Objekter

- Objekter er en del av et *kjørende* program som har en
  - *tilstand* – data den husker
  - *oppførsel* – spørsmål du kan stille og tjenester du kan be den utføre

yin yang og OO

- Selv om fokuset i objektorientert programmering er oppførsel, så utgjør tilstand og oppførselen en *dualitet*, hvor den ene sjelden kan eksistere uten den andre.
- Bruken av yin yang-symbolet henspiller på denne dualiteten (i tillegg til OO)

Tilstand og oppførsel-dualiteten

- Tilstanden er det et objekt husker
  - lagres i variabler (kalles også *attributter* og *felt*)
  - tilstanden er enten et poeng i seg selv eller et middel for å realisere en viss oppførsel
- Oppførselen er det du kan be den om å gjøre
  - implementeres i operasjoner/metoder
  - regler for når og hvordan de kan kalles og hva de gjør
  - oppførselen er enten et poeng i seg selv eller et middel for å administrere og begrense tilstanden
- Et objekt med fokus på
  - *tilstand* er *data*-orientert
  - *oppførsel* er *tjeneste*-orientert

Eksempel: en teller

- Tilstandsperspektivet
  - et **Counter**-objekt husker to tall, hvorav det ene er en øvre grense for det andre
  - det andre tallet kan økes inntil det når det første
  - ? *hvilke operasjoner trengs for administrere tilstanden?*

Counter

int end =
int counter = 1

- Oppførselsperspektivet
  - en Counter-objekt har operasjonene
    - **Counter(int)** – angir en øvre grense for telleren
    - **int getCounter()** – returnerer 1 fra starten
    - **boolean count()** – gjør at **getCounter()** neste gang returnerer neste naturlig tall i rekka, med mindre den øvre grensa er nådd. Returnverdien sier om grensa ble nådd.
  - ? *hvilken tilstand trengs for å realisere denne oppførselen?*

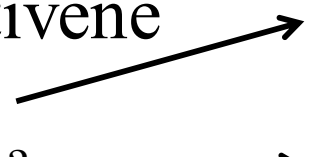
Counter

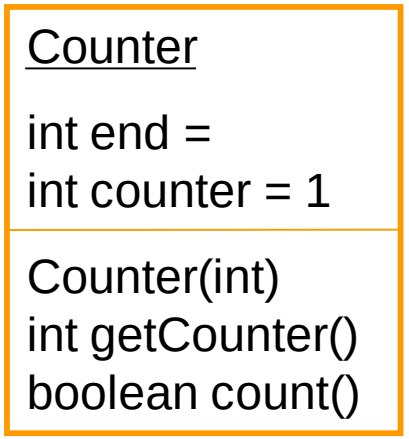
Counter(int)
int getCounter()
boolean count()

Realisering av Counter-klassen

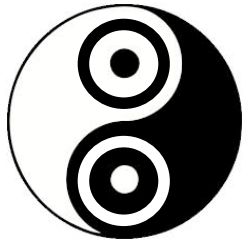
- **Counter**-*klassen* er koden som trengs for at **Counter**-*objektene* skal ha ønsket tilstand/oppførsel

- Når en realiserer klassen, må en kombinere perspektivene
  - hvilke attributter (tilstand) trengs?
  - hvilke operasjoner (oppførsel) trengs?

