SheepChat

Server och klient

DAT055, Objektorienterade Applikationer
Grupp 7
Fredrik Hansson, 891115-4832, fredha@student.chalmers.se
Robin Braaf, 870710-2053, braaf@student.chalmers.se
Morgan Eklund, 861210-0092, emorgan@student.chalmers.se
Tommy Kindmark, 900505-3419, tommyki@student.chalmers.se
Cristian Troncoso, 910917-5779, troncoso@student.chalmers.se
Andreas Nilsson, 910403-1092, andrn@student.chalmers.se
2012-02-24

Sammanfattning

Rapporten behandlar de mål vi satte upp under projektets första arbetsvecka och hur väl vi har uppnått dem. Utöver det finns det en enkel användarmanual med bilder och text som beskriver hur man enklast som användare/server använder sig av SheepChat. Det finns även UML-diagram som beskriver klassernas relation med varandra och hur koden är uppbyggd.

Syfte med arbetet: Vi ville lära oss om socket och nätverkskommunikation i java.

Uppnådda resultat

Jämförelse med milstolpar

1. Planerat klart och gjort en grov uppdelning av arbetet. Vi försökte arbeta utifrån UML:en, men vi hade inte gjort en chat tidigare. Vi träffade på problem med UML så ändrades UML allt eftersom då fick vi bättre bild över den slutgiltiga koden för projektet.
2. Simpel server och klient som kan kommunicera. Server och klient har ett simpelt GUI. Enkla kommandon och administration hos server (Ex kicka klient)
3. Server kan autentisera klienter. Klienterna har däremot inte olika rättigheter.
4. Loggar i både server och klient. Klara med dokumentation.(idag bam)

Innehållsförteckning

Klassbeskrivningar

Serverklasser

ChatServerProtocol

Motsvarar ett protokoll som processar kommandon som skickats från klienten. Den väntar först på en korrekt autentisering utav varje nyansluten användare, innan dess kan användaren inte använda sig utav några kommandon. Därefter kan till exempel "broadcast" användas för att skicka meddelanden till alla andra anslutna.

ClientConnection

Representerar en TCP-uppkoppling mellan servern och en klient. Den hanterar även information om klienten, såsom ifall den är autentiserad. Om klienten är autentiserad så kommer data som tas emot från klienten att processas utav ChatServerProtocol som i sin tur genomför uppgifter beroende på vad för kommando som skickats.

Logger

Hanterar loggning till fil utav alla händelser på servern, såsom när servern startats/stoppats och när användare loggar in/ut.

Server

Detta är knutpunkten för alla klasser, den innehar kontroll att starta och stoppa systemet samt lyssnar efter nya klienter. När en ny klient försöker ansluta sig så skapas en ny tråd för hantering av dess uppkoppling. Den vidarebefodrar även all information till användargränssnittet.

UserHandler

Hanterar alla användare och deras lösenord. Den sparar automatiskt användarlistan varje gång den har ändrats, dvs. då en ny användare registrerat sig. Den observeras utav Server-klassen i fall att det sker ett skrivfel.

Klientklasser

Client

Client är den centrala klassen i klientprogrammet. Här hanteras alla meddelanden som kommer från servern. Här finns metoder för allt som man vill kunna göra i samband med chatten från klientsidan.

ServerConnection

Denna klassen representerar uppkopplingen till servern när den anslutit. Efter anslutning till servern har skett så har den som huvuduppgift att skicka meddelanden till servern.

ServerListener

Den här klassen hanterar en tråd som lyssnar efter data från servern. All mottagen information vidarebefodras till Client-klassen som i sin tur processar den.

TextHandler
TextHandler har som uppgift att spara chatloggar. Loggarna namnges efter inloggad användare och datum.

