Användartester – Per

**Serversida**

Jobbigt att starta servern vid sidan av. Vill att servern redan ska vara startad.  
Vet inget om portar – Svårt att starta server.

Infofält –

Stäng av fönster

**Klientsida**

Framgår inte vad det är för IP-adress.

Förvånad över att loginfönstret finns kvar.

Kan inte trycka på enter för att logga in till

Vill att det automatiskt ska gå till ”skrivrutan”

\*chattar lite\*

Ingen aning vad save kan vara bra för.

Namn: text   
Vs  
Namn:  
text

Profilbild???

**Förbättring:**

Lägg till färg på texten, emoticons, skicka filer etc...

Simpel chatklient.

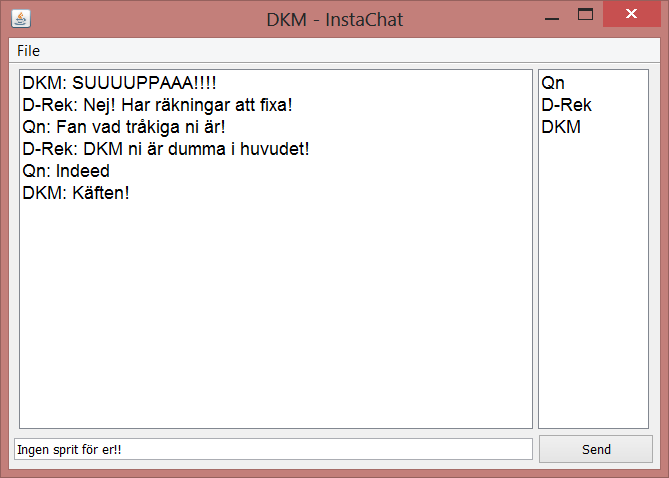
Förväntas registrera ett username snarare än att ha ett tillfälligt namn.

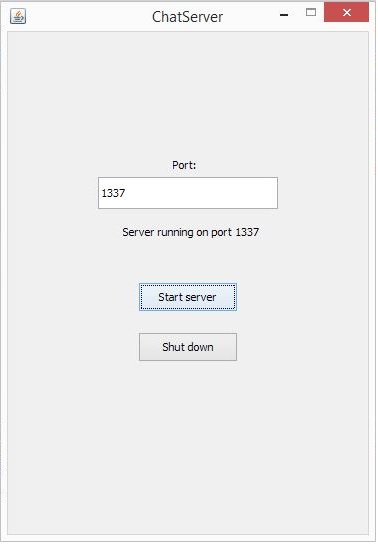
Lättanvänd när man väl kommit igång.

Sammanfattning:  
Komplicerat att starta server. Omständligt. Kräver kunskap om

Slutrapport del B

**Programbeskrivning**  
En enkel chatklient. Den har ett login-fönster där användaren anger sitt namn, samt IP-adress och port till den server hen vill ansluta till. Väl ansluten till en server dyker det upp ett chatfönster som representerar ett chatrum. I chatrummet kan man kommunicera med andra användare genom att skicka meddelanden genom denna server. Dessa meddelanden dyker upp i en chatruta som är synlig för alla som är anslutna till chatrummet. Namnet av alla som är ansluta till servern visas i en lista till höger av fönstret.



Vi har även gjort ett gränssnitt för att sätta upp själva servern över vilken alla användare av chatklienten kommunicerar. Den består till skillnad från chatklienten endast av ett fönster. I fönstret finns ett textfält där användaren anger portnumret av den port som servern ska sättas upp på. Genom att ange en giltig och ledig port, och klicka på ”Start server” så sätter användaren upp en server. För att stänga av servern klickar användaren på ”Shut down”. I mitten av fönstret finns en text som anger huruvida en server är igång.

**Användarbeskrivning**Programmet är användbart för alla människor som vill ha en enkel chatservice för att kommunicera med varandra. Användarna behöver ha vissa grundläggande kunskaper om hur en chatklient fungerar, och de måste kunna få tag på rätt IP-adress och port för att ansluta till rätt servern. Denna information bör tillhandahållas av den person som har satt upp servern. För att sätta upp en server måste denna användare veta vad som är ett giltigt portnummer.

**Användarscenarier**Scenario 1 – Logga in i chatklientenDet första användaren ser när hen öppnar programmet är login-fönstret, som ser ut som i ovanstående bilder. Användaren skriver in sitt önskade namn. För att kunna ansluta till ett chatrum måste användaren ha en IP-adress och port till servern chatrummet körs på. Dessa uppgifter skrivs in i respektive ruta. Därefter trycker användaren på login-knappen.

Scenario 2 – Använd chatklienten (chatta)  
Efter användaren har loggat in dyker nästa fönster upp, och hen får ett meddelande som talar om att hen lyckats ansluta till chatrummet. Användaren kan nu skriva in meddelanden i ett textfält längst ned på fönstret, och sedan trycka på ”send” för att skicka meddelandena till servern. Dessa meddelanden skrivs ut i chatrutan, tillsammans med de andra meddelandena som chatrummet tar emot från andra användare. Denna process sker för alla användare av chatklienten.

**Testplan**För att avgöra hur vårt program mottages och hur vi kan göra det bättre har vi utfört tre användartester. I dessa tester lät vi användarna sätta upp en server genom vårt servergränssnitt. Därefter fick de i uppgift att öppna chatklienten och försöka ansluta till servern de precis satt upp. Väl inne i chatrummet fick de möjlighet att skriva några meddelanden, och helt enkelt klicka sig runt i programmet för att bilda sig en uppfattning av det.

