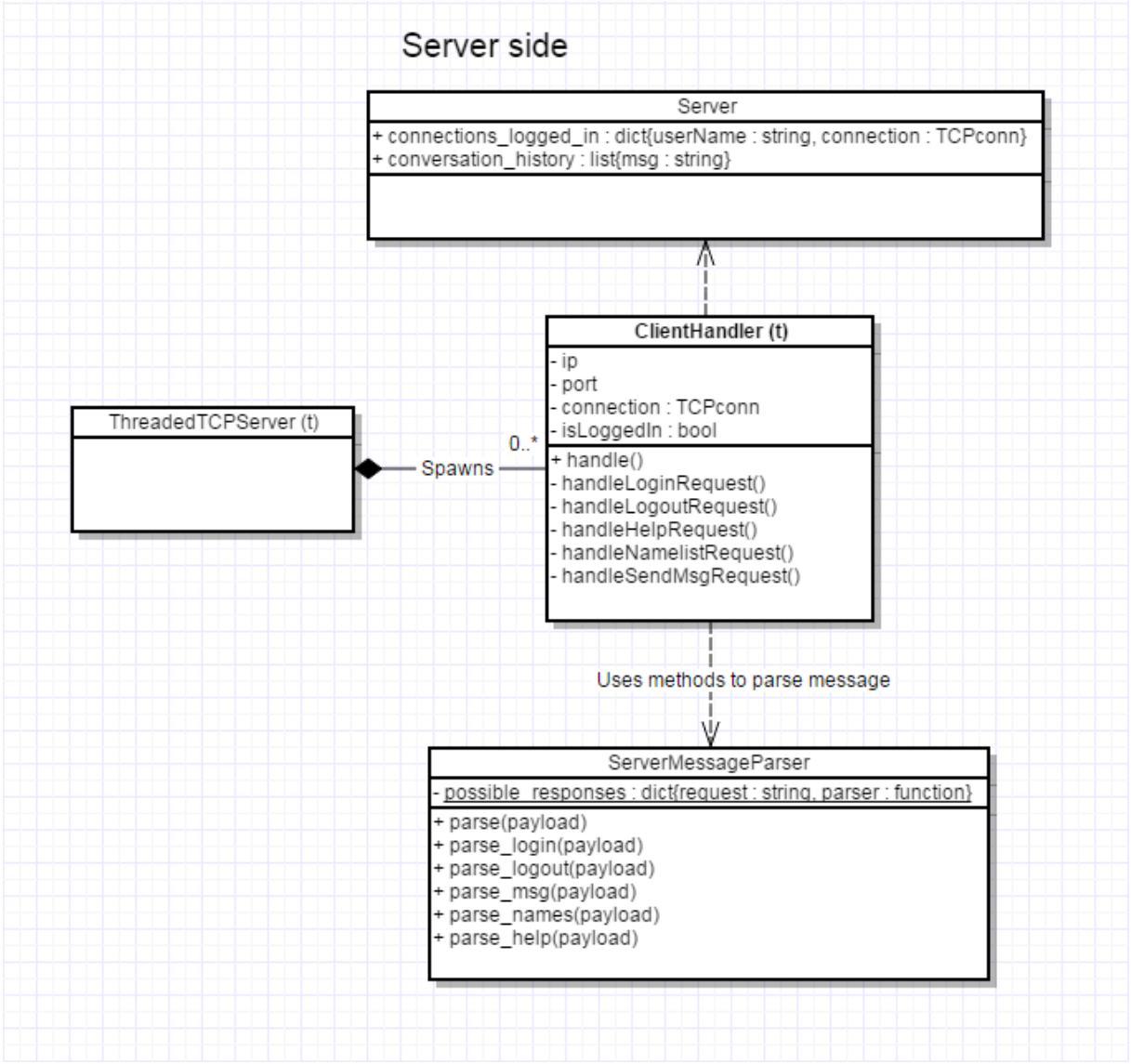
Vi tenker å benytte den utleverte skeleton koden. I klassediagrammene har vi lagt inn «(t)» bak klassenavnet til de klassene som vil opprette tråder.