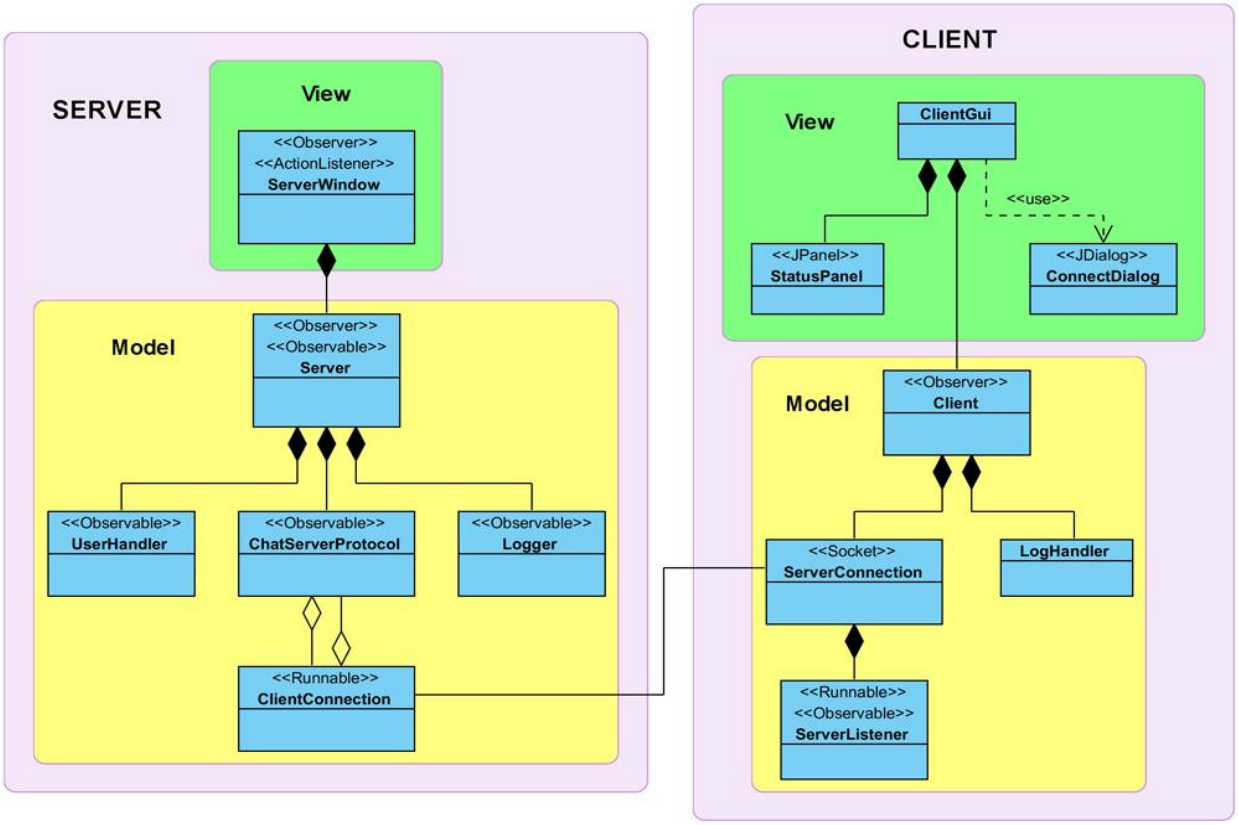
ClientGUI
Skapar ett användargränssnitt samt tar emot information från Client för att sedan skriva ut det.

ConnectDialog
Den här klassen hanterar utseendet på inmatning av användarnamn, lösenord och vilken IP-adress man vill ansluta till.