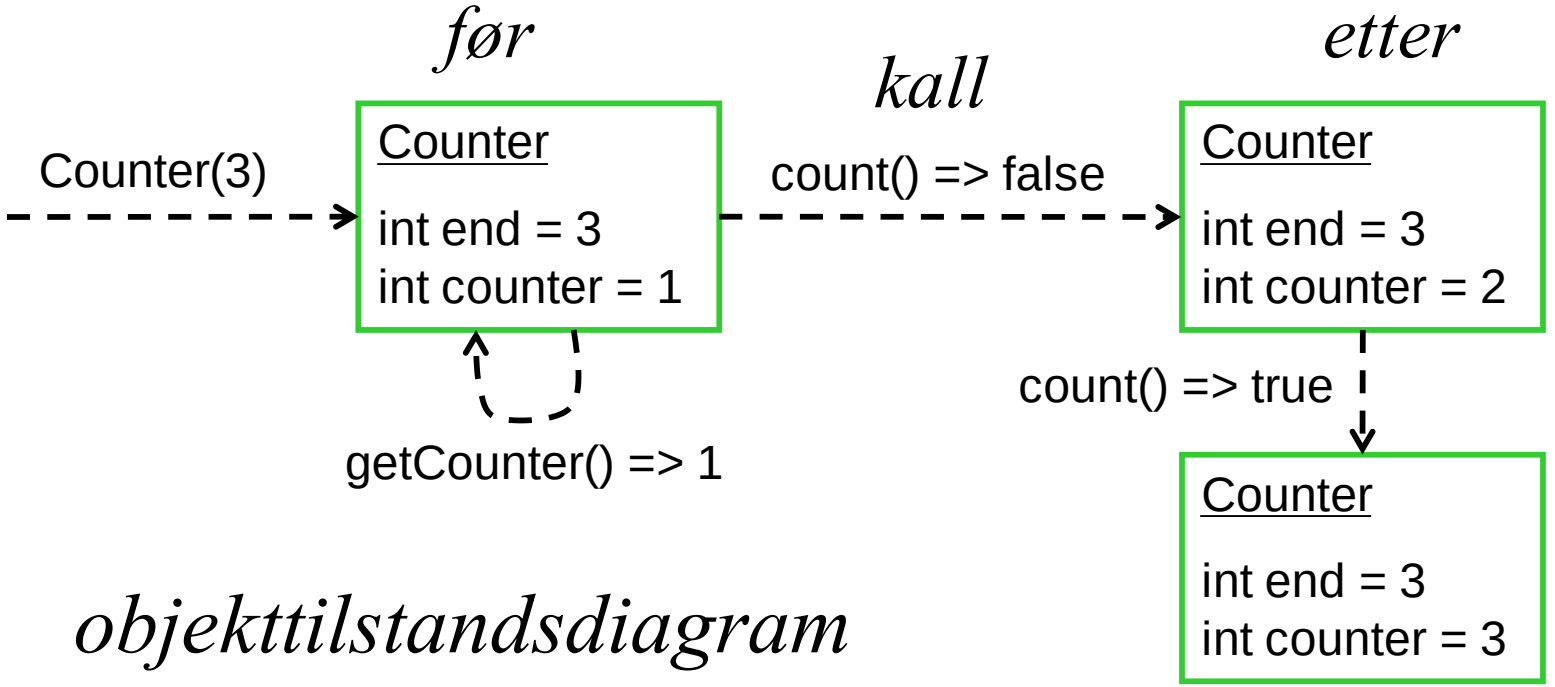




Tilstand og oppførsel



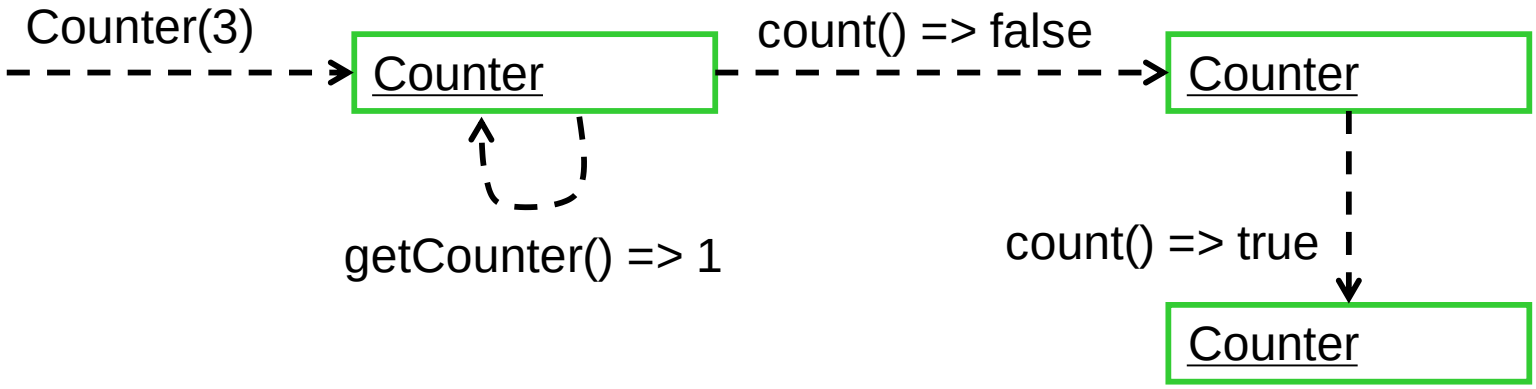
- Oppførsel kan defineres som hvordan de ulike operasjonene endrer tilstanden
- Visualiseres av et *objekttilstandsdiagram*:



objekttilstandsdiagram

Tilstand og oppførsel

- Et rent oppførselsesorientert diagram angir ikke variabler, kun sammenhengen mellom sekvenser av metodekall



- Et slikt diagram er en ren *funksjonell* beskrivelse, uten tanke på realisering

Realisering av Counter-klassen...

