|  |
| --- |
|  |
| InstaChat |
| Ett klient-server-chatsystem |
|  |
| **Richard Sjöberg och Felix De Silva** |
| **15/5-2014** |

|  |
| --- |
|  |

Innehållsförteckning

[Programbeskrivning 1](#_Toc387944108)

[A 1](#_Toc387944109)

[B 1](#_Toc387944110)

[C 2](#_Toc387944111)

[Användarbeskrivning 2](#_Toc387944112)

[A 2](#_Toc387944113)

[B 2](#_Toc387944114)

[Användarscenarier 3](#_Toc387944115)

[A 3](#_Toc387944116)

[B 3](#_Toc387944117)

[Testplan 3](#_Toc387944118)

[A 3](#_Toc387944119)

[B 3](#_Toc387944120)

[C 4](#_Toc387944121)

[Programdesign 4](#_Toc387944122)

[A 4](#_Toc387944123)

[B 4](#_Toc387944124)

[C 5](#_Toc387944125)

[Tekniska frågor 6](#_Toc387944126)

[A 6](#_Toc387944127)

[B 6](#_Toc387944128)

[C 6](#_Toc387944129)

[Arbetsplan 6](#_Toc387944130)

[A 6](#_Toc387944131)

[B 7](#_Toc387944132)

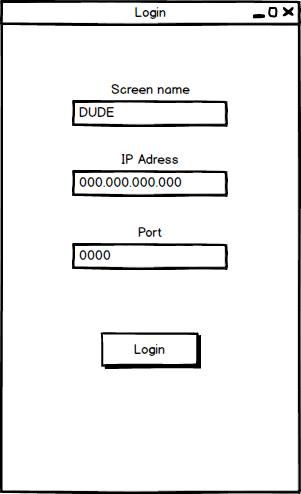
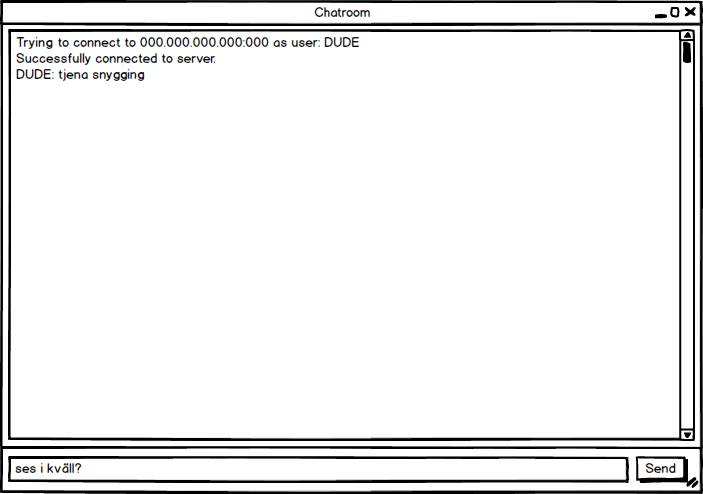
[C 8](#_Toc387944133)

[Sammanfattning 8](#_Toc387944134)