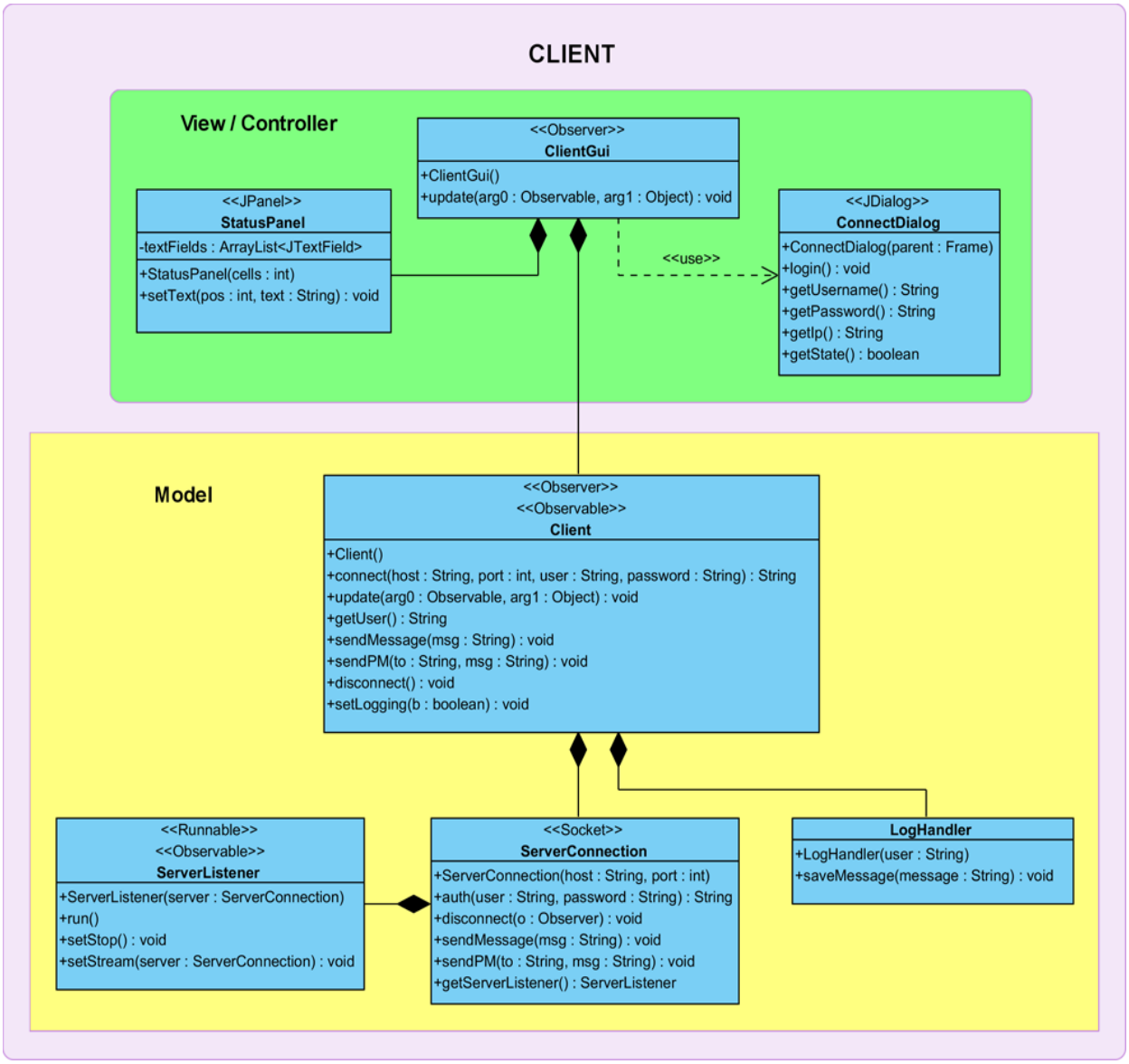
StatusPanel
Är ett statusfält som visar om användaren är inloggad och programmets version.

