ITE1810 -Arb.Krav del1

**Narvik Høgskole**

ITE1810, Høst

Allan Arnesen, aar029 / 140518

2017

Innholdsfortegnelse

[Selve oppsettet 1](#_Toc495613001)

[RMIServer 1](#_Toc495613002)

[ClientRepository 2](#_Toc495613003)

[Simulering 5](#_Toc495613004)

Figurliste

[Figur 1 - RMIServer tillegg 2](#_Toc495612876)

[Figur 2 - ClientRepository variabler 2](#_Toc495612877)

[Figur 3 - Starter UpdaterThread 3](#_Toc495612878)

[Figur 4 - UpdaterThread run() 3](#_Toc495612879)

[Figur 5 - RemovePlayers() 3](#_Toc495612880)

[Figur 6 - createMap() 4](#_Toc495612881)

[Figur 7 - sendUpdate() 4](#_Toc495612882)

[Figur 8 - addPlayer() 5](#_Toc495612883)

[Figur 9 - simulerer 1000 spillere 6](#_Toc495612884)

[Figur 10 - simulerer 4300 spillere 6](#_Toc495612885)

[Figur 11 - simulerer 8400 spillere 7](#_Toc495612886)

[Figur 12 - simulerer 9999 spillere 7](#_Toc495612887)

Forord

En enkel og kjapp gjennomgang av min kode.

Beklageligvis gikk denne oppgaven meg hus forbi inntil Benedicte i klassen nevnte den mens vi jobbet i lag med en obligatorisk oppgave i ITE-Webapplikasjoner 2.Da var det dessverre bare 5 dager til opprinnelig leveringsfrist. Jeg har derfor hatt litt hastverk med å få på plass en noenlunde representabel oppgave i hvert fall. Men jeg har fokusert på den biten jeg forstod var det viktigste. Nemlig kommunikasjon mellom server og klient.

Sluttresultatet har jeg klart å simulere 9000 spillere med når jeg bare har testet med så enkle spilleregler som mulig. Dog med en del exceptions underveis. På rundt 4000 spillere var det en som posisjonerte seg utenfor labyrinten av en eller annen grunn, men kom seg inn igjen etter hvert (eller bare forsvant). Men mer om det i teksten. Går kun gjennom de viktigste bitene i koden for oppgaven sin del da all kode er relativt godt dokumentert i koden.

# Selve oppsettet

Slik det er satt opp i sluttproduktet benytter serversiden og klienten 2 interface’s for å kommunisere seg imellom. ClientRepositoryInterface og PlayerRepositoryInterface. ClientRepositoryInterface lar nye Players registrere seg hos serveren for å motta oppdateringer om hvor alle andre spillere i labyrinten befinner seg., mens PlayerRepositoryInterface lar serveren registrere oppdaterte «kart» hos hver enkelt spiller igjen.

Enkelt sagt, ClientRepositoryInterface lar klienten kommunisere til ClientRepository hos serveren, mens Player Interface lar serveren kommunisere til Player hos klienten.

Det er i hvert fall ideen og det jeg har prøvd å oppnå.

Kommunikasjonen skal i så måte foregå med at alle spillerne registrere seg hos serveren når de begynner å spille, og sender så inn sine posisjoner fortløpende til serveren. Serveren på sin side sender jevnt og trutt ut oppdateringer til hver enkelt spiller som er tilkoblet. Når så det oppdaterte kartet er mottatt av klienten, oppdateres appleten med de nye posisjonene til spillerne. Om serveren ikke kan koble til klienten når den prøver å pushe ut oppdateringer, i form av en RemoteException, legger vi den aktuelle spilleren til en liste over spillere som skal fjernes. Før neste oppdatering skal ut vasker vi først ut frakoblede spillere.

Rinse and repeat..

Følgende klasser er lagt til av meg.

