CA-1 Chat-Server/Client + Project Web Server
Dokumentation

Bente Andersen, Steffen Juhl Madsen & Mikkel Vig

- Short description of the chosen design:

Til projektet valgte vi at benytte os af Observer Pattern, da dette er særlig brugbart, når der er et en-til-mange forhold mellem objekter. I programmet extender vores klient Observable, og vores GUI implementerer Observer. Åbner man GUI'en, bliver denne tilføjet som Observer med addObserver metoden, og ved connect, samt når man skriver beskeder, bliver metoderne setChanged() og notifyObservers() kaldt. Ved setChanged indikerer man at objektet er blevet ændret, og andre observere får besked om dette med notifyObservers().
Med implementeringen af Observer-interfacet, følger update-metoden. Denne metode bliver kaldt når observerbare objekter bliver ændret. I GUI'en var det oplagt at bruge den til at opdatere tekstfeltet, som indeholder chathistorik, således:

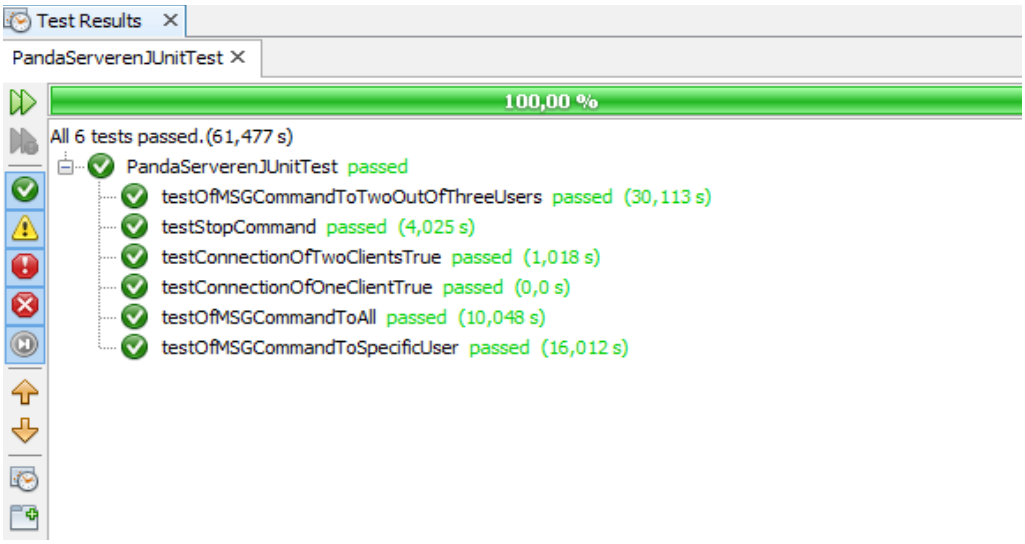
```
@Override
public void update(Observable o, Object o1) {
    outputArea.append(o1.toString()+"\n");
}
```

- Who did what:

Eftersom store dele af opgaven var for svær, til at vi kunne dele os op, besluttede vi os for at lave hele server/klient-delen sammen, hvor vi skiftedes til at skrive. Vores commits på GitHub giver en idé om ca hvem der har skrevet hvad. Det var først da server/klient-delen var færdig, at vi delte os op, således at Steffen sørgede for alt med Azure og deployment, Bente lavede hjemmesiden med security-delen, og Mikkel skrev dokumentation. Dog sad vi, også under denne del, sammen hele tiden, og kunne spørge hinanden til råds.

- Results from JUnit:

Under selve test-fasen havde vi en del problemer. Eksempelvis connecter vi først med én bruger, og derefter med en anden, for til sidst at tjekke om USERLIST bliver opdateret korrekt. Problemet er bare at registreringen af nye brugere, foregår i hver sin tråd. Man kunne derfor ikke være sikker på, hvem der var blevet gemt i listen af brugere først, og vores specifikke asserts fejlede. Det blev endnu værre, da vi havde lavet alle tests færdig, og kørte dem samtidig. Gamle tests der før var gået godt, fejlede nu pludselig igen, og vi fik et indblik i de mange problemer der kan opstå, når man tester noget der involverer tråde. Det lykkedes dog til sidst, ved hjælp af readLine() og Thread.sleep(), at få alle tests til at køre fejlfrit.



- Implementation of state behaviour:

Protokollen har naturligvis haft stor indflydelse på hvordan programmet ser ud. Vi har overholdt den således at det kun er de aftalte kommandoer der accepteres. Det allerførste en ny bruger sender som besked, bliver tjekket af serveren, som sikrer sig at første del består af USER#. Det der følger bliver derefter gemt som den nye brugers navn, og lagt i en concurrentMap. Skriver man noget helt andet, ingenting, eller USER# uden noget efter, modtager man en fejlbesked, og ens socket lukkes.

Efter vellykket login, har brugeren mulighed for at chatte med andre brugere med MSG#. Hvem der skal have beskeden, håndteres af serverens send() metode. Modtagerne kan efterfølgende se beskeden i den aftalte syntaks: MSG#(afsender)#(besked). Er syntaksen ikke overholdt, modtager afsenderen en fejlbesked.
Skrives kommandoen STOP#, bliver afsenderens socket lukket, og metoden removeHandler() bliver kaldt. Her fjernes brugeren fra føromtalte concurrentMap.
Når brugere logger af og på, modtager andre brugere en opdateret liste, af samtlige brugere som er online. Den tråd der sørger for at gemme brugeren i vores concurrentMap, samt removeHandler() metoden bruger begge følgende stykke kode til dette:

```
// denne blok tilføjer users til stringWithUsers og udskriver til alle klienter
String stringWithUsers = PandaProtocol.userlistCommand + PandaProtocol.delimiter;
for (ClientHandler value : clientMap.values())
{
    stringWithUsers += String.valueOf(value.getUsername() + PandaProtocol.userDelimiter)
}
for (ClientHandler value : clientMap.values())
{
    value.send(stringWithUsers);
}
```

Når der eksempelvis står ”PandaProtocol.delimiter”, er det fordi alle tegn der bruges i den definerede protokol, er gemt som statiske variabler i PandaProtokol-klassen. Skal et andet tegn end foreksempel ”#” bruges til at adskille brugere fra meddelelse, er det altså kun i den klasse det skal rettes.