Gruppuppgift DA343A VT22
Grupp 14

Medlemmar
Julia Tadic
Omar Ayoubi

Innehållsförteckning

Arbetsbeskrivning

Rent allmänt så har vi båda hjälpts åt väldigt mycket under hela utvecklingsprocessen och vid behov bytt av varandra när vi blivit blinda av att stirra på samma problem allt för länge. Detta har dels gjort att vi båda förstår koden bättre i helhet, dels för att kunna granska varandras arbete och se till att kraven som ställs följs.

Julia Tadic
Julia ansvarade till stor del för det grafiska användargränssnittet mer specifikt design och funktionalitet. Klasser som skrevs utav Julia inkluderar ChatGUI och inre klass (Receivers), BroadCastGUI, TrafficGUI, EntryGUI, stora delar av ChatToGUIInfo, User, MessageLogger, Message, ContactLog förutom det så har Julia även utökat både ChatServer och ChatClient med funktionalitet som stödjer GUI komponenter. Julia ansvarade till stor del för den kontinuerliga dokumentationen i form utav klassdiagram. Julia ansvarade för sekvensdiagrammen vad:

- som sker på serversidan när en klient ansluter till systemet (fall 1)
- som sker i klienten då servern skickar en uppdatering av anslutna användare (fall 4)

Omar Ayoubi
Omar ansvarade till stor del för att kommunikationen mellan server och klient skulle fungera. Klasser som skrevs utav Omar inkluderar ChatServer, ChatClient, Clients, UnsentMessages och stora delar till ChatToGUIInfo. Hade också uppgiften att specifikt felsöka kontinuerligt genom utvecklingsprocessen och säkerhetsställa att nya implementeringar inte rubbat tidigare kods funktionalitet. Detta med ett generöst antal testprogram. Omar ansvarade för sekvensdiagrammen för vad:

- som sker på klientsidan när en användare startar klienten. (fall 2)
- som sker på servern när en klient skickar ett meddelande. (fall 3)

Instruktioner för programstart

Extrahera zippad fil till lämplig katalog men se till att öppna den extraherade mappen i Intellij och inte katalogen som innehåller den extraherade mappen.

Fil med servertrafik, fil som sparar kontakter och flera andra filer med exempelbilder för testning ska finnas i projektets ”files” mapp.

För enkel uppstart av testprogram finns en compound run-konfiguration vid namnet completeTest vilket startar ett serverprogram och två stycken klientprogram vid behov av ytterligare klienter kan så kan mainClientTest exekveras flera gånger oavbrutet. Annars starta programmet mainServerTest och sedan mainClientTest önskat antal gånger.

Klient
Klient GUI visar online lista och kontaktlista, dubbelklicka på en användare i online listan för att lägga till denne användare som kontakt, dubbelklicka en användare i kontaktlistan för att ta bort denne användare som kontakt. Klicka på mottagare användar-ikon för att lägga till flera mottagare till ditt meddelande

Server
Server GUI visar alla meddelanden i en lista vid uppstart och uppdateras kontinuerligt, längst ner till vänster kan man välja att se specifik trafik. Glöm inte att mata in datum i rätt format. Dubbelklick på specifikt meddelande ger meddelande-specifikationer där eventuella bilder kan visas genom ytligare ett dubbelklick.

Systembeskrivning

Vårt system baseras på att anslutna klienter vid alla tidpunkter känner till de andra användarna på samma server detta genom att uppdatera alla klienter med en lista varje gång en klient kopplar upp eller ner sig och mellan dessa händelser skicka och hantera inkommande meddelanden. All kommunikation mellan server och klient sker med hjälp av serialiserade strömmar.

Vid uppstart av klienten så bes användaren om information för att
klienten ska kunna skapa en ny användare. En inre klass konstruerad för att köras på egen tråd hanterar kommunikationen till servern, denna nya användare skickas sedan till servern. Därefter inväntar den inre klassen tills ett avbrott på inläsning av objekt. Det inlästa objektet kontrolleras för om det är ett meddelande eller en lista. Om det är en lista så uppdateras online listan enligt denna nya lista. Ett litet pop-up fönster informerar användare händelsen.
Ett ankommet meddelande lagras med anknytning till sändare.
Detta görs för att kunna skifta mellan olika konversationer och komma tillbaka till samma meddelande historik.

