Chatbot

Introduktion

I denne opgave har jeg lavet en chatbot med en server, som kan håndtere flere clients vha. threads.
Man kan joine chatten, sende beskeder til hinanden, printe en liste ud over aktive klienter samt skrive en besked til alle aktive brugere på chatten. Derudover returneres der J_OK til klienten når der logges på.

Afgrænsning

Jeg er blevet nødsaget til at modificere protokollen lidt, da jeg ikke evnede at overholde den givne protokol. Derfor ser den nye protokol ud således:
JOIN <<ipadresse>>:<<port>>
Indtast brugernavn
Derefter sendes beskeder til brugere således (brugernavn er modtagerens brugernavn):
<<Besked>>:<<brugernavn>>
For at sende en broad message til alle aktive, erstattes brugernavn med all:
<<Besked>>:all
Derudover kan der printes en liste over aktive brugere:
active
Og til sidst logges der ud af programmet:
QUIT
Socket til klienten lukkes, og der skrives goodbye. Serveren skriver hvem der disconnectede.

Samtidig har jeg afgrænset mig ud af heartbeat-funktionen.
Derudover har jeg heller ikke benyttet en thread-pool til at limite antallet af brugere, men har derimod fastsat størrelsen af min vector, som holder aktive clients, således der ikke kan være mere end 5.
Jeg har heller ikke taget forbehold for flere brugere med samme navn, altså kan der godt være flere brugere med samme brugernavn. Dog vil de stadig være forskellige clients, selvom brugernavnene er ens. Dette kunne gøres ved at tjekke username på de eksisterende objekter i min vector som holder clients, og se om de er equal til den nye client.
Der er lavet forarbejdet til at implementere en fuldt funktionel log. En sådan log ville fungere vha. Logger-klassen som ligger i javas bibliotek, og jeg ville bruge en filehandler til at skrive loggen ud til en txt-fil.
Der tages heller ikke højde for hvis der skrives uden for syntaks. Dette kunne fikses ved at lave while-løkker der tester for input, og kun kører resten af programmet, hvis input passer med protokollen. Samtidig kunne der skrives en J_ER hvis ikke det passer.

Design pattern

Jeg har valgt at implementere singleton pattern, ved at gøre server-constructeren private, så der kun kan køre en instans af serveren.
Hvis ikke constructoren er privat, kan alle andre klasser kalde en server (Server server = new Server()), hvilket kan give enorme problemer.
Dette er især smart, hvis man er flere, som koder på samme projekt, da man ved at følge singleton, ikke kan crashe hinanden.

Kodning

I klienten har jeg valgt at have to tråde kørende, da man gerne vil have mulighed for at modtage eller sende beskeder, uden de to nødvendigvis skal have tilknytning, altså man skal kunne sende uden at modtage, eller modtage uden at sende, og det skal ikke være nødvendigt at sende en besked for at modtage og omvendt.
Derudover har jeg faktoreret main-metoden til sin egen klasse, således ip-adressen og porten kan sendes med, når jeg instantierer et klient-objekt.
Jeg har en clienthandler, som håndterer kommunikationen mellem klienten og serveren. Dette er smart, da hver client får sin egen clienthandler-thread, som serveren kan arbejde med. Denne thread opbevares så i en vector, der er statisk, således den kan bearbejdes fra flere klasser. Dette muliggør at clienthandleren vha if statements og for-each loops kan sende de rigtige beskeder til de rigtige brugere. Samtidig slettes objektet automatisk fra min vectorliste, hvis socket for klienten lukkes, altså der logges af.
For at udskrive en liste af aktive klienter, har jeg lavet en toString-metode i clienthandler, som skriver navnet på det givne clienthandler-objekt, altså hvilken klient der er tale om. Derfor kan der laves en for-each løkke som kører samtlige clienthandler-objekter igennem i vectoren, og udskriver denne toString-metode for hvert clienthandler objekt.

Konklusion

Jeg har lavet et chatsystem, som fungerer med flere clients. Dog følger den ikke helt den stillede protokol, men har næsten alle funktioner fra opgavebeskrivelsen. Med lidt mere tid og mere praktisk viden om emnerne, som f.eks. threadpools, ville opgaven have været mere realistisk at løse fuldt ud.