

CA1 Dokumentation

Kort beskrivelse af designvalg(Søren)

Vi har valgt et design hvor at hver del af projektet er delt op i tre pakker. En Chatclient, en Chatserver og en Webserver. Fordelene ved at dele det op, er at hver del fungerer uafhængigt af de andre dele i den forstand at en del kan udskiftes uden at de to andre skal modificeres. Desuden kan projektet skaleres op langt nemmere fordi at webserveren og chatserveren ikke behøves hostes på samme server. Var programmet kodet således at webserveren fx startede chatserveren ville presset på den ene maskine hvor "begge servers" står blive presset mere end hvis de var delt ud på to.

Chatclienten er forsøgt designet efter gang of four's observer pattern. Således at GUI implementere et observerinterface og registrere sig selv som lytter. Og hver gang at Clienten modtager en besked bliver lytteren notificeret og kan håndtere data som de har lyst.

Hvem har lavet hvad(Søren)

Som beskrevet før har vi delt projektet op i tre dele. Og der var også sådan vi startede med at kode. Vi startede op med at kode hver vores del og benyttede hen ad vejen git uden problemer, da vi arbejdede i hver vores pakke. Efter grundarbejdet var kodet og projektet virkede i store træk, begyndte vi at arbejde sammen om de sidste problemer.

Vi har delt projektet således:

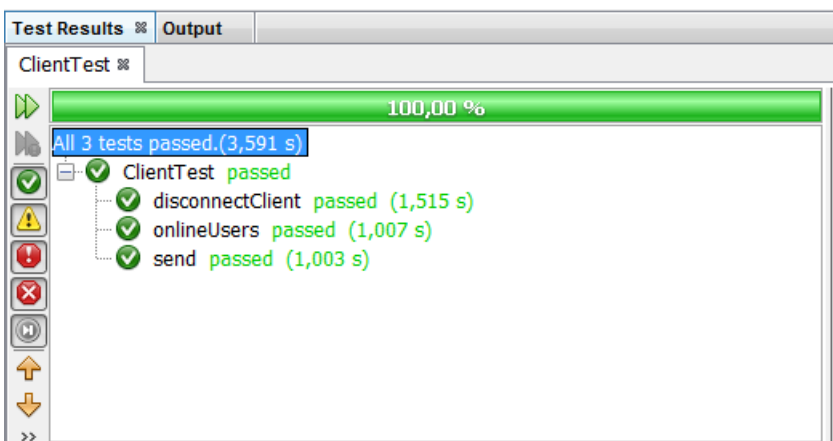
Chatserver er lavet af Morten

Webserver er lavet af Stefan

Chatclient er lavet af Søren

Derefter har vi udarbejdet vores test-class i fællesskab(Pair programming?), og lavet en metode hver.

JUnit test



State behaviour(Stefan)

Vores chatserver implementation benytter sig af 1 tråd som står og lytter på den givede port, når den så modtager en forbindelse på den port. Den laver så en ny handler som tager username og socket. Derefter gemmes denne nye handler i et hashmap også startes den nye handler i en ny tråd. Derefter vil server tråden fortsætte med at lytte efter nye forbindelser.

Wireshark sample(Morten)