Testing av realisering

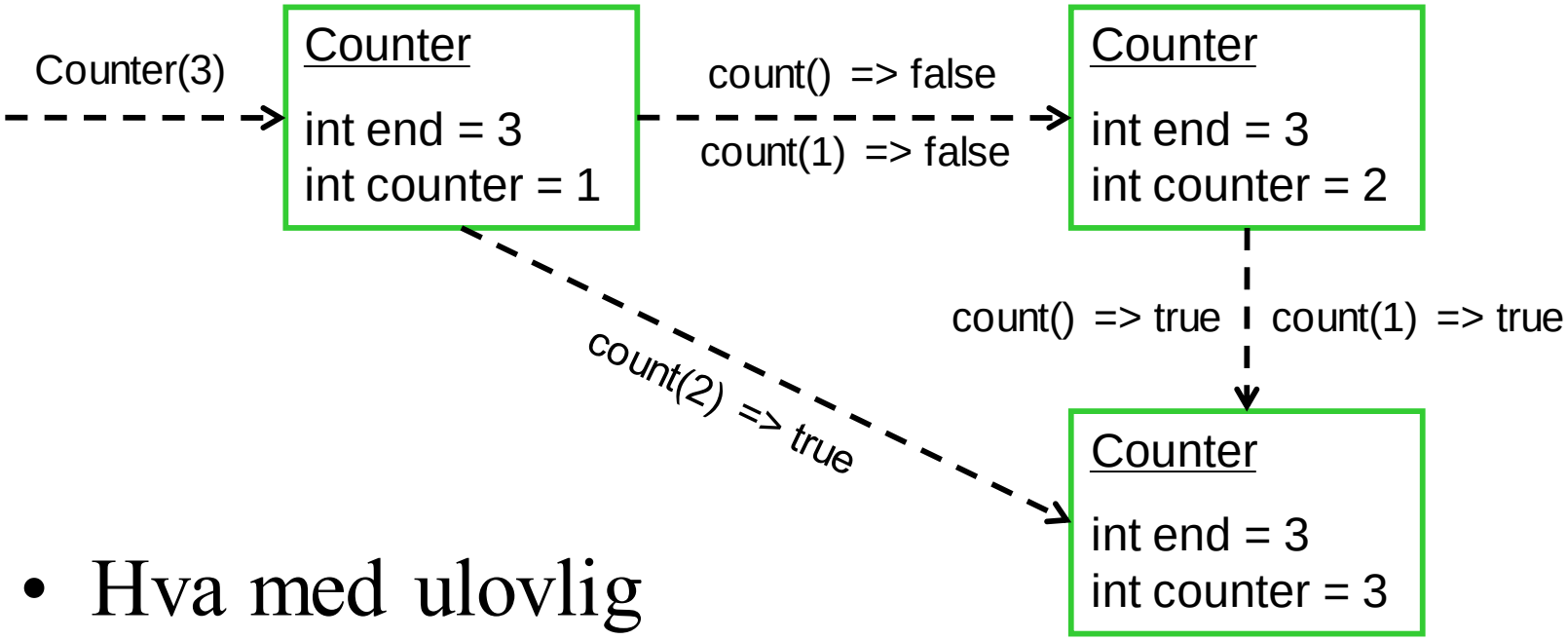
- Objekttilstandsdiagrammet spesifiserer oppførselen gjennom eksempler
- En kan teste om den realiserte oppførselen stemmer med den spesifiserte ved å lage et testprogram som
  - utfører kallene i diagrammet
  - sjekker om den *faktiske* tilstanden stemmer med den *forventede*

- Testkoden som ligger ved øvingene gjør dette vha. JUnit-rammeverket (mer senere)

Realisering av
main-metode i
Counter-klassen...

Variant 2 av Counter

- Oppførselen utvides med count(int)
  Hvordan blir objekttilstandsdiagrammet?



- Hva med ulovlig tilfeller?