Chat rum

Chat

2042-03-22

Projektmedlemmar:
Lukas Hamrin Brohult <Lukha243@student.liu.se>

Handledare:
Handledare <handledare@liu.se>

Innehåll

Projektplan

1. Introduktion till projektet

Internet Relay Chat (IRC) är ett protokoll för chattmeddelanden. Protokollet använder sig av en klient-server-arkitektur. Det vill säga det finns en server som flera klienter kommunicerar med istället för att kommunicera mellan varandra. Protokollet tillåter att man skriver till kanaler, öppna meddelanden för alla på kanalen, skriver direktmeddelanden till andra användare och mycket mer.

2. Ytterligare bakgrundsinformation

Bakgrunds info finns nedan.
Se Wikipedias artikel: https://en.wikipedia.org/wiki/Internet_Relay_Chat.

Samt IRC protokollet för klienter: https://tools.ietf.org/html/rfc2812

3. Milstolpar

#	Beskrivning
1	**Forska i hur man gör en enkel chat rum!** Titta up i hur man gör en enkel chat rum och sedan få en generel överblick på hur jag ska kunna göra framtida projekt
2	**Ansluta och starta en server!** I detta steg så ska jag få upp en server där man kan gå med i och gå ut ur.
3	**Skapa en client socket** Nästa steg kommer vara att skapa en client socket som kommer prata med min server. Tex ge över meddelanden och andra typer av requests
4	**Skapa olika typer av "request and data objeckt"** Här ska jag skapa några enkla data typer som båda server och client ska använda säg av. Detta för att göra det enkelt för att sicka över data
5	**Ta emot enkla request** På serversidan så ska vi kunna ta emot enkla request och messages och kunna sicka vidare det till resten av våra koplade clients
6	**Skapa en enkel GUI** För att göra det enklare att intragera med så kan vi skappa en enkel meny för medelanden
7	**Gör så gamla medelanden sickas** Om en client joinar senare så bör dem senaste 10-20 medelanden sickas till den nya clienten
8	**Gör så man kan ladda även äldre data** Gör så användaren kan ladda in även äldre medelanden om användaren så vill
9	**Gör så användaren kan gå med i andra kanaler och skapa egna** Gör så användaren kan skapa en egen kanal och också låta andra gå med i sin kanal

10	**Gör så det finns lössen ord på kanalen om man vill det** Gör så användaren kan göra ett lössenord på sin egna chat så man måste ha det för att gå med
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
…	

4. Övriga implementationsförberedelser

Planen är att låta användaren gå med i en chat och skriva medelanden.
Jag tror det viktiga är att byga många olika klasser som både servern och clienten kan använda säg av. Jag tror också att bygga projecktet så öppet som går är det viktigaste. Altså att klasserna har funktionerna som går utåt och det är det enda. Med det så kan man enkelt limitera förviring när man bygger andra system som ui och annat

Projektrapport

Även om denna del inte ska lämnas in förrän projektet är klart, är det **viktigt att arbeta med den kontinuerligt** under projektets gång! Speciellt finns det några avsnitt där ni ska beskriva information som ni lätt kan glömma av när veckorna går (vilket flera tidigare studenter också har kommenterat).

Tänk på att ligga på lagom ambitionsnivå! En välskriven implementationsbeskrivning (avsnitt 6) hamnar normalt på **3-6 sidor** i det givna formatet och radavståndet, med ett par mindre UML-

diagram och kanske ett par andra små illustrerande bilder vid behov. Hela projektrapporten (denna sista halva av dokumentet) behöver sällan mer än 10-12 sidor.

6. Implementationsbeskrivning

I det här avsnittet, och dess underavsnitt (6.x), beskriver ni olika aspekter av själva *implementationen*, under förutsättning att läsaren redan förstår vad *syftet* med projektet är (det har ju beskrivits tidigare).

Tänk er att någon ska vidareutveckla projektet, kanske genom att fixa eventuella buggar eller skapa utökningar. Då finns det en hel del som den personen kan behöva förstå så att man vet *var* funktionaliteten finns, *hur* den är uppdelad, och så vidare. Algoritmer och övergripande design passar också in i det här kapitlet.

Bilder, flödesdiagram, osv. är starkt rekommenderat!