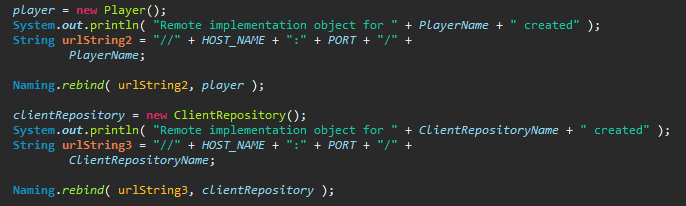
* Player.java
* ClientRepository.java

Følgende interface er lagt til av meg:

* PlayerInterface.java
* ClientRepositoryInterface.java

# RMIServer

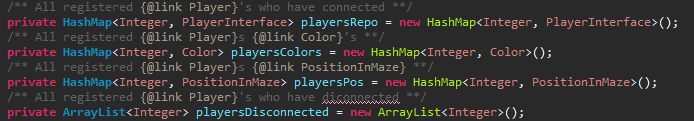
I denne klassen er det bare noen små endringer av meg fra utgangspunktet.



Figur - RMIServer tillegg

Jeg oppretter Player og ClientRepository som Remote Objects.

# ClientRepository

I denne klassen er det en del verdier som vil gå igjen. Dette er:

Figur - ClientRepository variabler

Disse holder på alle data for spillerne som er tilkoblet.

HashMap<Integer, PlayersInterface> playersRepo holder på en Int som Key, som også er spillerne id. Hver Key viser så til sin spiller.

HashMap<Integer, Color> playersColors holder på en Int som Key, som også er spillerens id.

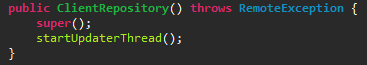
Hver Key viser så til den spillerens farge.

HashMap<Integer, PositionInMaze> playerPos holder på en Int som Key, som også er spillerens id.

Hver Key viser så til den spillerens siste innrapportere posisjon.

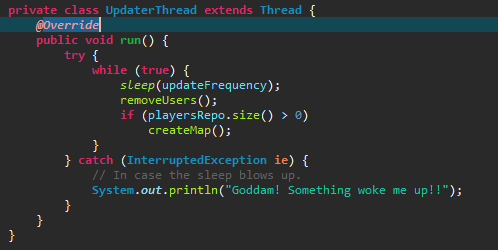
ArrayList<Integrer> playersDisconnected holder en Int som viser til spilleren som er frakoblet sin id.

Først og fremst sparker vi i gang en egen tråd fra ClientRepository som skal ta seg av våre kontinuerlige oppdateringer.



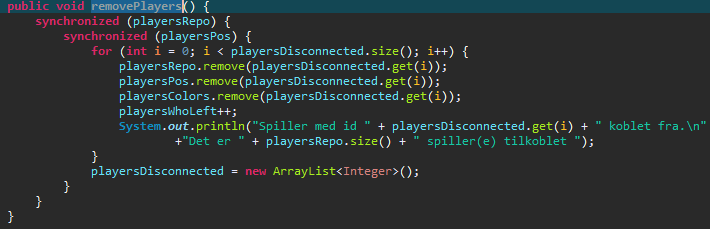
Figur - Starter UpdaterThread

UpdaterThread er selve kjerner i serveren. Den tar seg av oppgavene med å kontinuerlig oppdatere alle spillerne som er tilkoblet med posisjonene til de andre spillerne.



Figur - UpdaterThread run()

Det første vi gjør er å sove litt, bestemt av updateFrequency som er satt i toppen til en fast verdi. Her 75ms. Det første vi gjør etter å ha tatt en kunstpause er å kalle metoden removeUsers().