# Programbeskrivning

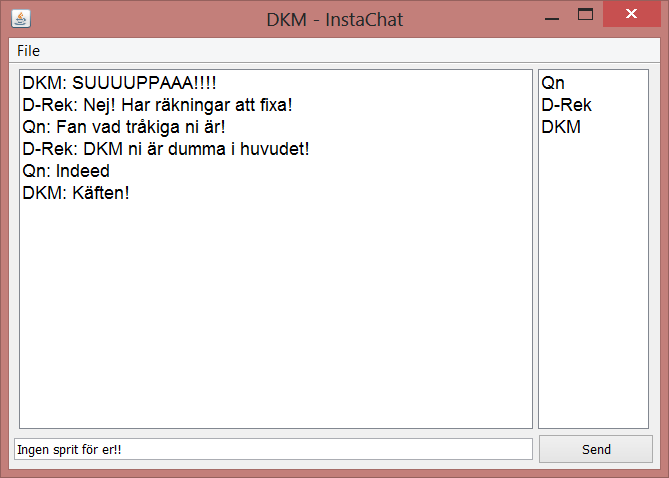
## A

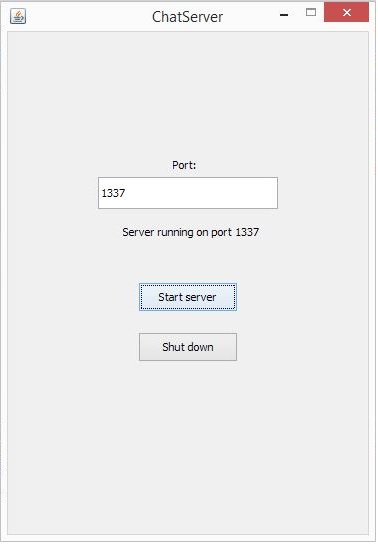
En enkel chatklient. Den ska ha en loginsida där du anger ditt screenname/nick, samt IP-adress och port till den server du vill ansluta till. I chatrummet ska man kunna kommunicera genom att skicka meddelanden genom denna server. Dessa meddelanden ska dyka upp i en chatruta som är synlig för alla som är anslutna till en viss server. Nedanstående konceptskiss illustrerar hur programmet kommer att se ut.



## B

En enkel chatklient. Den har ett login-fönster där användaren anger sitt önskade användar-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. 

## C

Av bilderna ovan framgår det tydligt att vi hållt oss till vår ursprunliga idé. Login-fönstret i vår chat-klient är i stort sett identisk med vår första skiss. Det samma gäller chatfönstret, med undantaget att vi även lagt till en lista över alla anslutna användare. Ursprungligen hade vi inga planer på att implementera ett gränssnitt för server-sidan av vårt program. Detta är någonting vi har lagt till, som ni kan se i del B ovan.

# Användarbeskrivning

## A

Programmet är användbart för alla människor som vill ha en enkel chatservice för att kommunicera med varandra. De behöver ha vissa grundläggande kunskaper om hur en chatklient fungerar, och de måste kunna hitta rätt IP-adress och port för att ansluta till en viss server.

## B

Programmet fungerar som vi tänkt oss från början, så användarbeskrivningen är den samma.

# Användarscenarier

## A

Scenario 1 – Logga inDet första personen ser när hen öppnar programmet är startskärmen som ser ut ungefär som i ovanstående skiss. Personer skriver in sitt önskade screen name. För att kunna använda programmet måste användaren ha en IP-adress och port till den server hen vill ansluta till. Dessa uppgifter skrives in i respektive ruta. Därefter trycker användaren på login-knappen.

Scenaro 2 – Använd chatklienten (chatta)  
Efter användaren har loggat in dyker nästa ruta upp, och hen får ett meddelande som säger att hen lyckats ansluta till servern. Användaren kan nu skriva in meddelanden i en textruta längst ned på fönstret, och trycka på send för att skicka meddelandena till servern. Dessa meddelanden skrives ut i chatrutan, tillsammans med de andra meddelandena som servern tar emot från andra användare.

## B

Programmet fungerar som vi tänkt oss från början, så scenariona i A är aktuella även för den slutliga versionen av vårt program.

# Testplan

## A

För att kontrollera att programmet fungerar som det ska kommer vi utföra enhetstester med JUnit. Därefter kommer vi att testa programmet i praktiken, dels genom att själva använda oss av det, men även genom att låta andra människor testa programmet. Användarnas uppgifter kommer att bestå av att starta programmet, logga in (ansluta till en server) och skicka meddelanden. Deras synpunkter kommer sedan kunna användas för att förbättra programmet och fixa eventuella buggar.

## B

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.

## C

Vi har inte gjort några automatiserade enhetstester i Junit. Istället har vi själva provat programmet på manuellt vis. Vi har tänkt igenom vilka möjliga problem som kan uppstå, och utformat programmet därefter. Precis enligt testplanen har vi dessutom utfört användartester. Dessa har varit till hjälp för att förbättra programmet.

