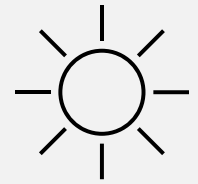
GRUPPE 11 IN1010 – UKE 12

Sivert Fjeldstad Madsen

I DAG

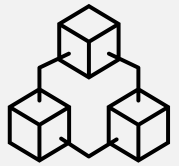
- Model-View-Controller
- Layout
- Fonter og farger
- Oppgaver



MODEL-VIEW-CONTROLLER

- Når vi skriver større programmer er det lurt å strukturere dem på en fornuftig måte
- Vi trenger ikke finne opp hjulet på nytt hver gang
- Derfor benytter vi oss gjerne av et allerede eksisterende **programmeringsmønster**
- I IN1010 lærer vi om **MVC**
  - I IN2000 lærer man **MVVM:** Model-View-ViewModel
  - Det finnes også andre

MVC - FORTSETTELSE

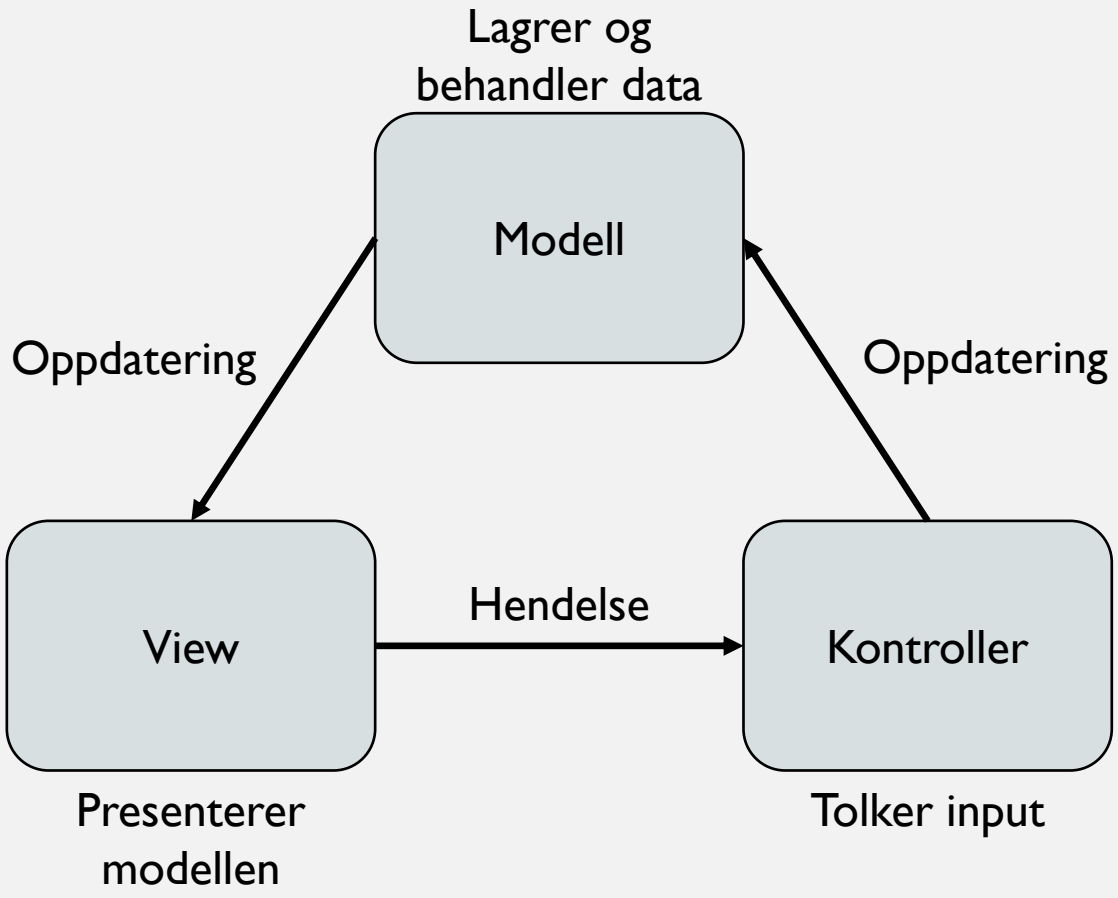


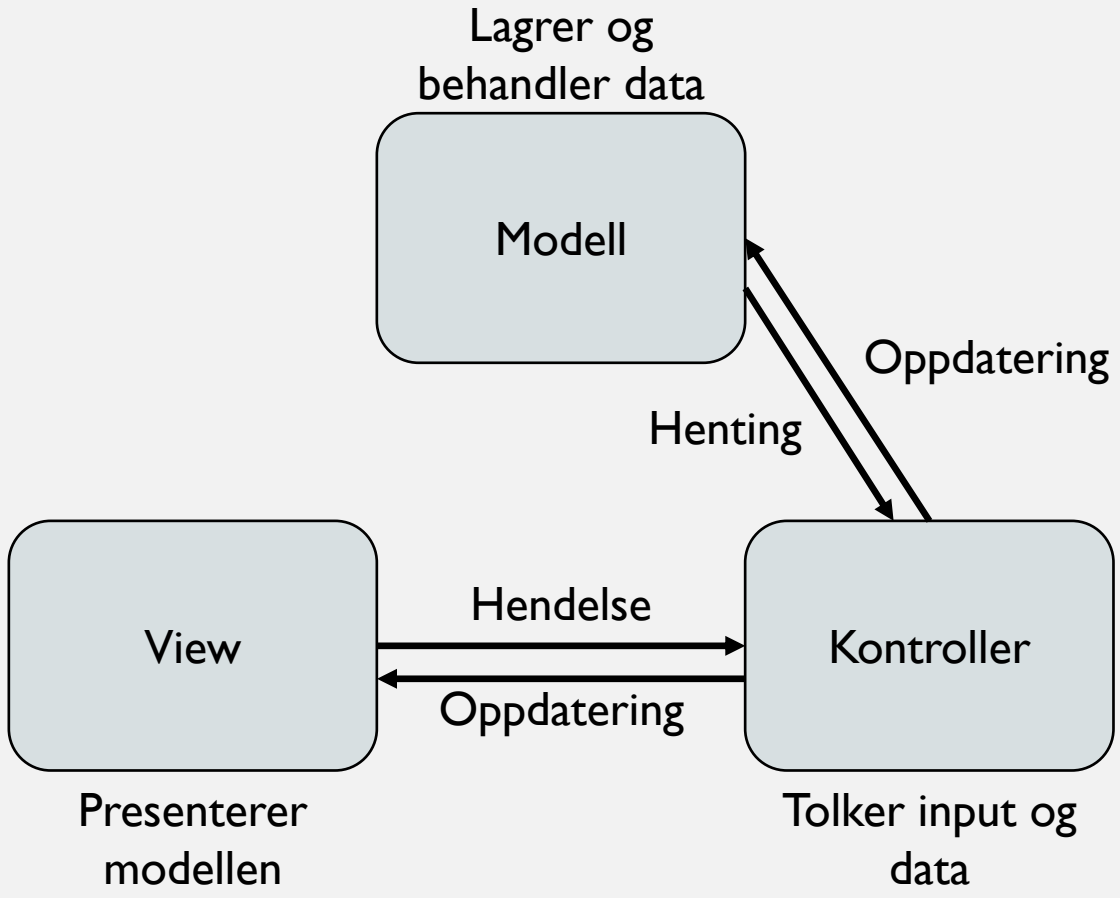
- Tanken bak MVC er å dele opp programmet vårt i tre ulike deler som hver har ansvaret for forskjellige ting
- **Modellen** er der all dataen og mesteparten av logikken bak programmet ligger
  - Dette kan f.eks. være en eller flere lister med data
- **Viewet** er det som presenterer dataen fra modellen til brukeren
  - Typisk **GUI**en til programmet
- **Kontrolleren** er det som gir beskjed om at modellen skal oppdateres
  - Modellen oppdaterer deretter viewet med den nye informasjonen

FINNES DET EN PRESIS DEFINISJON?

- Det kan virke som om det er en del uenighet knyttet til *nøyaktig* hvordan interaksjon mellom de ulike delene skal skje
- Årets forelesning presenterer det som at kontrolleren har ansvaret for å oppdatere modellen *og* oppdatere viewet
- I fjor ble det forelest at kontrolleren oppdaterer modellen, og *modellen* oppdaterer viewet direkte
- På internett finner man tilhengere av begge varianter
- Det viktigste er å ha tredelingen på plass

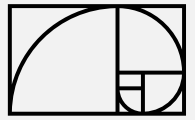
TO FORSKJELLIGE PROGRAMFLYTER





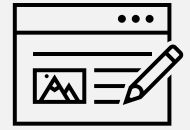