UML modeller

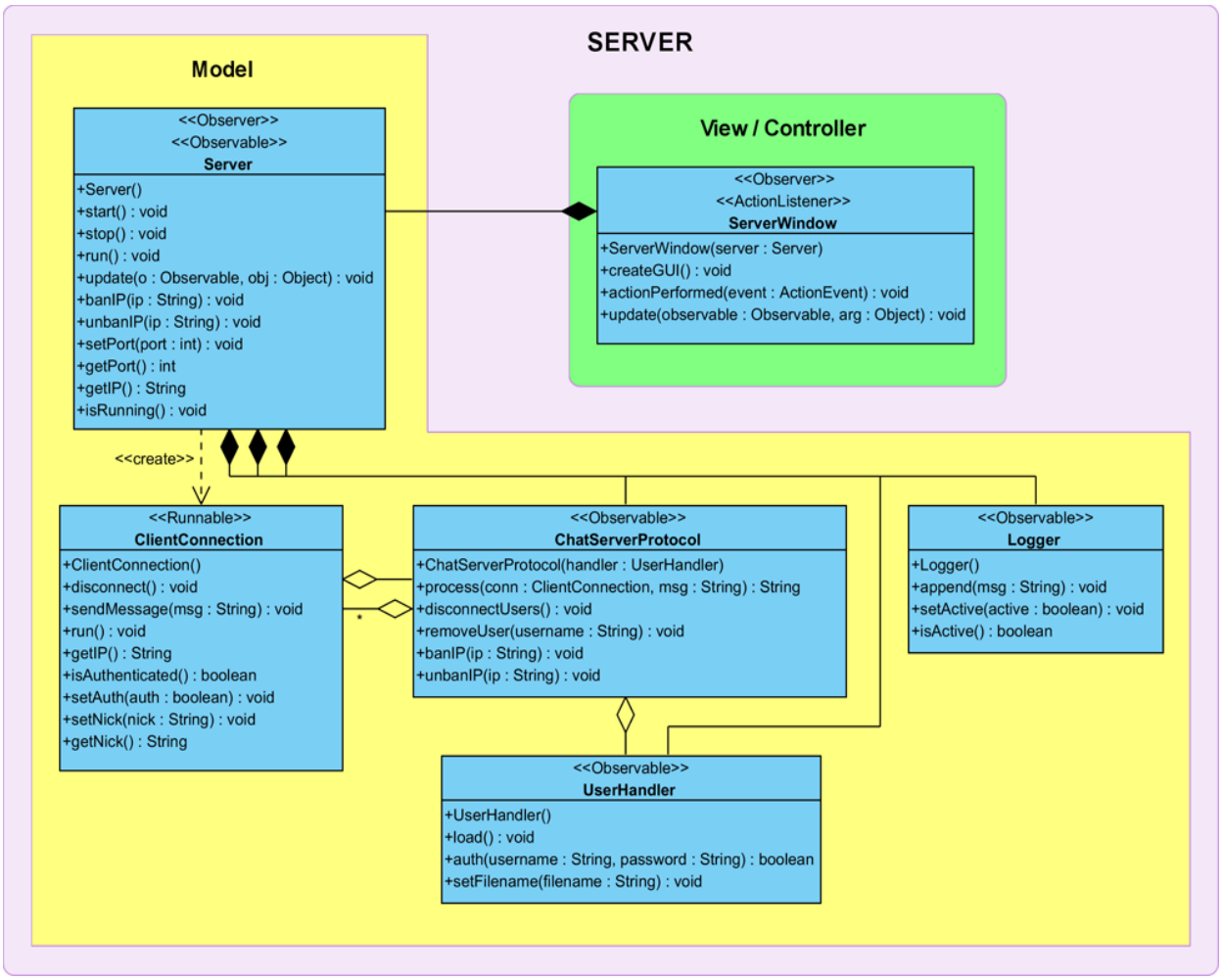
Översikt



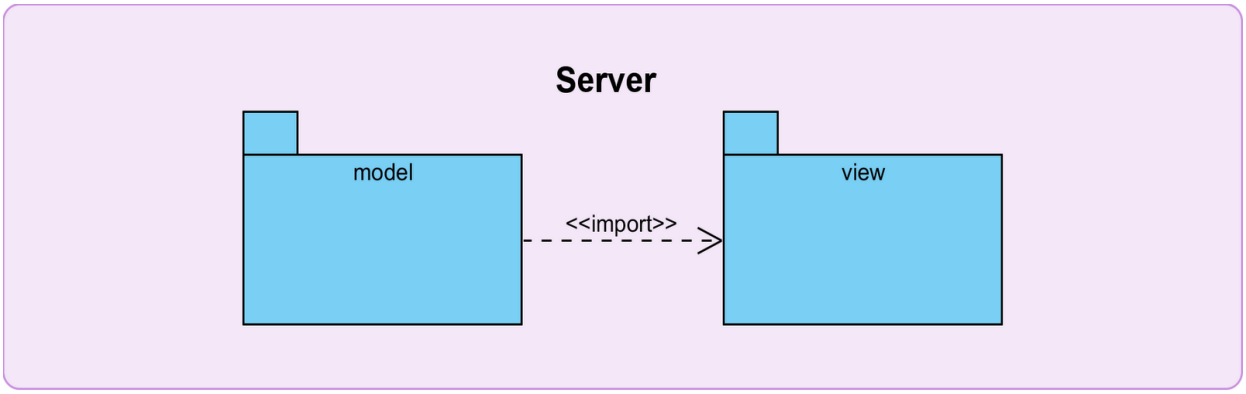
Klientens klassdiagram

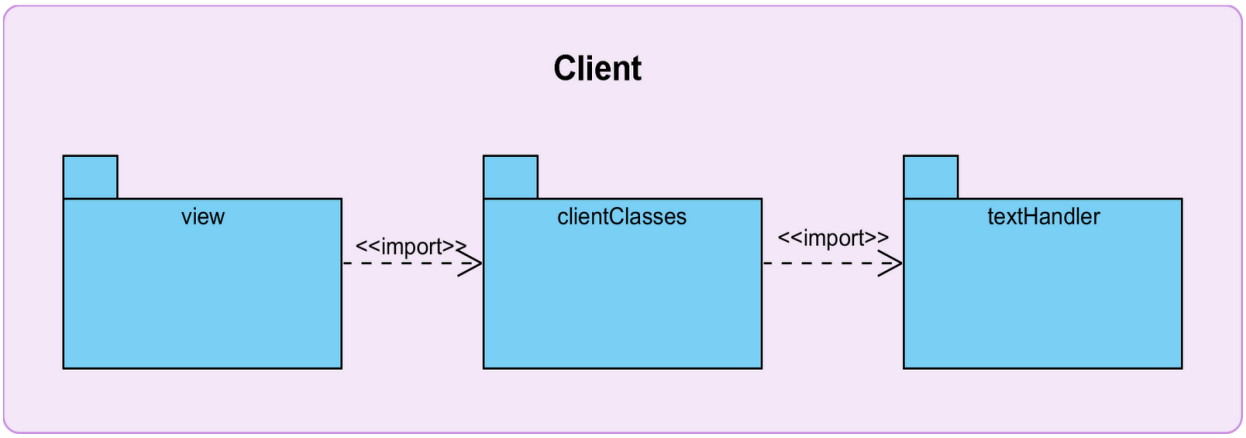


Serverns klassdiagram



Paketstruktur





SheepChat

användarmanual

Innehållsförteckning

Om SheepChat

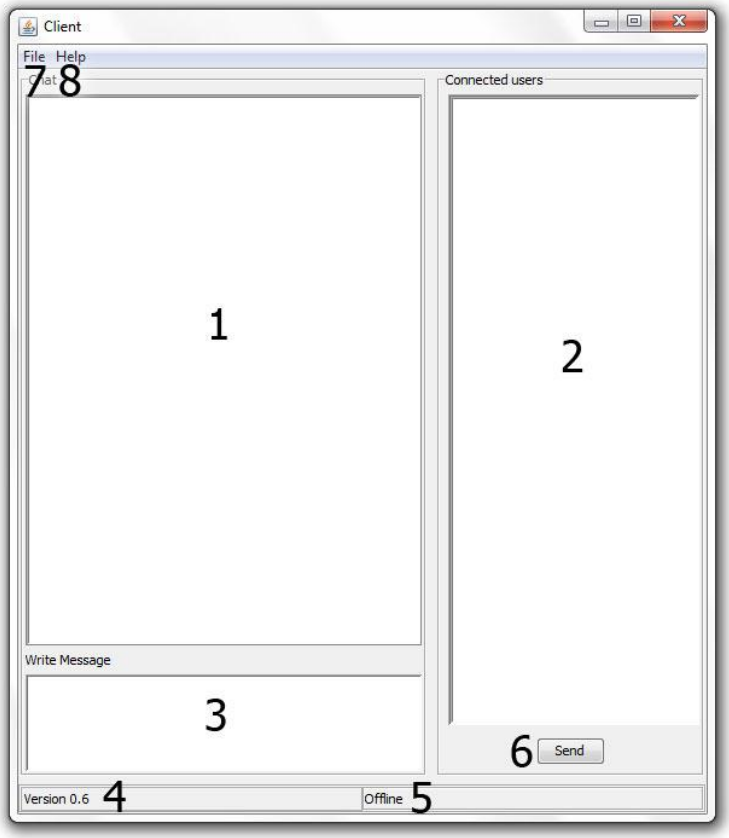
SheepChat är ett simpelt chatprogram (klient/server) som går ut på att man som användare ansluter sig till en server för att sedan kunna kommunicera med alla andra anslutna till samma server. När man väl är ansluten till en server kan man se alla andra användare som är anslutna till servern för att veta vilka som man kommunicerar med genom servern.