Vid uppstart av servern visas ett GUI för trafikgranskning. Server klassen innehåller två inre klasser en klientlyssnare, ClientListener och en inre klass Client för representation av klient på serversidan. Allra först börjar klientlyssnaren att lyssna efter uppkopplingar på en egen tråd så fort denna hittat en uppkoppling så skapas en ny instans av Client med hjälp av denna uppkoppling på en egen ny tråd. Det första Client instanser gör är att invänta ett användare objekt från klienten. Efter att servern tagit emot användarobjektet kan den nu associera denna med Client instansen vilket hanterar all inkommande och utgående data. Varje gång detta sker så skickas en ny online lista till alla anslutna klienter. Varje gång en klient tappar anslutning till systemet så plockas denna ur kännedom och även Client instansen associerad till denna användare och en ny online lista skickas till alla uppkopplade användare. På ett inkommit meddelande kontrolleras mottagarlistan och servern ser till att vidarebefordra meddelandet till antingen klienter ifall en klient med användare på mottagarlistan är online eller så sparas detta meddelande på servern tills att en klient med samma användare kopplar upp sig igen, därefter ser servern till att skicka ut lagrade meddelanden till denne användare.

Du kan med vårt chatprogram dela med dig roliga bilder med vänner eller familj, eller båda, samtidigt! Lägga till andra användare på kontaktlistan. Skriva till både online och off-lineanvändare. Det går att hålla reda på all eller specifik trafik genom servern för den som väljer att vara host och se ifall någon pratar bakom ryggen på dig. Ett stort försök har lagts på att försöka göra programmet användarvänligt.

Beskrivning av klasserna

User klassen innehåller den data som definierar en användare i systemet, med ett användarnamn och en användarbild kan man lätt skilja på olika användare i systemet. Denna klassen används och känns till av både klient och serversidan.

Message klassen innehåller den data som utgör ett meddelande i systemet. Klassen innehåller All nödvändiga data för hantering av meddelandet så som en sändare (User) och en Mottagarlista (ArrayList <User>) och plats för endast text och eller bild. Denna klass känns används och känns till av både klient och serversidan.

UnsentMessages klassen är den klass som hanterar meddelanden som ännu ej kan levereras på grund av att mottagaren är inaktiv på serven just nu. I och med att flera trådar interagerar med denna datasamling, dvs lägger till och tar bort från denna klass inre variabel. så synkroniseras metodanropen i denna klass. De olevererade meddelandena sparas i HashMap <User, LinkedList <Message>>. Denna klass används enbart av serversidan.

MessageLogger klassen hanterar inläsning och skrivning av alla meddelanden som någonsin skickats på denna server. Vid varje uppstart av servern läser denna klass in alla meddelanden (ArrayList <Message>) och tillåter klienter att förlänga denna lista under exekvering. Förlängning av lista sker med ett synkroniserat anrop. Denna klass används enbart av serversidan.

Clients klassen används för att hålla koll på olika Client objekt som är associerade med en användare. Denna klass gör det möjligt för servern att mappa ut olika trådar och veta vilken tråd som ska utföra vad. Klassen innehåller en HashMap <User, Client>. Synkroniserade metoder för tillägg och borttagning följer med. Denna klass används enbart av serversidan.

ChatServer klassen innehåller i stort sett all logik för serversidan. den innehåller variabler av typen, Clients, MessageLogger och UnsentMessages. I stort sett styr denna alla ingående datastruktur och anropar lämpliga metoder vid lämpliga tillfällen. Ser till så att alla uppkopplade klienter förses med lista varje gång antalet användare ändras och meddelanden till den det angår. Vid avstängning ser servern till att alla skickade meddelanden sparas på hårddisken. Innehåller även lite viktiga komponenter för GUI.

