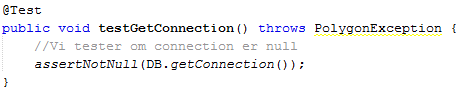
# Junit-tests

For at et program kan virke er det vigtigt at man har et godt test miljø, og at man tester de vigtige metoder, der får ens program til at virke. Dog er det ikke nødvendigt at teste alle metoder i programmet, men de væsentlige metoder der får programmet til at virke.

I vores program har vi valgt at teste vores mappers og database connection, da det er de væsentlige metoder, og uden dem så ville programmet ikke kunne hente information fra databasen.

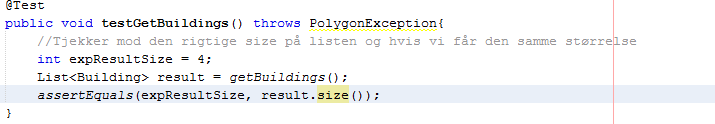
Det betyder at vi lige hurtigt har en test for at tjekke om connection virker:



En simpel metode der bare tjekker for om vi har fået en nullpointer.

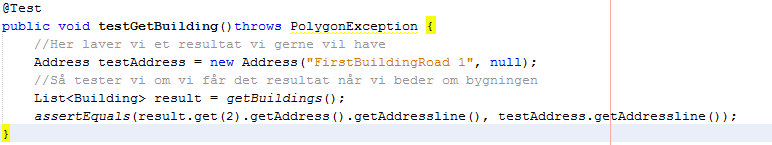
Vi har også valgt at teste vores databasefacade, da det er vigtigt at strukturen i vores program holder.

Så hvis vi går videre til vores databasefacade, Starter vi med at tjekke om metoden GetBuildings virker, det er en væsentlig metode der får listen af bygninger og er rygbenet i vores databasefacade:



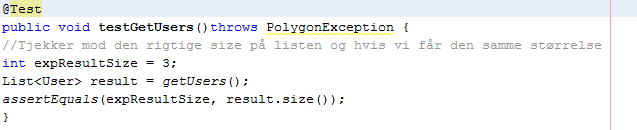
Det er en simpel metode der få listen tilbage og simpelthen tjekker om størrelsen er den samme som det forventede.

Derefter er vi nået til metoden for at modtage information om en specifik bygning, som også er vigtigt da vi har en side, for hver bygning med information, og til det skal vi bruge denne metode ofte:



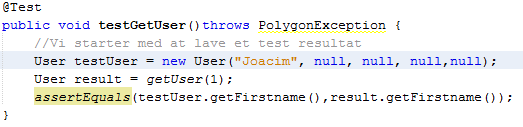
Vi starter med i metoden, at lave et forventet resultat, også henter vi samme information fra en building som vi ved har det resultat, og hvis intet går galt burde det være det samme.

Så har vi en metode der tester for om vi får Users og den er næsten den samme som den der tjekker for buildings listen:



Det er dog vigtigt at users virker, og derfor tester vi det.

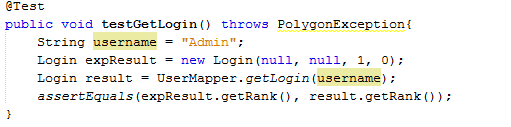
Så har vi også den individuelle user:



Her hiver vi også informationen ud for en individuel user og tester den mod et testresultat.

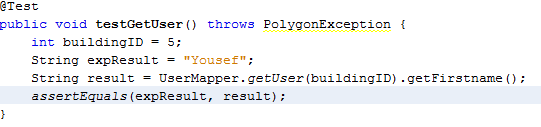
Så går vi videre til vores mappers. For hvis informationen vi får tilbage fra vores mappers ikke virker er vores program i problemer. Så lad os starte med vores usermapper.

Vi kigger først på Login:



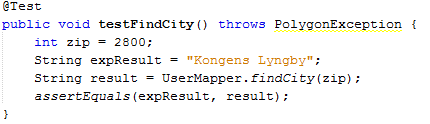
Her har vi en metode som tjekker om GetLogin metoden virker. Det gør vi ved at lave et nyt forventet resultat, også tjekke metoden ved at give den et username, og hvis de to rank attributter passer så virker metoden. Vi tester det her så vi ved at vores login virker, da det er en vigtig metode for at vores hjemmeside kan virke.

Så kommer vi til en metode der henter alt information om en user, og vi tester den her metode så vi ved at metoden virker i alle led af vores struktur:



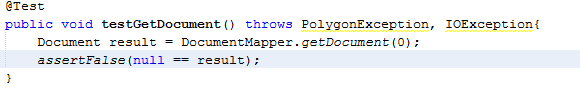
Her tester vi hurtigt om det navn vi før hvis vi indtaster id 5 i metoden er lig det forventer og hvis sandt så virker det.

Så videre til vores FindCity metode, og her tester vi om vores metode der finder bynavn ud fra zip kode er i stand til at fungere, så vi henter de rigtige informationer:



Her laver vi en test zip og så forventer vi en by der passer til den zip kode, og ser så om metoden giver det samme.