Serverprogrammet är lätt att starta för en användare, det enda som behövs är att port 1234 på datorn som kör serverprogrammet är öppen för datatrafik.

SheepChat kan förutom att fungera som ett chatrum även hantera privata meddelanden (PM) mellan två användare. För att använda sig av funktionen skriver användaren: "/<mottagare> <meddelande>" så är det endast mottagaren och avsändaren som kan se meddelandet. Meddelandet kommer att visas i chatfönstret likt ett meddelande ämnat åt alla anslutna användare.

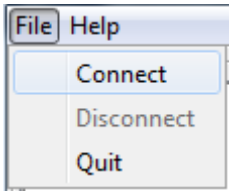
Användargränssnitt

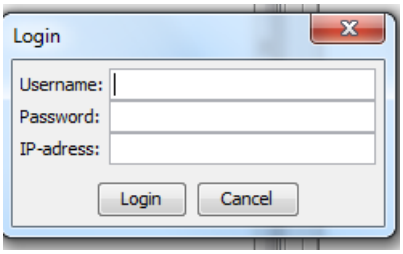
När du som användare har anslutit till en server kommer följande fönster att visas. Det här är huvudfönstret som användare kommer att spendera mest tid framför. I bilden nedan finns siffror som markerar de viktigaste elementen och till siffrorna finns en kort beskrivning av vad de representerar.



1. Chatfönster - Fönstret där alla meddelanden visas.
2. Klientlista - Här visas en lista på alla inloggade klienter.
3. Meddelandefönster - Här skrivs texten som skall skickas.
4. Versionsnummer – Version utav klienten.
5. Uppkopplingsstatus (online/offline) - Visar om klienten är uppkopplad mot servern.
6. Skicka - Skickar texten i meddelandefönstret (3) till önskade användare.
7. Filmeny - innehåller alternativ som används för att ansluta, koppla ifrån eller stänga klientfönstret.
8. Hjälpmeny - Innehåller nyttig information om klienten för oerfarna användare.