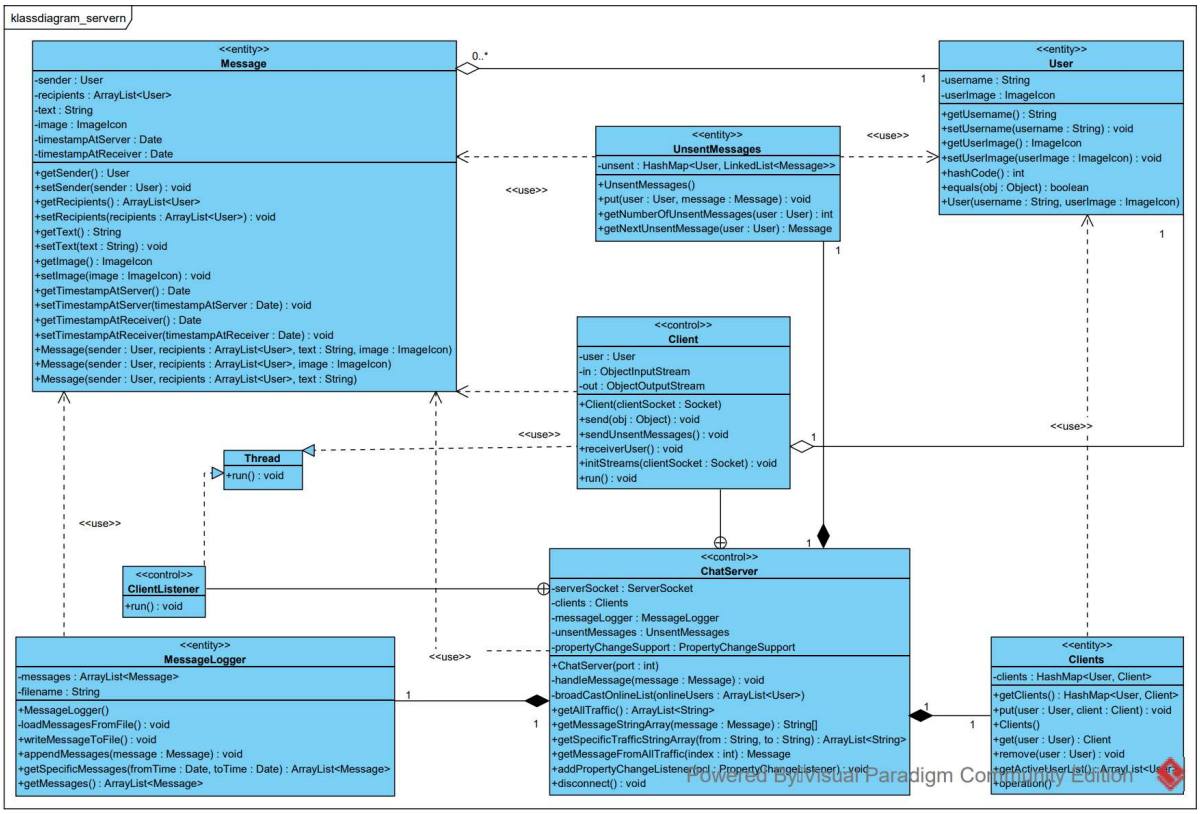
ContactLog klassen används för att hålla koll på sparade kontakter och läser in dem för varje gång den instansieras. Klassen håller koll på en användares kontakter genom en HashMap <User, ArrayList <User>>. Klassen innehåller metoder för att manipulera kontaktlistan och skriva över den till hårddisk. Denna klass används enbart av klientsidan. Och en instansvariabel för ChatClient.

ChatToGUIInfo klassen agerar främst som en datastruktur som sparar relevant information för GUI men även lite annat som annars bara tar plats i controller klassen, denna innehåller klientens användare, den valda användaren i GUI och online listor och annat som är relevant. Den sparar även de konversationer som skett medans klienten är igång så att an enkelt kan skifta mellan chatfönster utan att förlora data. Denna klass används enbart av klientsidan. Och en instansvariabel för ChatClient.

