Chattforum med TCP

Projekt i kursen EDA095 - Nätverksprogrammering
19 maj 2017

Amanda Eliasson, Carl Johan Balck,
Niklas Ovnell & Lisa Silfversten
Handledare: Pierre Nugues

Innehåll

1 Bakgrund **ii**

2 Kravspecifikation **ii**

3 Modell **ii**

3.1 Server ii

3.2 Client iii

3.3 GUI iv

3.4 Util iv

4 Användarhandledning **iv**

5 Utvärdering **v**

6 Programlistor **v**

1 Bakgrund

Den här rapporten beskriver ett projekt i kursen Nätverksprogrammering. Målet med projektuppgiften är att tillämpa de tekniker som ingår i kursmaterialet. Den projektuppgift vi valt är att utforma ett chattsystem för multipla användare. Chattsystemet är uppbygt av en multitrådad TCP server och TCP klienter. All kod är skriven i java och för kommunikation användes javas Socket- respektive ServerSocket-klasser.

2 Kravspecifikation

I början av projektet sattes en kravspecifikation av systemets funktioner upp. Kraven visas nedan i en punktlista.

- Systemet ska hantera flera användare
- Användaren ska kunna skicka textmeddelanden i en gruppchatt till samtliga användare (broadcast)
- Användaren ska kunna skicka privata textmeddelanden till specifika användare
- Användaren ska kunna skicka bilder i gruppchatten
- Användaren ska kunna svara på både broadcast meddelanden och privata meddelanden
- De användare som är online i chatten ska visas i en användarlista
- När användaren startar chatten ska hen kunna välja ett användarnamn
- Användarnamnet får inte vara mindre än tre bokstäver
- Användarnamnet får inte innehålla space eller hakparanteser
- Användarnamnet måste vara unikt bland deltagarna i chatten.
- När ett icke giltigt användarnamn skrivs in ska ett felmeddlenade visas

3 Modell

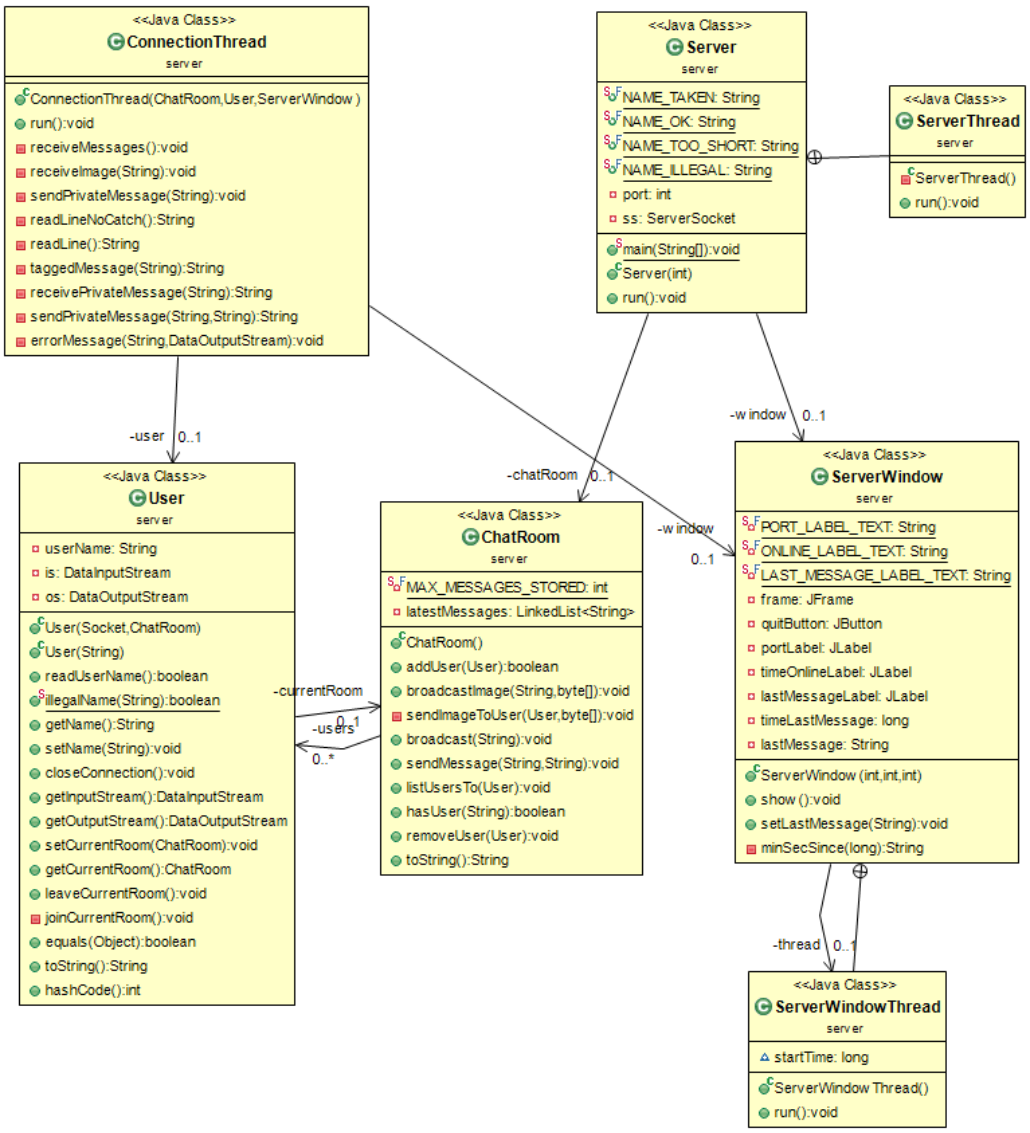
Nedan presenteras de viktigaste klasserna i modellen.