Skapa gärna egna delkapitel för enskilda delar, om det underlättar. **Ta inte bort några rubriker!**

Även detta är en del av examinationen som visar att ni förstår vad ni gör!

6.1. Milstolpar

Ange för varje milstolpe om ni har genomfört den helt, delvis eller inte alls.

Detta är till för att labbhandledaren ska veta vilken funktionalitet man kan ”leta efter” i koden. Själva bedömningen beror *inte* på antalet milstolpar i sig, och inte heller på om man ”hann med” milstolparna eller inte!

6.2. Dokumentation för programstruktur, med UML-diagram

Programkod behöver dokumenteras för att man ska förstå hur den fungerar och hur allt hänger ihop. Vissa typer av dokumentation är direkt relaterad till ett enda fält, en enda metod eller en enda klass och placeras då lämpligast vid fältet, metoden eller klassen i en Javadoc-kommentar, *inte här*. Då är det både enklare att hitta dokumentationen och större chans att den faktiskt uppdateras när det sker ändringar. Annan dokumentation är mer övergripande och saknar en naturlig plats i koden. Då kan den placeras här. Det kan gälla till exempel:

- **Övergripande programstruktur**, t.ex. att man har implementerat ett spel som styrs av timer-tick n gånger per sekund där man vid varje sådant tick först tar hand om input och gör eventuella förflyttningar för objekt av typ X, Y och Z, därefter kontrollerar kollisioner vilket sker med hjälp av klass W, och till slut uppdaterar skärmen.

- **Översikter över relaterade klasser** och hur de hänger ihop.
- Här kan det ofta vara bra att använda **UML-diagram** för att illustrera – det finns även i betygskraven. Fundera då först på vilka grupper av klasser det är ni vill beskriva, och skapa sedan ett UML-diagram för varje grupp av klasser.
- Notera att det sällan är särskilt användbart att lägga in hela projektet i ett enda gigantiskt diagram (vad är det då man fokuserar på?). Hitta intressanta delstrukturer och visa dem. Ni behöver normalt inte ha med fält eller metoder i diagrammen.
- **Skriv sedan en textbeskrivning av vad det är ni illustrerar med UML-diagrammet**.

Texten är den huvudsakliga dokumentationen medan UML-diagrammet hjälper läsaren att förstå texten och få en översikt.

- IDEA kan hjälpa till att göra klassdiagram som ni sedan kan klippa och klistra in i dokumentet. Högerklicka i en editor och välj Diagrams / Show Diagram. Ni kan sedan lägga till och ta bort klasser med högerklicksmenyn. Exportera till bildfil med högerklick / Export to File.

I det här avsnittet har ni också en möjlighet att visa upp era kunskapen genom att diskutera koden i objektorienterade termer. Ni kan till exempel diskutera hur ni använder och har nytta av (åtminstone en del av) objekt/klasser, konstruktorer, typhierarkier, interface, ärvning, overriding, abstrakta klasser, subtypspolymorfism, och inkapsling (accessnivåer).

Labbhandledaren och examinatorn kommer bland annat att använda dokumentationen i det här avsnittet för att förstå programmet vid bedömningen. Ni kan också tänka er att ni själva ska vidareutveckla projektet efter att en annan grupp har utvecklat grunden. Vad skulle ni själva vilja veta i det läget?

När ni pratar om klasser och metoder ska deras namn anges tydligt (inte bara ”vår timerklass” eller ”utritningsmetoden”).

Framhäv gärna det ni själva tycker är **bra/intressanta lösningar** eller annat som handledaren borde titta på vid den senare genomgången av programkoden.

Vi räknar med att de flesta projekt behöver runt **3-6 sidor** för det här avsnittet.

7. Användarmanual

När ni har implementerat ett program krävs det också en manual som förklarar hur programmet fungerar. Ni ska beskriva programmet tillräckligt mycket för att en labbhandledare själv ska kunna *starta det, testa det och förstå hur det används*.

Inkludera flera (**minst 3**) **skärmdumpar** som visar hur programmet ser ut! Dessa ska vara ”inline” i detta dokument, inte i separata filer. Sikta på att visa de relevanta delarna av programmet för någon som *inte* startar det själv, utan bara läser manualen!

(Glöm inte att ta bort våra instruktioner, och exportera till PDF-format med korrekt namn enligt websidorna, innan ni skickar in!)