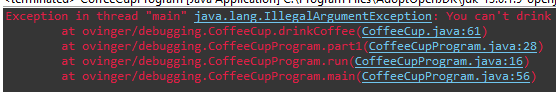
**Del 1: teori**

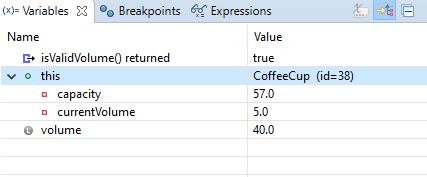
**Teori i coffeoppgaven**

**Oppgave 1**

Første feilmelding jeg fikk når jeg kjørte CoffeeCupProgram var

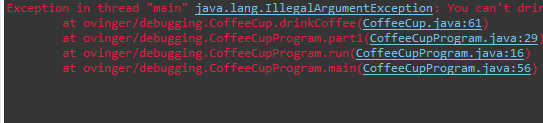


Da er det naturlig å fokusere på part 1() linje 28. Jeg satte en breakpoint der, og undersøkte nærmere. Da fikk jeg meldingen vist under.

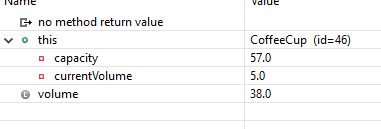


Denne viser at opprinnelig volum er 5 og at kapasiteten er 57. Derfor utløser det et unntak når man prøver å drikke mer enn det som er i koppen.

Etter å ha kommentert ut linje 28, så fikk jeg følgende beskjed.

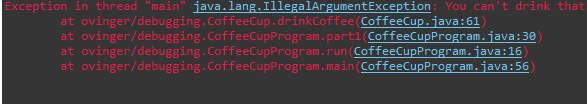


Da må jeg gjør det samme med linje 29. Jeg satte en breakpoint i linje 29, og fikk følgende verdier

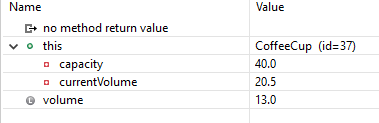


Av samme årsak som i stad vil dette utløse et unntak.

Fikk deretter følgende beskjed.

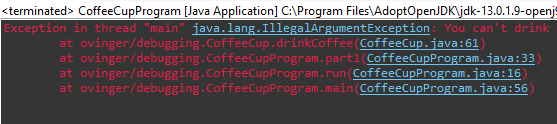


Prøvde en ny strategi nå. Jeg satte breakpoint litt før linje 61 i CoffeCup. Da fikk jeg følgende verdier.



Av samme årsak som før vil dette utløse et unntak.

Fikk deretter følgende beskjed.



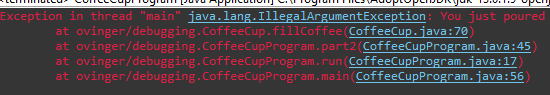
Ved samme strategi som forrige gang fikk jeg følgende verdier.



Igjen hvis volumet er større enn opprinnelig volum, så vil dette utløse et unntak.

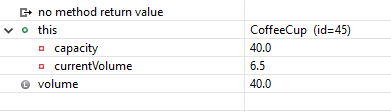
2)

I oppgave en så var det drinkCoffee metoden som utløste et unntak, men i denne oppgaven vises følgende melding.



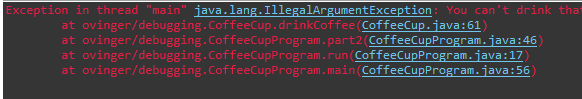
Som sier meg at på linje 70 i CoffeeCup utløser unntaket. Det er fillCoffee metoden.

Setter breakpoint på linje 45 i CoffeeCupProgram, og får følgende verdier

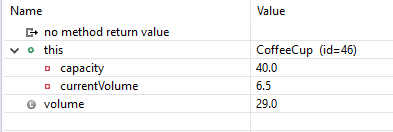


Altså prøver vi å helle oppi for mye kaffe.

Neste melding er



Som da henviser til at drinkCoffeemetoden utløser et unntak.



Her prøver man også å drikke mer enn man har i koppen.

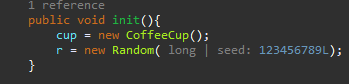
Etter å ha kommentert ut linje 46 i CoffeeCupProgram, så får jeg ingen flere feilmeldinger eller unntak.

**Spørsmål øving 3**

1. ***Hva gjør breakpoints, og hvorfor kan det være praktisk å bruke disse under feilsøking?***

Hvis man mistenker at koden blir feil i et spesielt sted i koden, kan man bruke breakpoints til stoppe debugginsmoduset i det stedet. Da har man mulighet til å ta et og et steg fra der du har laget en breakpoint.

1. ***\*I CoffeeCupProgram-klassen finnes metoden init(). Finn denne metoden.\*Hva skjer når cup = new CoffeeCup(); kjører?***



Når init() blir kjørt så blir det opprettet et objekt med navnet cup. Tilstanden til cup er capacity = 0 og currentVolum = 0.

I tillegg blir det opprettet et tilfeldig tall, men ved hjelp av seed, så vil det tilfeldige tallet være likt hele veien.

1. ***Hva er forskjellen på et objekt og en klasse?***

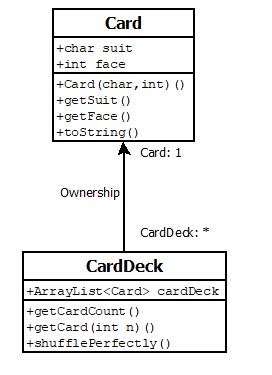
En klasse er en oppskrift/blueprint for hvordan noe skal oppføre seg eller se ut. Et objekt er en ting som følger reglene til den spesifikke klassen.

For å sammenligne så tenker jeg på leiligheter. Klassen er «blueprinten», som forklarer hvordan en spesifikk type leilighet skal se ut, mens objektet er selv leiligheten når den er bygd. Det finnes mange leiligheter som har fulgt samme «blueprint».

1. ***Hva vil skje om man endrer synlighetsmodifikatoren til konstruktøren til CoffeeCup-klassen fra public til private?***

Når jeg endrer konstruktøren til private får jeg beskjed om at konstruktøren ikke er synlig for CoffeCupProgram. Dette fordi at man deklarer noe for private kun hvis man ønsker at det skal være synlig innenfor klassen, og ikke for andre.

**Klassediagram**



**Tanker rundt øving 3.**

Temaet for denne øvingen er i hovedsak debugging og klassediagram. Debugging synes jeg virket som er veldig nyttig verktøy.

I oppgaveteksten anbefalte de å gjennomføre Cardoppgaven. Her lærte jeg at man kunne knytte/assosiere to klasser ved å opprette objekter innad i klassene.

Det som tok mest tid for meg var å forstår det perfectShuffle()-systemet, men ellers gikk det greit.