Att koppla upp sig mot en server

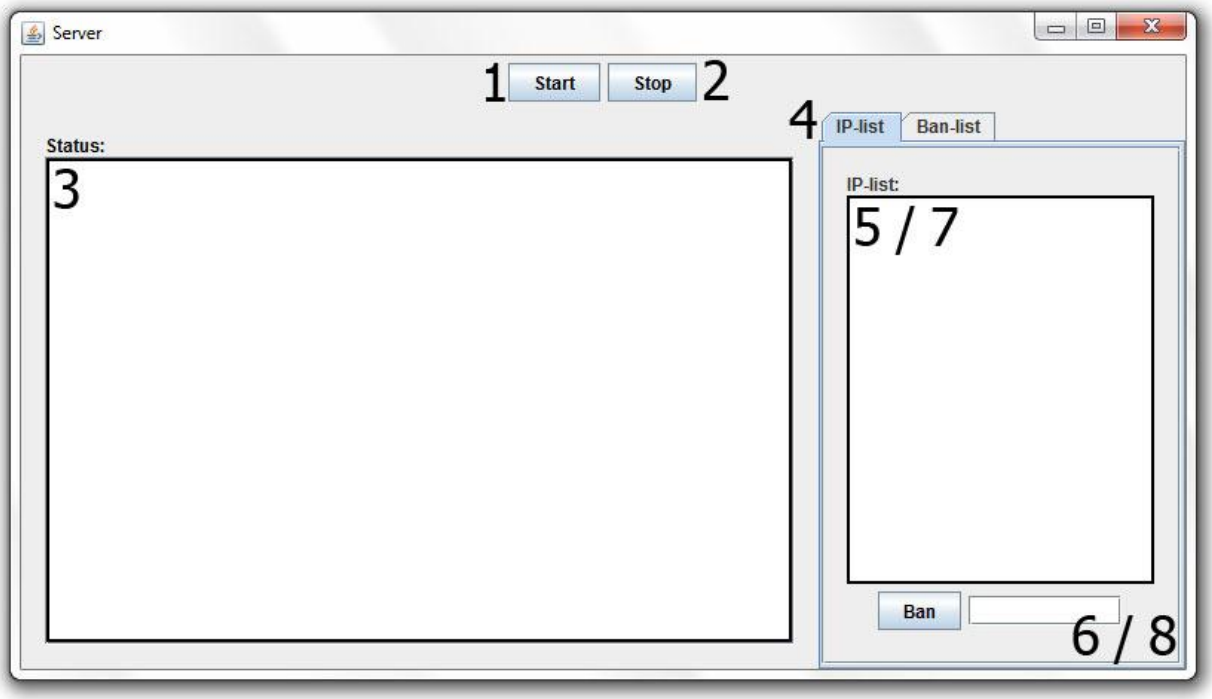




Om en användare önskar koppla upp sig mot en server krävs följande steg:
Se till att port 1234 är öppen för datatrafik.
Välj sedan alternativet “Connect” i File-menyn (se 8 i användargränssnittet) för att sedan i följande fönster välja användarnamn (om användarnamnet inte existerar hos servern lagras det automatiskt tillsammans med tillhörande lösenord), lösenord och IP-adress.

Att starta och hantera en server

För att låta en dator agera chatserver så startar man först programmet Server för att få upp följande fönster:



1. Start - Denna knapp startar servern, den börjar alltså lyssna efter nya klienter. Den skapar även en användarlista om inte en sådan redan finns.
2. Stop - Denna knapp stoppar servern, den kopplar bort alla anslutna klienter och slutar lyssna efter nya.
3. Statusfönster - Här kan man observera olika händelser på servern och vilken tid de skedde. För vidare information, läs sektionen *Statusmeddelanden*.
4. Listflikar - Klicka här för att växla mellan att visa *Klientlistan* eller *Blockeringslistan*.

5. Klientlista (IP-List) - Visar IP-adresser för alla klienter som är anslutna till servern.
6. Blockeringsfält - Genom att skriva in en IP-adress i textfältet och sedan klicka på *Ban* så blockeras alla anslutningar från denna IP-adress. Om det redan finns anslutningar från denna adress så kopplas de ifrån.
7. Blockeringslista (Ban-List) - Visar en lista över alla IP-adresser som är blockerade från att ansluta till servern.
8. Avblockeringsfält - Genom att skriva in en IP-adress här i textfältet och sedan klicka på *Un-ban* så hävs blockeringen och klienter från denna adress har nu möjlighet att ansluta igen.

Statusmeddelanden

Statusmeddelanden listas först med vilket datum och tid händelsen skedde följt av händelseinformationen.

SERVER STARTED - visar att servern startades och vilket IP och portnummer den lyssnar på.
SERVER STOPPED - visar vilken tid servern stoppades.
JOIN - visar att en klient anslutit sig till servern och från vilken IP-adress är ansluten ifrån. Adressen visas även i *IP-listan*.
LEAVE - visar att en klient har lämnat servern.