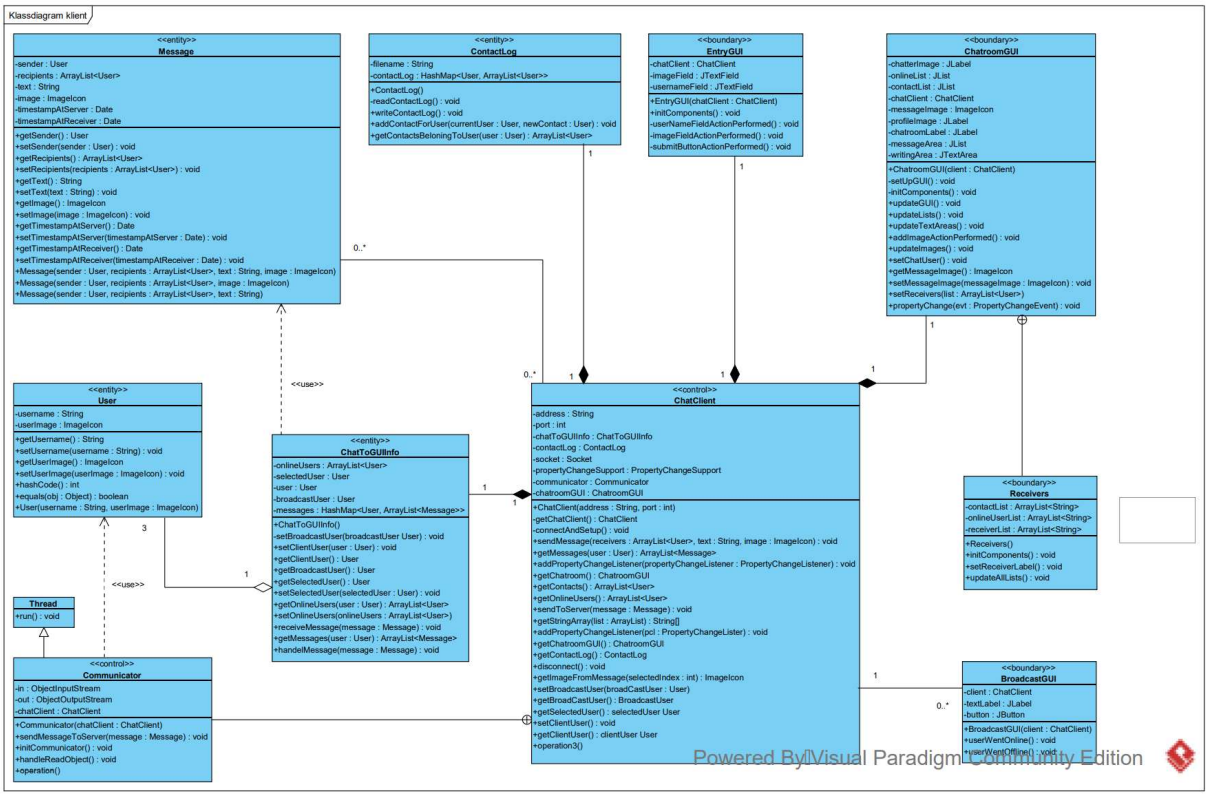
ChatClient klassen innehåller nästan all logik som sker på klientsidan alltifrån att konstruera och skicka ett nya meddelande inmatade i GUI till att ta emot inlästa objekt och avgöra ifall det är en ny online lista eller ett inkommit meddelande. Vid avslut sparas alla aktuella kontakter på hårddisken