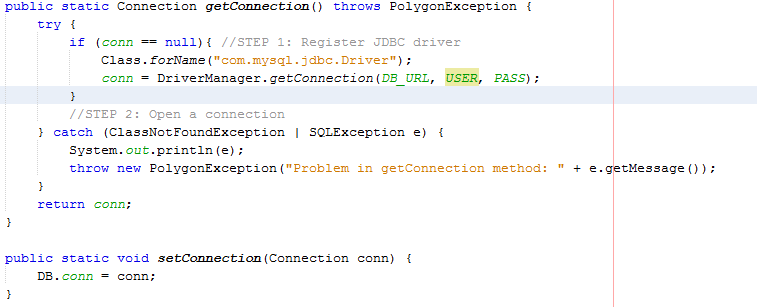
Så tjekker vi lige om vores vigtigste metode i vores documentmapper virker, ved at tjekke om GetDocument metoden kører fint:



En simpel metode der bare hurtigt tjekker om, vi får det som vi forventer hvis vi giver den en falsk building id. Dog kan vi kun lave en meget simpel metode her fordi at hvis vi skulle lave et dokument at teste mod, ville vi oprette et dokument hver gang metoden kørte og det ville ikke være så smart.

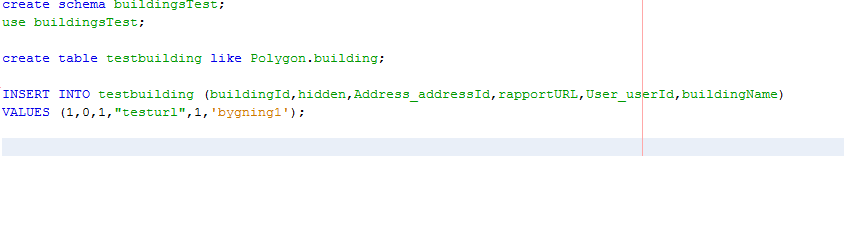
Det har vi kigget på til vores buildingmapper, da det nemlig er den vigtigste mapper for at vores program virker, da det hele er centreret rundt om bygninger. Så her har vi valgt at lave et mock-up altså en falsk database som kan fyldes op og tømmes med testresultater som det passer os.

Men før vi kan kigge på det skal vi kigge på vores getConnection metode, for her havde vi nemlig valgt hvilken database den skulle vælge direkte i getConnection metoden, så hvis vi skulle ændre det til en test database måtte vi ændre lidt i koden.



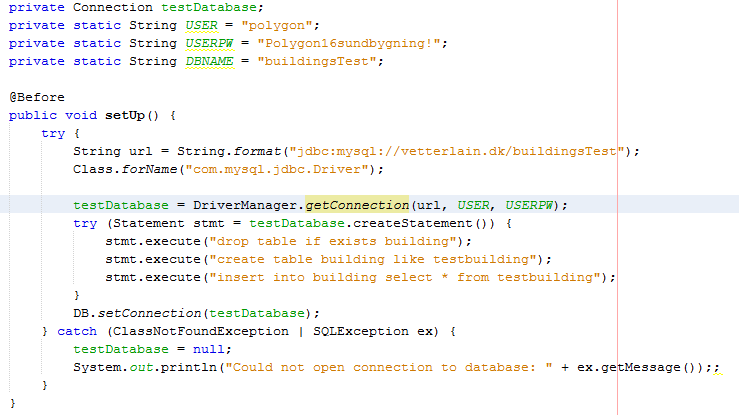
Det vi gjorde var at gøre connection til en singleton ved, at lave en ”If” sætning som kun sætter Connection på ny hvis Connection er null i forvejen, og så lavede vi en setConnection metode som man kan kalde i test klasserne hvor vi kan sette Connection til test databasen. Derved kan vi få test klasserne til at bruge test databasen, men samtidig kan resten af vores klasser stadig fungere.

Vores database SQL script er følgende for at sætte vores testdatabase op:



Det kører vi en gang, og så er det klar til at teste med og burde ikke ændres der fra.

Først sætter vi så vores connection i setUp metoden i vores JUnit class, så den kører for hver metode:



Her sørger vi også for at genoprette vores testbuilding table hver gang, så vi starter med et rent table, hver gang vi tester.

Vi starter med at sørge for at vores GetBuildings metode virker, for at tjekke om vi kan få listen af buildings, da vi vil have den til at virke igennem hele vore struktur:

**BILLEDE HER AF OPDATERET METODE**

Her har vi jo allerede en forventet værdi fra test table, i vores test database så vi tjekker bare hurtigt om størrelsen af listen er den samme.

Vi tester så også getBuilding af samme grund som ovenover:

**BILLEDE HER**

Her tjekker vi så bare om id passer med det resultat vi har fra testdatabasen.

Nu kan vi så teste en metode hvor vi opdaterer en building uden at rode i vores rigtige database og se om det virker. Det er vigtigt at man kan opdatere en building så den får ny information, da det er en User Story i vores program, derfor er det vigtigt det virker:

**………..BILLEDE**

Her tjekker vi så om den opdaterede bygning er lig den forventede værdi vi sætter det til.

## Konklusion?