På klient siden vil vi ha en klasse (Chat) som arver Thread klassen på samme måte som MessageReceiver. Når klienten starter vil en liste over mulige kommandoer og litt info om hvordan å bruke chat'te programmet vises. getInputFromUser() vil lese inn tekst fra brukeren gjennom standard I/O, den vil sjekke om strengen er en kommando og i så fall vil den bruke en av funksjonene login(), logout(), mm. til å lage en melding som den vil sende til serveren ved å kalle funksjonen send_payload(data) til Client klassen. Hvis strengen fra brukeren ikke blir tolket som en kommando vil den bli sendt ved hjelp av sendMsg() funksjonen som kaller send_payload(data) til Client klassen.

MessageReceiver klassen prøver å lese fra receive socketen hele tiden og hvis det har kommet en melding bruker den funksjoner fra ClientMessageParser før den kaller

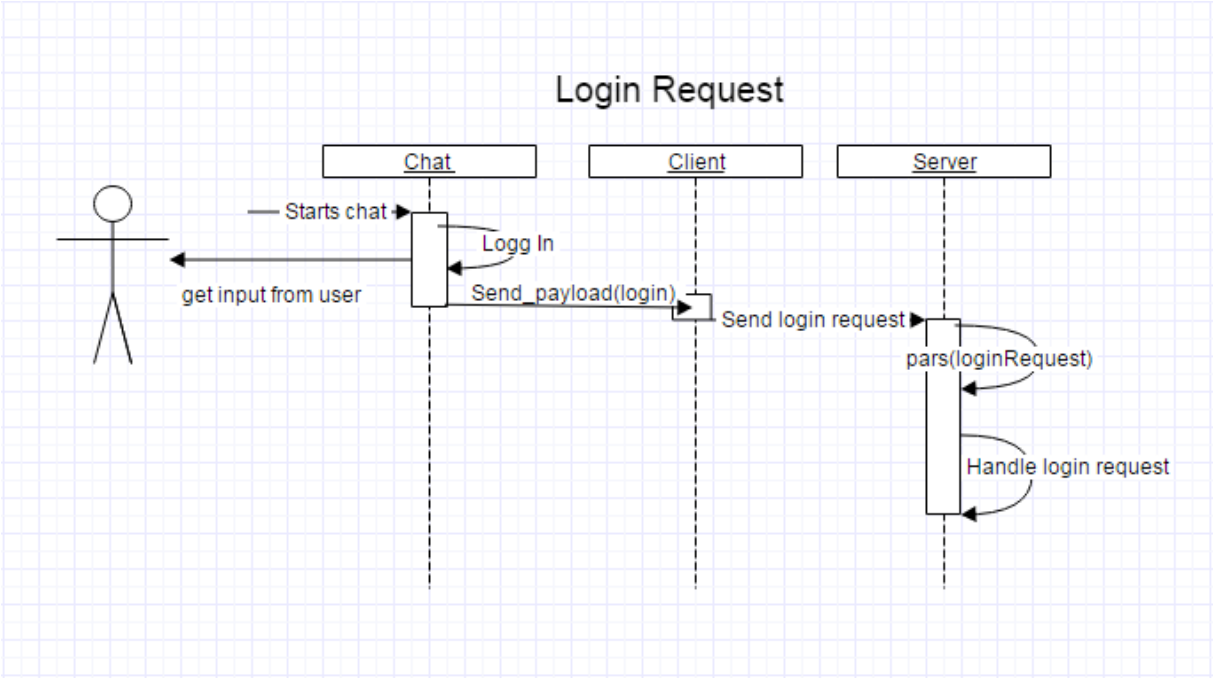
receive_message(message) til Client klassen som igjen kaller DisplayMsgToUser() til Chat klassen som sender innholdet i meldingen til standard output.