Klassdiagram

Klassdiagram server



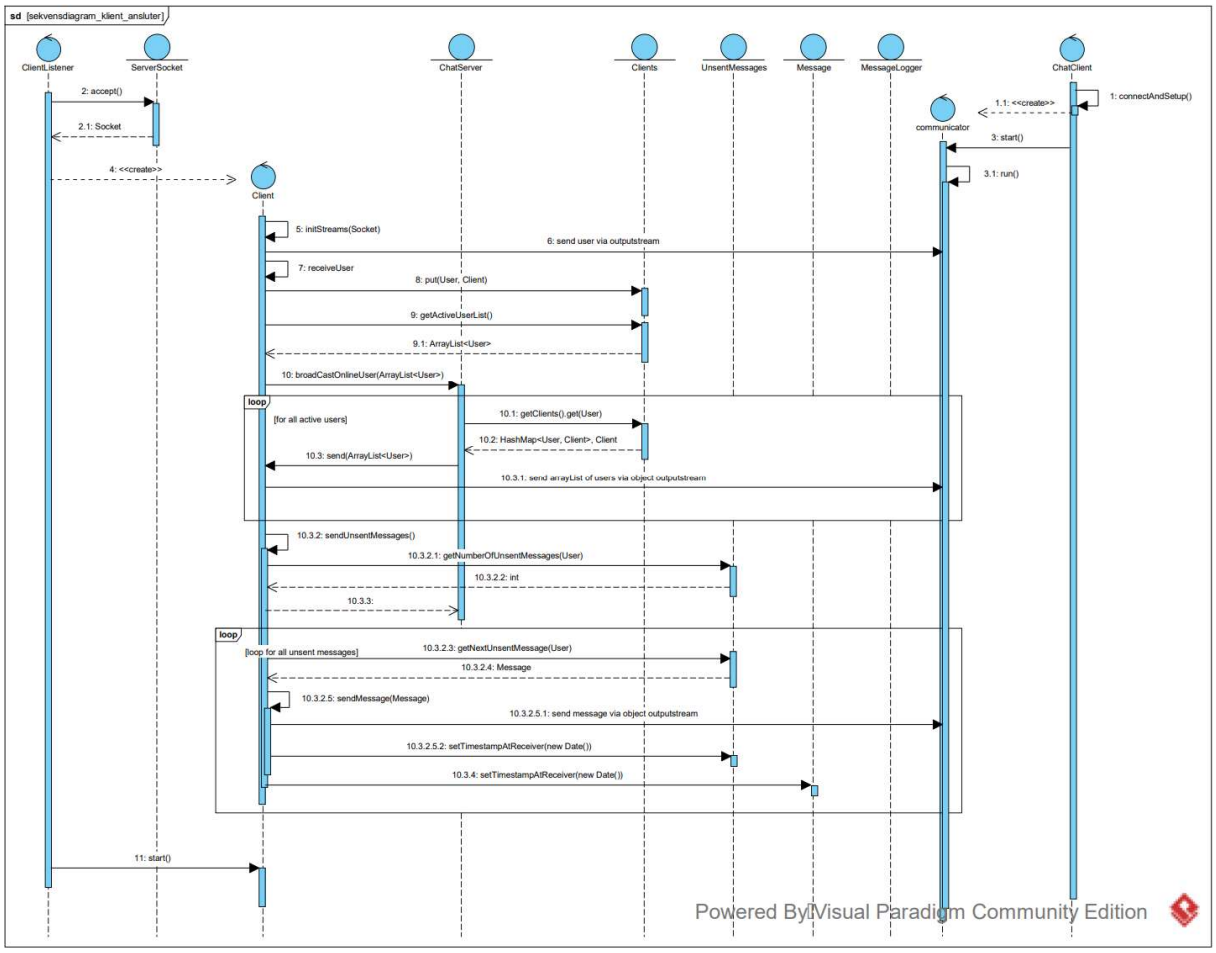
Klassdiagram klient



Sekvensdiagram

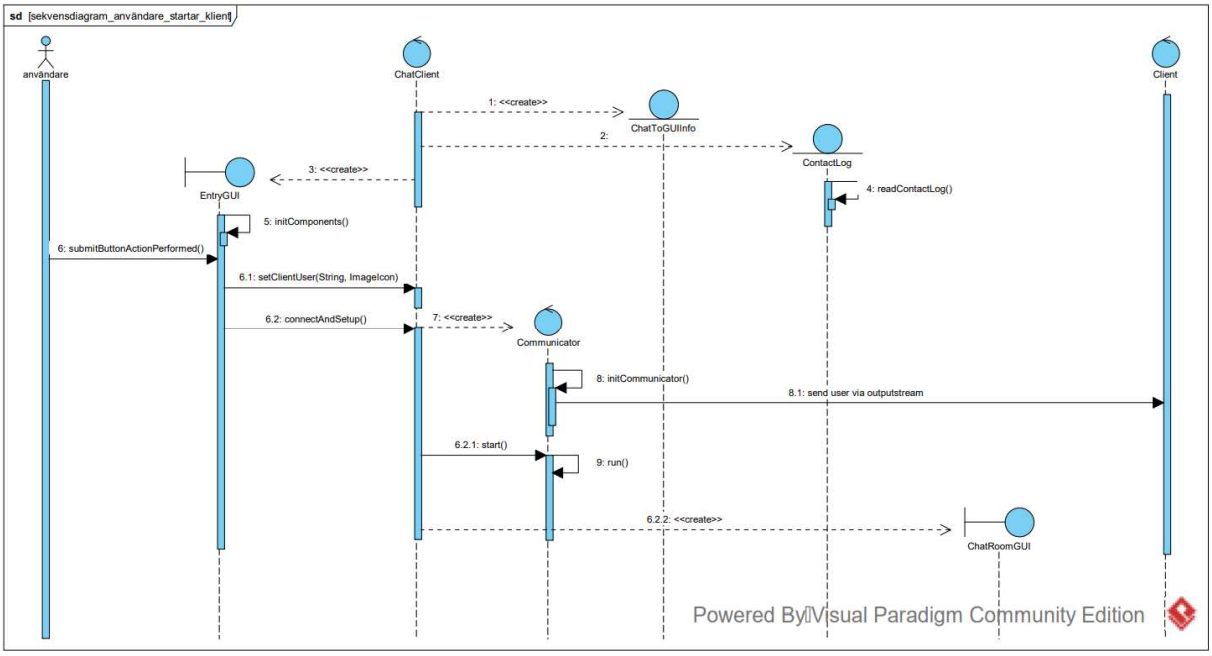
Sekvensdiagram: Klient ansluter sig till server (server sida)