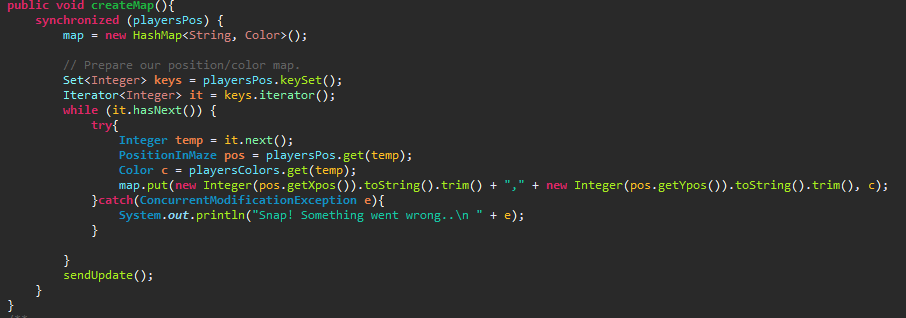
Figur - RemovePlayers()

Denne metoden går gjennom en ArrayList med klient-id til alle som er koblet fra i form av Integer. Vi vasker denne Arraylisten mot de 3 HashMapene nevnt i starten. Når vi er ferdig med å vaske og har blitt kvitt alle spillerne vi ikke lengre kan kontakte, setter vi playersDisconnected til en blank liste igjen.

Som du ser av figur-4 er det neste vi gjør er å produsere et kart. Dette er kartet vi skal sende ut til alle tilkoblete spillere.

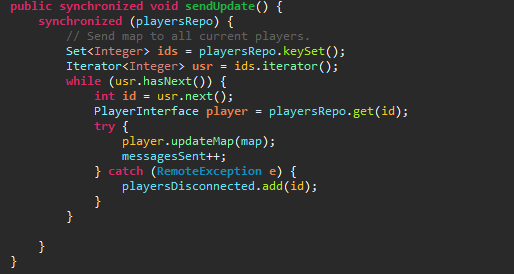
createMap() itererer enkelt og greit over playerPos og playersColors for å sette sammen et nytt HashMap<String, Color> bestående av X og Y koordinatene til hver enkelt spiller, samt fargen som spilleren skal tegnes i.

Når så tegningen er gjort, kaller vi på metoden sendUpdate().



Figur - createMap()

Metoden sendUpdate() itererer gjennom playersRepo, og sender det oppdaterte kartet til alle spillerne våre gjennom PlayersInterface’et.

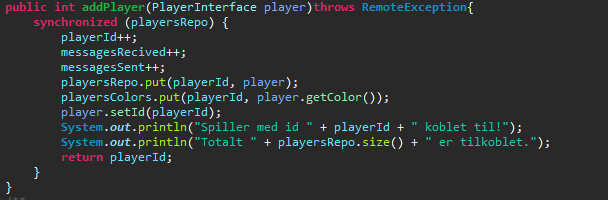


Figur - sendUpdate()

Og om vi ikke kan få kontakt med en spiller, så legges han til playersDisconnected for å bli fjernet til neste runde med oppdateringer.

Og slik fortsetter det.

Gjennom metoden addPlayer() kan nye spillere koble til og delta i spillet. Vi har en teller gående for hver enkelt spiller som legges til, som startet på null når serveren startet. For hver spiller som deltar legger vi til en, så gir vi den tellerverdien vi har som id til neste spiller og legger han til i playersRepo. Så returnerer vi spiller id til spilleren.



Figur - addPlayer()

Og det er vel de viktigste funksjonene som er i koden slik den står nå. Det er selvfølgelig mange flere funksjoner det kunne vært skrevet om, men de er alle sammen så godt dokumentert i selve koden at det føles overflødig å skulle gå gjennom disse på nytt.

# Simulering

I Maze.java er det lagt til 2 felt som er aktuell om du skal ut å teste i.



Boolean ARE\_WE\_SIMULATING indikerer om vi skal eller ikke skal simulere spillet. Det vil uansett bli startet en spiller som representerer deg, pluss det antallet av spillere som du indikerer skal simuleres via playersToSimulate. Men om du har satt ARE\_WE\_SIMULATING til true vil også bevegelsene til din spiller bli simulert. Så ønsker man å simulere spillere og kunne bestemme når din egen spiller skal bevege seg selv, må det kjøres en Maze med ARE\_WE\_SIMULATING til false, og en Maze med ARE\_WE\_SIMULATING til true og det antallet spillere som ønskes simulert. Alle bevegelsene som skal tas er hentet inn fra VirtualUser. Hver Player har sin egen AI tildelt i form av VirtualUser.