INTERFACE

- Det kan være en god idé å lage et **interface** for view-klassen
  - La alle views tilknyttet programmet implementere dette
- På denne måten blir det enklere å erstatte viewet med et annet dersom man skulle ønske det
- I kontrolleren (og eventuelt modellen) kan man da alltid vite at metodene man bruker finnes uansett hvilket view man bruker
- Dette gjør det enklere for andre å bygge videre på programmet ditt
- Det vil også kunne gjøre testing enklere



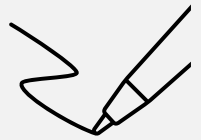
LAYOUT

- Når vi lager GUI vil vi ofte presentere de ulike elementene på bestemte måter
- Vi kan da bruke ulike former for **layout**
  - Vi kan bestemme dette for alle tegneflater (JPanel)
  - Merk også at vi kan ha tegneflater inne i tegneflater
- **BorderLayout** lar oss velge noen forhåndsdefinerte posisjoner
- **GridLayout** lar oss lage et rutenett
- Slide **39** og **40** fra forelesningen gir gode eksempler på dette



FONTER OG FARGER

- Man kan endre fonten og stilen på teksten i alle elementer som inneholder tekst
  - Dette gjøres med element.setFont()
- I tillegg kan man endre størrelse, farge, rammer,og bakgrunner på de fleste elementer
- Igjen gir forelesnings-pdfen en god gjennomgang av dette, på slide **41-44**



OPPSUMMERING

- MVC er et nyttig programmeringsmønster for programmer med brukergrensesnitt
  - Men det er ikke noen konsensus om *akkurat* hvordan det fungerer
- Swing og AWT lar oss endre på utseendet til de ulike elementene
  - Som gjør at vi kan få ting til å se ut (nesten) akkurat som vi vil

- Ukas tips:
  - Husk at du kan skrive nye subklasser av veldig mange av Javas innebygde klasser!
  - Du kan f.eks. definere en variant av JLabel som alltid bruker en viss font