# Programdesign

## A

Det är i dagsläget svårt att avgöra precis vilka klasser vi kommer att använda oss av, då vi inte har så mycket erfarenhet av nätverksprogrammering. Vartefter vi lär oss mer om hur detta fungerar, kommer vi få allt mer klarhet i hur vi ska utforma programmet. Det finns dock några klasser vi tror att vi kommer att använda oss av.

De två nyckelklasser är server och client. Serverklassen ansvarar för kommunikationen mellan klienterna, och behöver således en ”receive”-metod såväl som en ”send”-metod. Client-klassen ansvarar för kommunikationen mellan servern och de enskilda klienterna. Denna klass behöver således också send- och receivemetoder.

Utöver dessa nyckelklasser kommer vi behöva någon klass för själva chatfönstret (ChatWindow) och en klass för loginfönstret (LoginWindow). För att skriva ut meddelanden behöver vi en print-metod i ChatWindow-klassen. För att ansluta till servern behöver vi någon typ av ConnectToServer-metod i LoginWindow-klassen.

## B

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.

Klientsidan består av tre huvudklasser, till denna sida är det Client, ClientWindow och LoginWindow. En instans av LoginWindow representerar login fönstret som är det första du möts av när du startat klientsidan av programmet. En instans av ClientWindow representerar användargränssnittet för användaren, vilket är det som hanterar vad användaren ser och vart användaren skriver in till. En instans av Client representerar det logiska bakom själva fönstret, här är det där man tar emot och skickar alla meddelandena.

Clientklassen som sköter det största delarna för client sidan har två metoder som är viktiga. Send skickar meddelandet som användaren skickar. Meddelandet tas från ClientWindow och skickas genom send metoden till servern där servern hanterar meddelandet sen. Recieve tar emot meddelanden som skickas från servern. Den metoden tar emot meddelandet och sedan skriver ut det i ClientWindow, vilket är konversationsfönstret som användaren ser.

ClientWindow och LoginWindow är två användargränssnitt och det är inte så mycket gemensamt, men det som de har gemensamt är en metod. Define är som beskriven ovan, metoden som definierar själva fönstret och det innehåll.

ClientWindow sköter det användaren ser och bortsätt från metoden som skriver ut till meddelandet till skärmen och den som skickar iväg meddelandet, så finns det bara en metod som är ganska viktigt för att programmet ska fungera som vi vill. UpdateOnlineUserList är metoden som hållet koll vilka det är som är anslutna för användaren. Den tar emot en lista med användare som är anslutna och skriver ut det på skärmen för att visa användaren vilka det är som är anslutna.

LoginWindow sköter inloggningen för användaren och dess ända uppgift är att kolla att uppgifterna användaren skriver in är ogiltiga eller använda. Den metoden som hantera och namnet är redan använda är LoginCheck. LoginCheck skickar namnet som användaren skriver in och skickar det till servern för att verifiera att namnet inte är använt, är det använt så får användaren ett meddelande att namnet är använt och går chansen att skriva in något nytt.

## C

Första gången vi redogjorde för designen av vårt program, visste vi inte riktigt hur programmet skulle fungera eller vad det skulle bestå av. Del A av detta avsnitt präglades av denna anledning av större osäkerhet än del B, som skrevs då vi var färdiga med programmet och visste precis hur det fungerade.

Redan innan vi började med projektet hade vi några grundläggande idéer på hur det skulle utformas. Vi förutsade exempelvis att programmet skulle bestå av en server- och en klientsida, och detta har vi hållit oss till. När det kommer till mer specifikt hur programmet skulle fungera var vi ganska osäkra, men vi lyckades ändå göra några precisa förutsägelser. Precis som vi angav i del A så innehåller vårt program en serverklass såväl som en klientklass, som båda innehåller send- och receivemetoder. Även klasserna ChatWindow och LoginWindow är implementerade i vårt program. En sak vi inte förutsåg var användandet av ett server-gränssnitt, vilket är någonting vi har lagt till senare i utvecklingen av programmet.

# Tekniska frågor

## A

För att kunna bygga vårt system måste vi skaffa information inom följande områden:

* Github.
  + Hur använder man sig av det?
    - Hur sätter man upp ett projekt som kan redigeras över github?
* Nätverksprogrammering
  + Hur ansluter man en klient till en server
    - lokalt?
    - över internet?
* Objektorientering/design.
  + Hur skapar man system med bra objektorientering?
    - Klassschema.
    - Arv.
    - Interface.