Så om du skal styre din egen spiller har du pr dags dato ikke flere muligheter enn å bestemme når neste pre-satte trekk skal foretas.

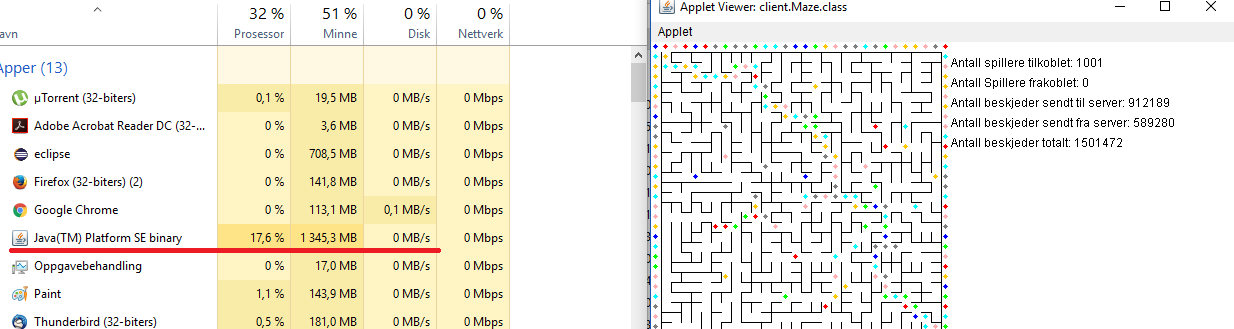
Det er lagt inn sjekk på om feltet du skal flytte til er ledig eller ikke. Er det ikke ledig vil ikke flyttet bli utført.

Det skulle legges inn WASP kontroll for opp, ned, høyre, venstre med som flyttet deg til tilsvarende boks. Så fremst det ikke er vegger på den siden, og at det ikke allerede er noen i den boksen. Dette er det bare ikke tatt tid til å implementere siden jeg er for sent ute, og jeg ikke anså det som en stor del av oppgaven da det ikke innebærer kommunikasjon mellom server eller spiller, men utføres kun på spillersiden. KeyListener er klar, feltet er klart – men ble ikke prioritert.

Men du kan som sagt fyre i gang 2, eller flere, Maze å sitte å se på at du «spiller» mot deg selv ved å bytte på hvilket vindu som tar imot tastetrykk(fokus). KeyListener er deaktivert når simulering kjører.

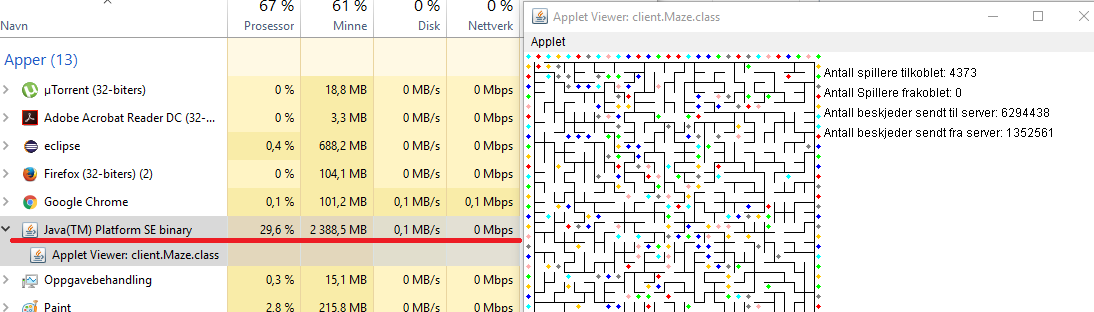
Men her merket jeg også fort en del begrensninger – Spesielt under simulering. Når jeg har aktivert sjekk for om en boks er ledig eller ikke faller antallet spillere som kan simuleres betraktelig. Av den enkelte grunn av at en spiller «krangler» med en annen spiller og de blir stående fast og ingen beveger seg. Og siden de står fast ender det fort opp med en lang kø spillere som ikke kommer seg noen vei. Så med «spotkontroll» om man kan kalle det for det, er det liten vits å simulere med mer så 25-50 spillere.