Time	10.50.130.161	104.40.194.61	Comment
7.320267000	(38058) → 38058 → 7777 [SYN]	(7777) → 7777 → 38058 [SYN...]	TCP: 38058 → 7777 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=662197532 TSecr=0 WS=128
7.339947000	(38058) → 7777 → 38058 [SYN...]	(7777) → 7777 → 38058 [SYN...]	TCP: 7777 → 38058 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1440 WS=256 SACK_PERM=1 TSval=32952280
7.339991000	(38058) → 7777 [ACK]	(7777) → 7777 [ACK]	TCP: 38058 → 7777 [ACK] Seq=1 Ack=1 Win=14720 Len=0 TSval=662197552 TSecr=32952280
7.340280000	(38058) → 7777 [PSH, ACK]	(7777) → 7777 [PSH, ACK]	TCP: 38058 → 7777 [PSH, ACK] Seq=1 Ack=1 Win=14720 Len=16 TSval=662197552 TSecr=32952280
7.362529000	(38058) → 7777 [PSH, ACK]	(7777) → 7777 [PSH, ACK]	TCP: 7777 → 38058 [PSH, ACK] Seq=1 Ack=17 Win=131584 Len=16 TSval=32952281 TSecr=662197552
7.362591000	(38058) → 7777 [ACK]	(7777) → 7777 [ACK]	TCP: 38058 → 7777 [ACK] Seq=17 Ack=17 Win=14720 Len=0 TSval=662197574 TSecr=32952281
10.114377000	(38058) → 7777 [PSH, ACK]	(7777) → 7777 [PSH, ACK]	TCP: 38058 → 7777 [PSH, ACK] Seq=17 Ack=17 Win=14720 Len=19 TSval=662200326 TSecr=32952281
10.195206000	(38058) → 7777 [PSH, ACK]	(7777) → 7777 [PSH, ACK]	TCP: 7777 → 38058 [PSH, ACK] Seq=17 Ack=36 Win=131584 Len=29 TSval=32952565 TSecr=662200326
10.195237000	(38058) → 7777 [ACK]	(7777) → 7777 [ACK]	TCP: 38058 → 7777 [ACK] Seq=36 Ack=46 Win=14720 Len=0 TSval=662200407 TSecr=32952565
12.349319000	(38058) → 7777 [PSH, ACK]	(7777) → 7777 [PSH, ACK]	TCP: 38058 → 7777 [PSH, ACK] Seq=36 Ack=46 Win=14720 Len=7 TSval=662202561 TSecr=32952565
12.378214000	(38058) → 7777 [PSH, ACK]	(7777) → 7777 [PSH, ACK]	TCP: 7777 → 38058 [PSH, ACK] Seq=46 Ack=43 Win=131584 Len=8 TSval=32952783 TSecr=662202561
12.378272000	(38058) → 7777 [ACK]	(7777) → 7777 [ACK]	TCP: 38058 → 7777 [ACK] Seq=43 Ack=54 Win=14720 Len=0 TSval=662202590 TSecr=32952783
12.378272000	(38058) → 7777 [FIN, ACK]	(7777) → 7777 [FIN, ACK]	TCP: 7777 → 38058 [FIN, ACK] Seq=54 Ack=43 Win=131584 Len=0 TSval=32952783 TSecr=662202561
12.378933000	(38058) → 7777 [FIN, ACK]	(7777) → 7777 [FIN, ACK]	TCP: 38058 → 7777 [FIN, ACK] Seq=43 Ack=55 Win=14720 Len=0 TSval=662202591 TSecr=32952783
12.397441000	(38058) → 7777 [ACK]	(7777) → 7777 [ACK]	TCP: 7777 → 38058 [ACK] Seq=55 Ack=44 Win=131584 Len=0 TSval=32952785 TSecr=662202591

Vi har brugt Wireshark til at sniffe de TCP pakker, det bliver sendt mellem vores Chatserver og vores klient. Det første der sker er, at der bliver lavet et "Three way handshake" som kan ses i det grønne felt. Det består af at klienten sender et [SYN] og forventer at få et svar fra serveren med [SYN, ACK]. Til sidst sender klient et [ACK] hermed at forbindelsen oprettet.

I det sorte felt, sender vi en besked med Connect#Username [PSH, ACK] som server forventer som det første at få. Serveren sender et Online#users[PSH, ACK] tilbage. Til sidst sender klient en [ACK] på at beskenden er modtaget.

Det røde felt:

Der sendes et CLOSE# command, fra klienten til serveren. Herefter sender Serveren et CLOSE# tilbage. Herefter sender [FIN, ACK] fra serveren til klient, hvis klienten er færdig med at sende og klar til at lukke sender den en [FIN, ACK] tilbage. Server sender en [ACK] når den har modtaget [FIN, ACK] fra klienten.