3.1 Server

Servern består att ett chattrum, en Serversocket och ett fönster som visar information om servern. För varje användare som ansluter till serverns körs en tråd som representeras av en klass ConnectionThread. Varje tråd tilldelas ett chattrum, en användare och ett fönster.

En användare representeras av en Socket och ett chattrum. Användaren representeras i chattfönstret av ett användarnamn.

När en användare ansluter sig till servern via klienten skapas en instans av User-klassen. I User finns Streams, en Reader och en Writer för att kommunicara med användaren. Användaren läggs in i chattrummet (ChatRoom) och en ny tråd skapas för att hantera användarens meddelanden.



Figur 1: UML-diagram för servern. I diagrammet visas inte hur ConnectionThread-objekt hör ihop med resten; de skapas när en användare ansluter till servern.

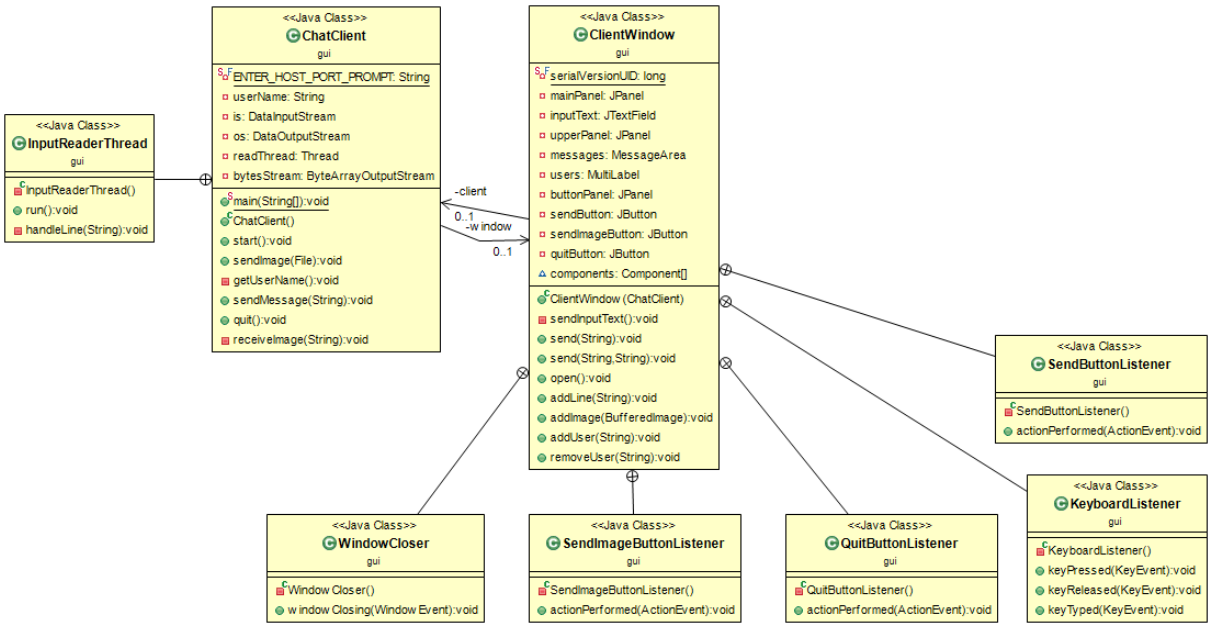
Varje meddelande från klienten börjar med några tecken som visar vad användaren vill göra (t. ex. Skicka meddelande till alla, privat meddelande eller koppla ner). ConnectionThread-objektet läser dessa och ser till att användarens begäran hanteras, med hjälp av chattrummet om något meddelande ska skickas.