Men uten noen sjekk på dette, og tillat at spillere stackes på samme området, kunne det kjøres bra mange spillere i simulering. Jeg tok 3 screenshots underveis i siste test jeg kjørte.



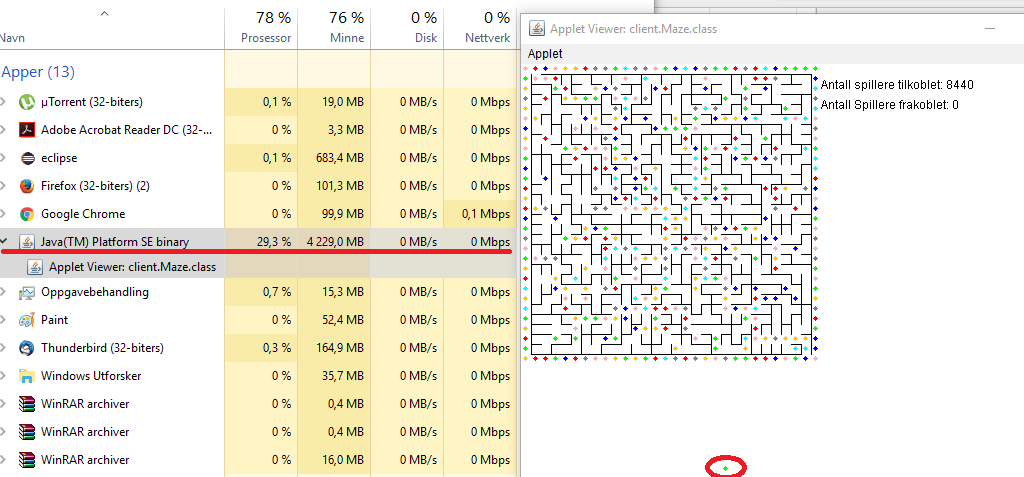
Figur - simulerer 1000 spillere

Ved 1000 spillere trekkes det vel 17% prosessorkraft, gode 1300MB minne. Og det har allerede blitt sendt bortimot 1.5 millioner beskjeder mellom serveren og spillerne.



Figur - simulerer 4300 spillere

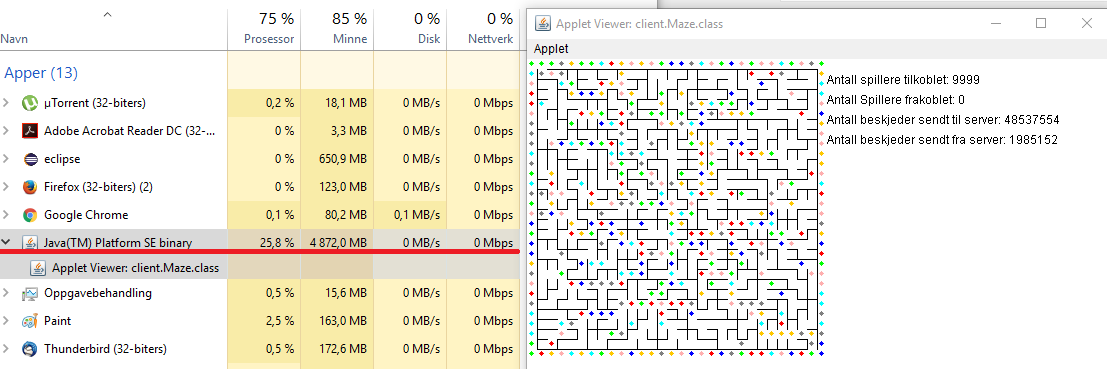
Ved 4300 spillere trekker det 30% av prosessorkraften min og 2400MB minne. Det begynner nå å trekke en del. Med tanke på at jeg kjører dette mot en 4.4GHz I7 prosessor. Og man kan vel strengt tatt si at det lugger generelt en del i kartet.