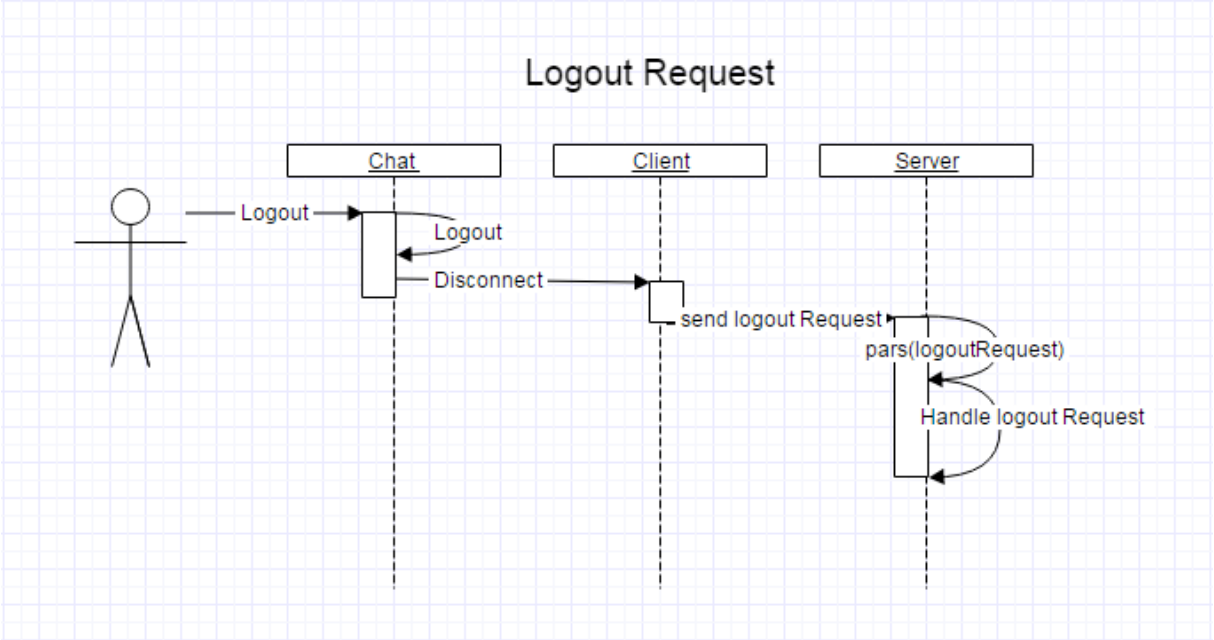
På server siden vil ThreadedTCPServer objektet/klassen opprette nye ClientHandler objekter for hver TCP forbindelse som blir opprettet. Hver ClientHandler tråd vil ta imot meldinger fra en klient, benytte funksjoner fra ServerMessageParser til å dekode meldingen før den vil bruke en av handle…Request() funksjonene til å utføre passende handlinger. F.eks. vil handleNamelistRequest() og handleHelpRequest() sende passende respons tilbake til den klienten som ba serveren om noe. Og hvis klienten ber om å sende en melding til alle de andre klientene vil handleSendMsgRequest() ta meldingen fra klienten og sende den til alle de andre klientene som er logget på serveren. Det globale dictionary'et connections_logged_in blir brukt til å holde styr på hvem som er logget inn til en hver tid.