Ansvarig: Julia Tadic



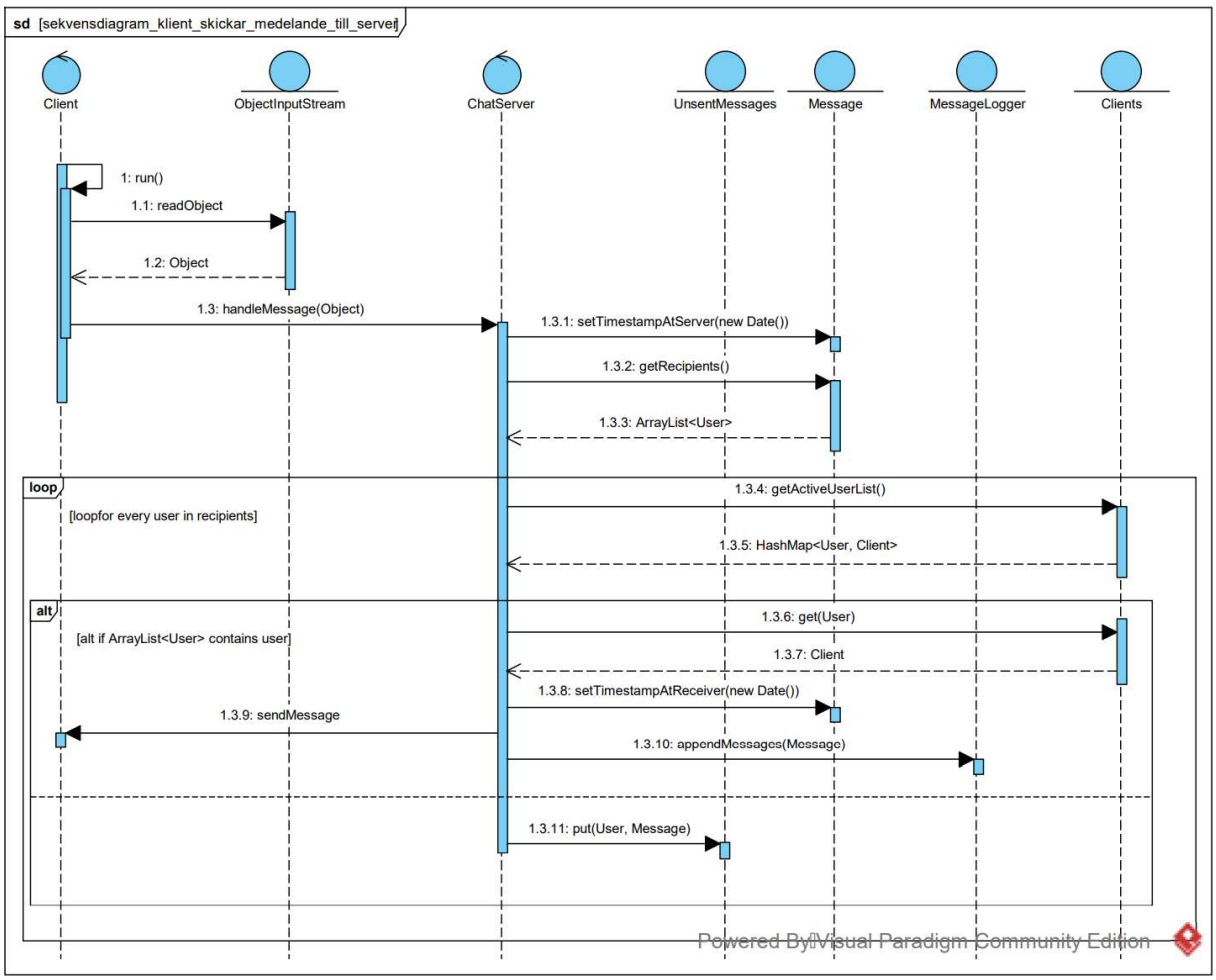
Sekvensdiagram: Användare starta klient

Ansvarig: Omar Ayoubi



Sekvensdiagram: Klient skickar meddelande

Ansvarig: Omar Ayoubi



Sekvensdiagram: uppdatering av anslutna användare (klient sida)

Ansvarig: Julia Tadic