Figur - simulerer 8400 spillere

Når vi passerer 8000 spillere oppstår det vår første klare feil – En spiller plasserer seg der du ser den røde ringen. Der blir også denne spilleren stående i still en god stund før han a) går tilbake til kartet med at han fikk en ny, og korrekt posisjon, eller b) han bare forsvant.

Fra 4000 til 8000 spillere begynner det å komme hyppigere og hyppigere Exceptions, men det kjører enda og spillere får enda koble til.



Figur - simulerer 9999 spillere

Og slutten er nådd!

Målet i denne simuleringen var 10 000 spillere. Det vil si, som beskrevet over, at vi skulle hatt en total på 10 001 spillere tilkoblet. Men dessverre. Fra 8000 og ut har ting gått drastisk nedover. Minnebruket er nå på nesten 5GB. Men utrolig nok ikke mer prosessorbruk. Jeg hadde regnet med at den skulle øke også, men den har derimot gått litt nedover. Tipper det har en sammenheng med at det begynner å gå rimelig tomt for minne på maskin som kjører dette, og det derfor ikke lar seg gjøre å jobbe fortere. Men 10001 spillere kom jeg aldri til. Jeg kom til 9999, men der var det stopp. Etter 15 minutter stod den enda på 9999 spillere. Antall beskjeder sendt og mottatt av server ble fortsatt oppdatert med spillerposisjonene oppdaterte seg ikke lengre. Det ble også kastet ConnectException.

Nå er jeg ganske imponert over antallet, men samtidig ser jeg at det hadde vært stopp betraktelig tidligere om dette hadde vært noe mer avansert og trengt mer kommunikasjon. For i denne simuleringen kjøres det allerede forhåndsbestemte posisjoner. Som rapporteres inn til server av hver enkelt, som igjen server sender ut til samtlige. Så for et trekk får du n \* antall spillere + n \* antall spillere med beskjeder som sendes.

Nå kan vel det meste i dette spillet ellers sjekkes direkte av spilleren siden han til enhver tid har et oppdatert kart over hvor alle andre er. Og hvor det er og ikke er vegger. Så det skulle vel strengt tatt ikke bli noen særlig mer trafikk om så de siste bevegelsesmetodene hadde blitt implementert. Det som genererer ekstra trafikk er når spillere kobler til, eller når spillere henter oppdaterte tall til statistikk. Men her hentes det i 3 kall. Det kunne lett vært endret til 1 kall. Det vil si at 1/3 av kall for statistikk kunne vært fjernet. Men utenom det ser jeg ikke hvordan det kunne blitt egentlig noen flere kall eller trafikk generelt om bevegelse hadde vært komplett.

# Installasjon

For å kjøre i Eclipse, samme som før.

Men ellers har jeg ikke noe å komme med. Som du sikkert ser i ZIP filen så har jeg forsøkt. Men jeg har ikke fått til enda. Så dessverre har jeg lite å komme med her.

Annet så at klienten må ha tilgang på Player, PlayerInterface, Maze, Box, BoxMaze, BoxMazeInterface, ClientRepositoryInterface.

Serveren må ha tilgang på RMIServer, Player, PlayerInterface, ClientRepositoryInterface, ClientRepository.

Det i en virkelig verden. Slik som det er satt opp pr nå med simulering også må både klient og server også ha VirtualUser og PositionInMaze i tillegg.

Så jeg har nok ikke kommet helt i mål enda med oppgaven, men håper at dette skulle i hvert fall holde til at den er bestått. Så skal jeg ikke bruke mer tid, siden jeg allerede er langt over leveringsfrist og krysser fingrene for at det går litt bedre på neste oppgave.