Testerna var så kallade ”think aloud”-tester. Detta innebär att användaren hela tiden talar om vad hen tänker på då hen försöker lösa de givna uppgifterna. Under tiden användarna testade programmet noterade vi alla deras reaktioner och synpunkter. Materialet vi samlade in under dessa tester har sedan kunna användas för att förbättra programmet, men vi kunde inte finna några nya buggar.

Överlag var användarna rätt positiva till vårt program. De hade alla tre erfarenhet av chatklienter sedan tidigare, och två av dem nämnde exempelvis MSN i jämförelse med vår applikation. Efter att de satt upp servern och kommit in i chatrummet tyckte de att användargränssnittet var enkelt och lättförståeligt. En av användarna hade dock några synpunkter på designen, och skulle exempelvis önska att programmet automatiskt fokuserade på textfältet när användaren tryckte ned tangenter.

En sak samtliga användare upplevde var komplicerad var just att de var tvungna att själva sätta upp en server innan de kunde ansluta till ett chatrum. Detta steg är förstås bara någonting som görs en gång för varje chatrum, och bör således inte påverka användandet av själva chatklienten; är ett chatrum redan skapat så behöver användaren endast ansluta till servern som driver chatrummet.

**Programdesign**

Programmet består av två delar som egentligen kan betraktas som två separata program: serversidan och klientsidan. Nedan beskrivs de klasser som utgör dessa två delar, samt några av de viktigaste metoderna i respektive klass.

Serversidan består av två huvudklasser, nämligen Server och ServerWindow. En instans av serverklassen representerar en server, över vilken användare kan kommunicera genom chatklienter. En instans av ServerWindow är användargränssnittet för att sätta upp och stänga av en server. Instanser av Server skapas genom ServerWindow, enligt program-beskrivningen ovan.

Den största metoden i ServerWindow är define, vilken definierar innehållet av serverfönstret; den bestämmer utseendet och funktionen av användargränssnittet. ServerWindow innehåller även en main-metod som körs då man startar applikationen. Denna metod driver hela programmet, genom att kalla på vissa metoder i en viss följd.

Serverklassen består av ett större antal mer eller mindre viktiga metoder. De två av störst intresse är nog acceptRequest och communicate. AcceptRequest lyssnar efter, och accepterar, anslutningsförfrågningar från chatklienter, på en viss port. Den kontrollerar att alla nödvändiga villkor, för att en klient ska kunna ansluta till servern, är uppfyllda, och sätter i så fall upp en anslutning mellan servern och klienten. Communicate sätter upp instanser av den inre klassen ChatService, som sköter kommunikationen mellan alla klienter anslutna till servern.

\*lägg till chat-sidan\*

**Tekniska frågor**Under detta projekt har vi först och främst lärt oss enormt mycket om nätverks-programmering. Vi har utforskat olika möjligheter att sätta upp kommunikation mellan datorer över nätverk, och har således lärt oss att hantera sockets i Java. Mer intressant för kursen, så har vi även lärt oss att tillverka enklare användargränssnitt genom Swing.

Vi har även bekantat oss med Github, vilket har varit till stor hjälp under arbetets gång. Denna erfarenhet kommer troligtvis att komma till nytta även i framtida projekt, då det underlättar samarbete, inom sådana här typer av projektarbeten, i mycket stor utsträckning.

**Arbetsplan**För att komma igång med arbetet började vi med att sätta upp konton på github. Vi laddade ned de applikationer, och skaffade oss den kunskap, som krävs för att kunna sätta upp och använda oss av Git. Därefter satte vi upp en ny git repository, i vilken vår kod skulle lagras och hanteras.

Efter att vi bekantat oss med Git satte vi igång att på egen hand läsa på om nätverks-programmering. Detta var ganska komplicerat, och för att utöka vår förståelse tog vi hjälp av diverse tutorials på youtube. När vi väl fått en överblick av hur ett klient-server chatsystem fungerar, så träffades vi och började skissa på klassdiagram tillsammans. Vi identifierade vilka delar som skulle behövas i vårt system och kunde därefter dela in arbetet i olika ansvarsområden.

Det chatsystem vi hade i åtanke var som gjort för att dela upp mellan två personer, då det bestod av två tämligen fristående komponenter, nämligen en server- och en klientsida. Felix ansvarade över klientsidan, och hade därmed huvudansvaret för klasserna Client, ClientWindow och LoginWindow. Richard ansvarade över serversidan, och hade därmed huvudansvaret för klasserna Server och ServerWindow.

När vi hade delat upp arbetet mellan oss kunde vi börja koda. Under de första två dagarna fokuserade vi främst på våra egna ansvarsområden. Därefter var vi tvungna att arbeta allt mer tillsammans, för att få server- och klientsidan att kommunicera med varandra. Detta var ganska komplicerat och tog rätt lång tid, men vid slutet av första veckan fick vi det att fungera.

Efter att vi lyckats starta en server, och kommunicera över den genom chatklienten, återstod många buggar att fixa, många features att implementera och mycket design att förbättra. Ungefär fem dagar senare hade vi en färdig prototyp som var redo att användartestas. Onsdag den 14 maj utförde vi användartester, som ni kan läsa om på sid x. Baserat på de synpunkter vi fick ifrån användartesterna gjorde vi några slutliga ändringar i programmet. Därefter var vi klara att göra oss redo för den slutliga redovisningen av vårt arbete.

Under programmets utformning har vi försökt hålla principerna bakom objektorienterad programmering i åtanke. Detta innebär att vi försökt designa klasser som representerar objekt med tydliga och enhetliga funktioner. Vi har försökt att tillämpa samma princip på våra metoder, genom att försöka begränsa dem til

Trots denna tydliga arbets-fördelning har vi utvecklat mycket nära varandra och granskat varandras kod. Detta har gjort att vi fått ett helhetsperspektiv av alla delar av programmet