Sambandet mellan klasserna hos servern visas i Figur 1.

3.2 Client

Varje klient (ChatClient) består av Streams, en Writer och en Reader som kommunicerar med servern, ett fönster (ClientWindow) där användaren kan ange kommando och se meddelande från andra användare, och en tråd (InputReaderThread) som kontinuerligt läser meddelanden från servern. När användaren startar klienten bes de ange host och port för servern, följs av ett namn. Efter det kommer ett ClientWindow upp som hanterar resten av interaktionen med användaren.

UML-diagrammet för klienten (utan klasser som hanterar användargränssnitt) visas i Figur 2.



Figur 2: UML-diagram för klienten.

3.3 GUI

ClientWindow är fönstret som användaren ser. Klassen innehåller subklasser som är ActionListener till knapparna, en WindowAdapter som ser till att klienten stängs korrekt och en KeyListener som gör att användaren kan skicka meddelande eller stänga klienten med tangentbordet. Fönstret skickar kommando till klienten med hjälp av knapparna. Placeringen av komponenterna i fönstret bestämdes med hjälp av Java:s Layouts.

Meddelanderutan och användarlistan består av subklasser till MultiLabel. Dessa ser till att de senaste meddelandena visas respektive att alla (upp till en övre gräns) användare som är anslutna syns.

Subklasserna MessageArea och UserListArea till MultiLabel ser till att meddelanderutan och användarlistan får rätt beteende. Till exempel ska en bild visas på mer än en rad bland meddelandena, och användare ska kunna tas bort från användarlistan.

3.4 Util

Paketet util har klasser som innehåller metoder för att göra jobbet lättare för de andra klasserna. I klassen Communication finns konstanter som definierar vad meddelanden betyder. Den innehåller också metoder för att skicka meddelanden till en användare, eller en writer som används på flera håll i både klienten och servern.

4 Användarhandledning

För att starta chattsystemet måste användaren först köra filen Server.java. Denna klass kör igång servern och visar en ruta med information om vilken server som körs och hur länge servern varit igång. Därefter startar användaren klassen ChatClient.java och kör programmet via localhost genom att klicka OK på rutan där det ges möjlighet att skriva in host och port. Användaren skriver sedan in sitt namn och klickar OK för att sedan skickas vidare till chatten. För att skicka gruppmeddelanden skickar användaren

sitt meddelande genom att skriva text i textrutan och sedan trycka på Send. För att skicka privata meddelanden klickar användaren först på namnet på den person som hen vill skicka till och skriver därefter in meddelandet i textrutan och klickar på Send. För att lämna chatten trycker användaren på Quit.

5 Utvärdering

Det slutgiltiga chattforumet uppfyller samtliga krav i kravspecifikationen och arbetsprocessen och resultatet blev som vi föreställt oss. Programmet fungerar på ett bra sätt även om en del designmässiga förväntningar ej uppfyllts på grund av tidsbrist. Om vi fått mer tid till projektet hade vi utökat vårt program med fler funktioner samt gjort en snyggare design av användargränsnittet.

Vi valde att inte genomföra övergången mellan TCP till HTTP då vi redan imlempenterat samtliga funktioner och ansåg att detta blev för mycket arbete. Detta beslut hade dock sett annorlunda ut om vi fått möjlighet att göra om uppgiften. Att byta till HTTP hade förbättrat vårt program och genom att göra övergången i början av projektet hade den blivit enklare att genomföra.

Övrigt framtida arbete i projektet skulle vara att utöka chatten med olika chattrum.

Alla gruppmedlemmar tycker att upplägget av projektuppgiften varit bra. Särskilt bra är att det ges möjlighet att själv bestämma vad för typ av uppgift som ska genomföras.

6 Programlistor

Referens till källkod: https://github.com/Balck92/eda095_projekt.git