## B

Frågorna vi ställde innan vi påbörjade projektet har väsentligen blivit besvarade. Vi tog reda på hur man använde sig av Github, och kunde därefter använda oss av det för att organisera och redigera vår kod. För att få programmet att fungera, läste vi på om nätverksprogrammering, och tog exempelvis reda på hur sockets fungerar i Java. När det kommer till hur man skapar system med bra objektorientering är det svårt att säga att vi funnit något konkret svar. Det handlar mer om att följa konventioner, och designa väldefinierade klasser som representerar objekt på ett tydligt sätt. Dessa principer har vi försökt följa, även om det finns gott om utrymme för förbättring.

## C

Vi har fått svar på alla våra tekniska frågor rörande Github och nätverksprogrammering.

# Arbetsplan

## A

Vi kommer till stor del att arbeta med projektet tillsammans. Detta för att kunna diskutera olika tillvägagånssätt och komma överens om programmets utformning. Projektet kommer att genomföras enligt följande arbetsplan:

1. Lära oss github på egen hand.
2. Läsa på om nätverksprogrammering på egen hand.
3. Strukturera upp programmet med hjälp av t.ex. klassdiagram tillsammans.
4. Dela upp projektet i olika ansvarsområden tillsammans.
5. Koda, tillsammans såväl som på egen hand med våra egna ansvarsområden.
6. Enhetstester – Få programmet att fungera.
7. Användartester.
8. Förbättring.
9. Färdigställ programmet.

Under hela projektets gång kommer vi att dokumentera vårt arbete och sammanfatta detta i en slutrapport.

## B

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 3. Baserat på de synpunkter vi fick ifrån användartesterna gjorde vi några slutliga ändringar i programmet. Därefter var vi klara att göra oss redo för redovisningen av vårt arbete.

## C

Att döma av del B av detta avsnitt har vi följt vår arbetsplan till punkt och pricko. Vi har gått igenom varje steg i arbetsplanen i den ordning de står uppskrivna i del A. Arbetet fungerade mycket bra, och vi lyckades få ihop ett fungerande program i god tid före deadline.

# Sammanfattning

Under detta projekt har vi först och främst lärt oss enormt mycket om nätverksprogrammering. Vi har utforskat olika möjligheter att sätta upp kommunikation mellan datorer över nätverk, och har således lärt oss att hantera sockets i Java. Dessa kunskaper har vi tillämpat för att få vårt program att fungera.

Vi har även bekantat oss med Github, vilket har varit till stor hjälp under arbetets gång. För att organisera, redigera och granska källkoden har vi satt upp en Git repository, i vilken vi kunnat utveckla vårt program. 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.

Mer intressant för kursen, så har vi även lärt oss att tillverka enklare användargränssnitt med hjälp av Java Swing. Våra användargränssnitt är inte särskilt avancerade, men det är inte det som är syftet med dem heller. Vår idé var att skapa en chatklient med ett mycket enkelt användargränssnitt, och detta har vi lyckats bra med. Det finns dock oerhört mycket utvecklingsmöjligheter, både sett till gränssnittet och programmets funktionalitet.

Under arbetets gång har vi fått många idéer på hur vi skulle kunna utveckla programmet. Nedan följer en lista på några av dessa idéer:

* Göra det möjligt att skicka andra typer av meddelanden (inte bara strängar) som t.ex. grafiska emoticons.
* Göra det möjligt för användaren att ändra typsnitt och färg på den text som hen skriver in i chattfönstret.
* Göra en serverlobby som listar alla chatrum (servrar) som går att ansluta till. Ungefär som en lobby för onlinespel, fast med chatrum istället. Istället för att behöva ange IP och port till den server användaren vill ansluta till, behöver hen endast markera en server och klicka på ”anslut”.
* Lösenordsskyddade chatrum.
* Låta användarna registrera användarkonton i stället för att ange tillfälliga användarnamn.
* Göra chatklienten tillgänglig över internet.
* Etc…