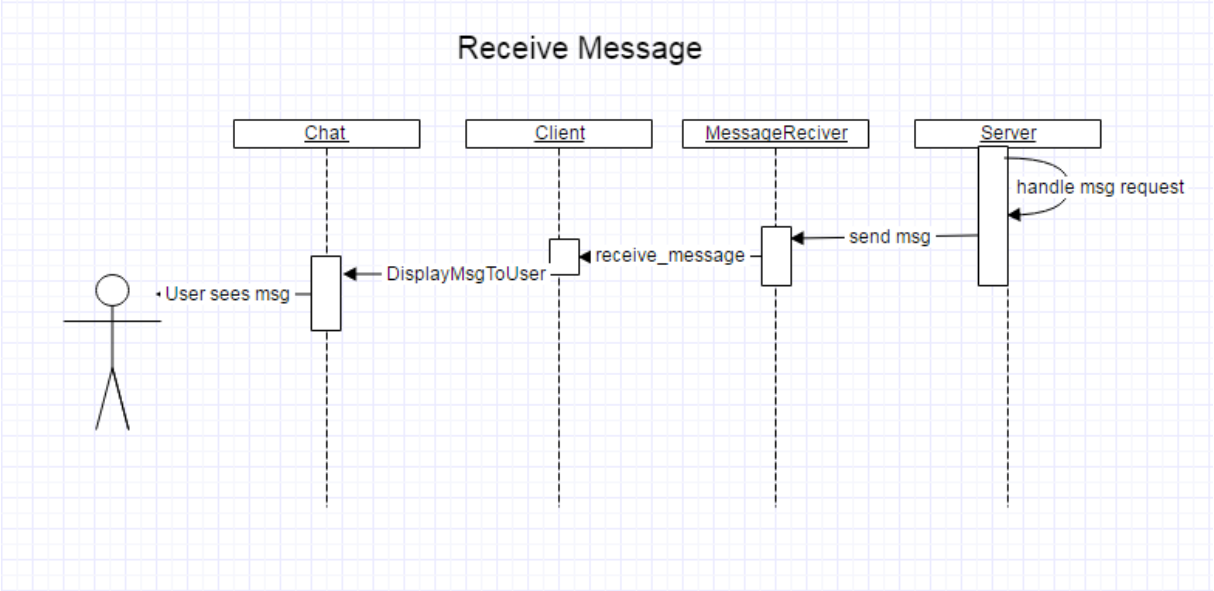


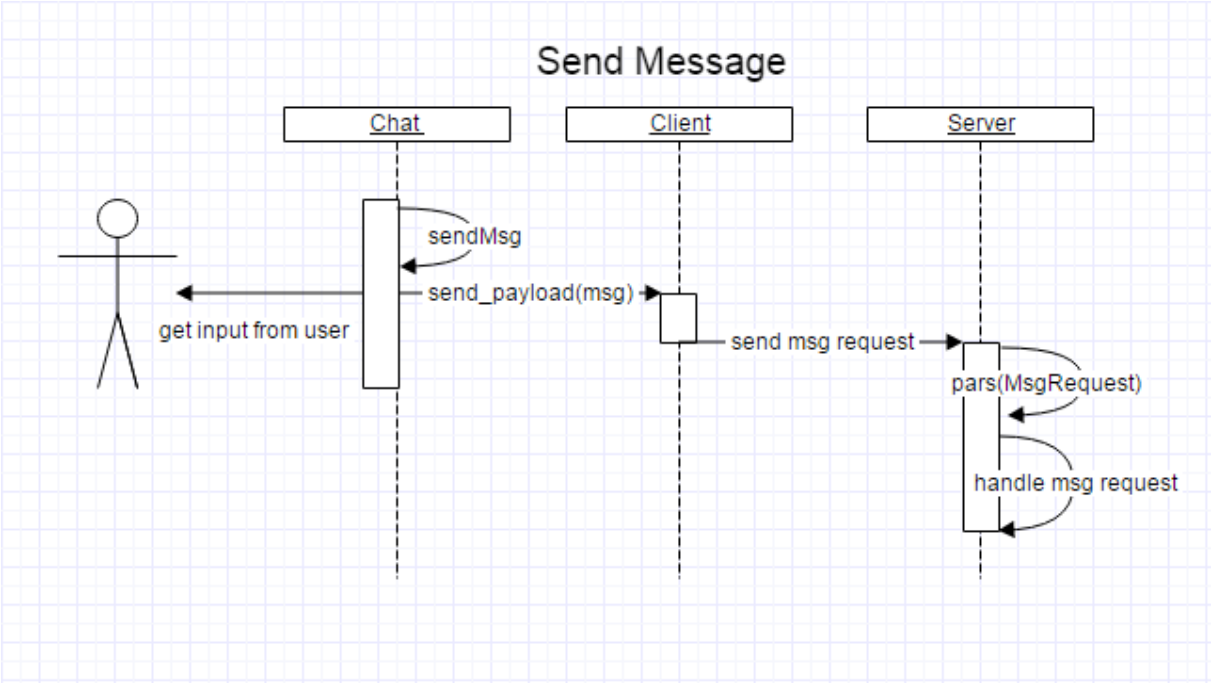
En klient må være logget inn for å kunne sende og motta meldinger. Serveren vil lagre alle chat'te meldinger som er mottatt i den globale listen conversation_history og sende denne listen til en klient hvis klienten spør om det.

Under er det tegnet fire sekvensdiagram som viser hvordan bruker, klient og server skal samhandle for å til sammen utgjøre en chat'te tjeneste.






Receive Message
Chat
Client
MessageReciver
Server
handle msg request
send msg
receive_message
DisplayMsgToUser
User sees msg


Send Message
Chat
Client
Server
sendMsg
send_payload(msg)
get input from user
send msg request
pars(MsgRequest)
handle msg request