INF1010 - Puslegruppa
-
Kom i gang med PusleChat

Sigmund Hansen – sigmunha@student.uio.no

31. mars 2011

Last ned dokumentasjonen til kildekoden.

1 Klassestruktur

1.1 inf1010.pusle.chat.client.ChatClient

Dette er selve klientprogrammet. Denne klassen lager et vindu og en Socket og kobler til tjeneren [1]. I tillegg kjører den en løkke som tar imot pakker fra tjeneren og sender disse pakkene til ChatListener-objektene som har blitt registrert. Vinduet og klienten er ChatListener-objekter. Hvis en ChatListener sier at den har håndtert pakken, sendes denne ikke videre til andre lyttere i køen.

1.2 inf1010.pusle.chat.client.ChatListener

Dette er et grensesnitt som hjelper til å skille presentasjon og programlogikk. Dette baserer seg på Observer-mønstret. Se http://en.wikipedia.org/wiki/Observer_pattern

1.2.1 public boolean handleMessage(Message m)

Denne funksjonen skal returnere true hvis denne lytteren håndterte pakken.

1.3 inf1010.pusle.chat.client.ChatFrame

Et enkelt brukergrensesnitt med en liste over tilkoblede brukere, et tekstfelt for å skrive beskjeder og en formatert tekstboks for å vise samtalene. ChatFrame er en ChatListener og håndterer alt bortsett fra frakobling og feilmeldinger.

1.4 inf1010.pusle.chat.Message

Dette er en enkel liten klasse som representerer en pakke sendt mellom klient og tjener.

1.5 inf1010.pusle.chat.ChatUtilities

Inneholder konstanter og hjelpefunksjoner for tjeneren og klienten. Metodene send- og recvMessage kan brukes både av tjeneren og klienten. Da slipper vi å skrive duplisert kode i tjeneren og klienten, men vi må sende med lese- eller skrivestrømmen som et argument til metoden.

[1] Tjener er norsk for server.

1.6 inf1010.pusle.chat.server.ChatServer

Selve tjenerprogrammet som sitter og venter på at klienter skal koble seg til.

1.7 inf1010.pusle.chat.server.Client

En klient representert på tjeneren. Denne klassen er Runnable slik at den kan kjøres på en egen tråd. Disse trådene brukes til å ta imot pakker fra klienter hos tjeneren.

2 Hva som mangler i klienten

2.1 I ChatClient

Når man starter klienten, leses opptil tre kommandolinjeargumenter: adressen til tjeneren, porten om den er oppgitt, og brukernavnet til brukeren. Disse puttes i noen lokale variabler som brukes i konstruktøren til ChatClient. I konstruktøren må dere sette variablene sock (klientens Socket), sin (strømmen for å lese data) og sout (strømmen for å sende data). Så må dere til slutt sende en innloggingspakke til tjeneren. Denne kan sendes ved hjelp av metoden sendMessage i ChatUtilities.

handleMessage ChatClient er en ChatListener og har derfor en handleMessage-metode. Det er tre pakketyper som ChatFrame ikke håndterer: frakoblingspakken og feilpakkene.

run I run går en evig løkke. Denne løkka har ansvar for å lese data som er blitt sendt fra tjeneren. Den sender det så videre til lytterobjektene inntil én av dem behandler pakken, eller det ikke er flere lyttere. Dette gir en lagmodell for håndtering av pakker som er veldig modulær. For å lese pakker kan man bruke metoden recvMessage i ChatUtilities.

2.2 I ChatUtilities

Her mangler en del av innholdet i de to metodene.

sendMessage Her skal dere sende data via strømmen som er gitt som argument. Konvertering fra streng til en byte-array er gjort for dere, men siden vi kun bruker én byte til å si hvor mye data som skal sendes, kan vi ikke sende mer data enn 255 byte. Hvis en melding er mer enn 255 byte lang,

skal derfor metoden returnere false. Et alternativ er å dele opp beskjeden i flere mindre pakker, men da bør nok retur-verdien endres til int (antall byte sendt eller antall pakker sendt).

recvMessage Her skal dere motta data fra strømmen som er gitt som argument. Konvertering fra byte-array til streng er gjort for dere i retursetningen, men dere må lese inn typen og lengden fra strømmen og til slutt lese inn data basert på lengden dere har fått undersøkt.

3 Hva som mangler i tjeneren

3.1 I ChatServer

Her mangler det meste av nettverkskoden. Noen små hint til andre deler av koden ligger i «shutdown hooken» i konstruktøren.

processQueue Er ansvarlig for å gå gjennom køen med klienter som har koblet til, men ikke er innlogget ennå. Her må vi passe på at de har sendt en innloggingspakke (navnet er ikke null):

- Feil pakke - Navnet er tomt i dette tilfellet
  - Client rapporterer feilen til klienten
  - Vi må koble fra klienten
- Opptatt brukernavn
  - Si fra til klienten at brukernavnet var opptatt
  - Koble fra klienten
- Innlogging
  - Fjern klienten fra køen
  - Fortell klienten om hvilke andre brukere som finnes
  - Legg til klienten til brukerlista
  - Fortell alle at brukeren har koblet til

Hvis vi kommer til en bruker som ikke har logget inn etter ett minutt, må vi koble fra klienten. Vi trenger ikke å si til brukeren at noe gikk galt.

removeClient Her fjerner vi en klient fra lista med tilkoblede klienter. Hvis brukeren var innlogget, kobler vi fra klienten og forteller de andre klientene om dette.

broadcast Gå gjennom alle innloggede klienter og send dem beskjeden fra sender (husk å sende MESSAGE_FROM-pakker også).

run Her mangler det å motta forbindelser og opprette tråder for dem.

3.2 I Client

Client representerer klienter tilkoblet tjeneren. Denne trenger dere ikke å gjøre noe med, men dere må gjerne fikse på den om dere vil.

4 Protokollen

Alle pakker består av et pakkehode på to byte pluss eventuelle data. Den første byten sier hvilken pakketype det er, og den andre sier hvor mange byte med data det er.

Pakketype	Tjener	Klient
0x01 data	Velkommen ingen	Innlogging brukernavn
0x02 data	Brukerliste brukernavn	ingen ingen
0x03 data	Ny bruker brukernavn	ingen ingen
0x04 data	Bruker dro brukernavn	ingen ingen
0x05 data	Koble fra beskjed	Koble fra ingen
0x10 data	Beskjed fra brukernavn	ingen ingen
0x11 data	Beskjed fra Beskjed	Beskjed Beskjed
Feilpakker:		
0x80 data	Brukernavn tatt ingen	ingen ingen
0x90 data	Invalid packet ingen	Invalid packet ingen