UKE 11
OO & UML modellering II

IN1030 - Gruppe 8

Hva skal vi i dag?

- Repetisjon
  - Sekvensdiagram
  - Use case diagram
- Klassediagram
- Aktivitetsdiagram
- Ukesoppgaver

Repetisjon

UML - hvorfor modellerer vi?

- I systemutvikling er modellering prosessen hvor man utvikler abstrakte modeller av et system
- Det er viktig å forstå at én modell av et system ikke er en komplett representasjon av systemet, men at den kun viser ett perspektiv
- ...derfor trenger man flere typer modeller som synliggjør ulike aspekter ved systemet; ulike modeller representerer ulike måter å se systemet på

Hva er forskjellen på aktører og interessenter?

- Aktører:
  - de som bruker/brukes av systemet
    - individer
    - systemer
- Interessenter:
  - de som påvirker/påvirkes av systemet
    - individer
    - grupper
    - organisasjoner
    - institusjoner

Brukerhistorier → Use-case diagram

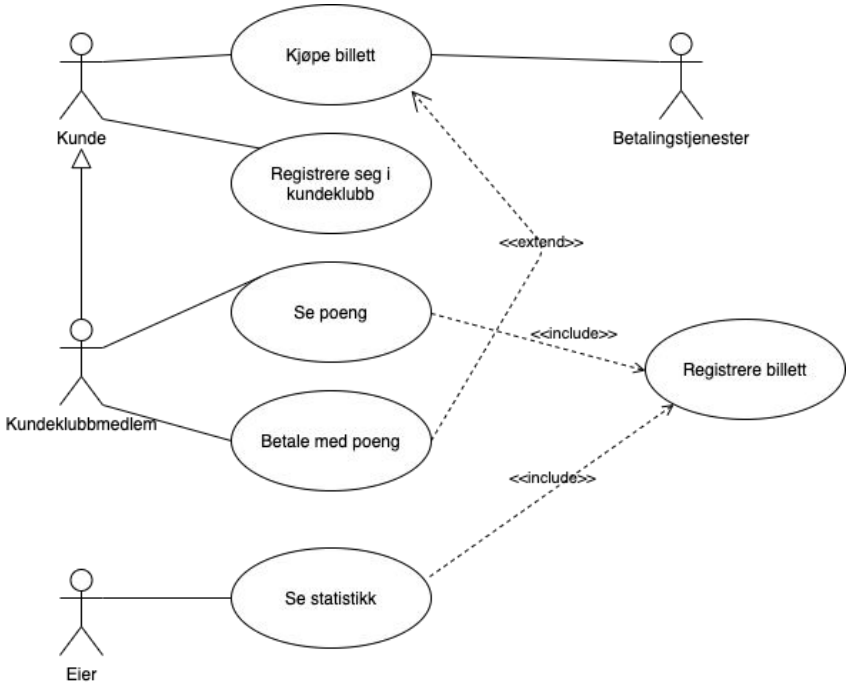
Som kunde ønsker jeg å kjøpe billett slik at jeg kan gå på kino

Som eier ønsker jeg å kunne se statistikk over hvor mange billetter som er solgt slik at jeg kan planlegge visninger fremover

Som kundeklubbmedlem ønsker jeg en oversikt over mine poeng for at jeg kan se om jeg har nok til å kjøpe en kinobillett

Som kundeklubbmedlem ønsker jeg å kunne tjene poeng, slik at jeg kan bruke disse til å betale for billetter

Som kunde ønsker jeg å registrere bruker for å bli medlem av kundeklubben, slik at jeg kan tjene poeng



Tekstlig beskrivelse

Navn: Kjøpe billett
Aktører: Kunde, betalingstjeneste
Prebetingelse: Ingen
Postbetingelse: Billett er kjøpt og PDF genereres

Hovedflyt:
1. Kunde velger en forestilling
2. Systemet viser ledige seter
3. Kunde velger sete blant de ledige
4. Setet holdes av til kunde i 10 minutter
5. Kunde sendes videre til betalingsløsning
6. Kunde velger kort
7. Betaling gjennomføres
8. Bekreftelse på betaling sendes til kunde og registreres i systemet.
9. Billett genereres (PDF)

Alternativ flyt:
4.1 Kunde bruker mer enn 10 minutter på å fullføre kjøp
4.2 Kjøpe avsluttes
4.3 Returnerer til steg 2

6.1 Kunde velger poeng som betalingsløsning
6.2 Kunde logger inn
6.3 Returnerer til steg 7

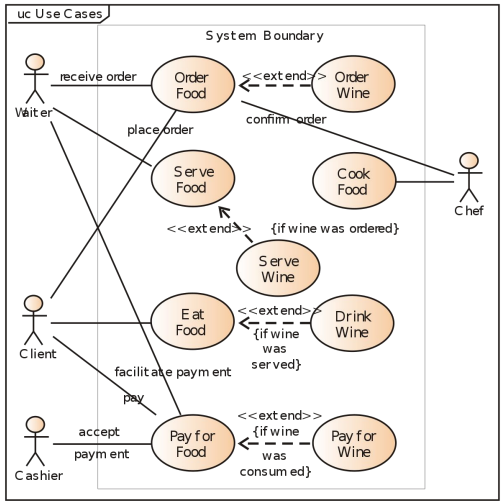
7.1 Betaling kan ikke gjennomføres
7.2 Returnerer til steg 2

Hvilke 4 diagrammer skal dere kunne?

Diagrammer for dette emnet

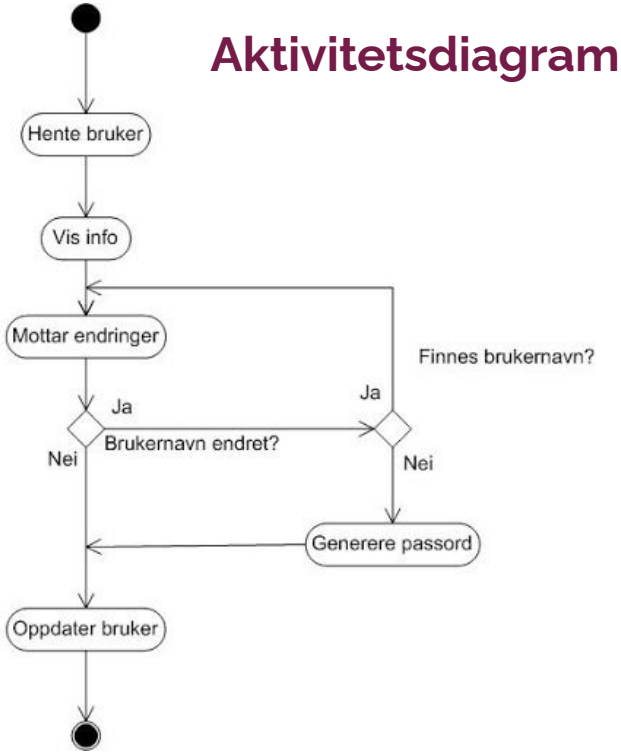
Use Case-diagram:

1.



Aktivitetsdiagram:

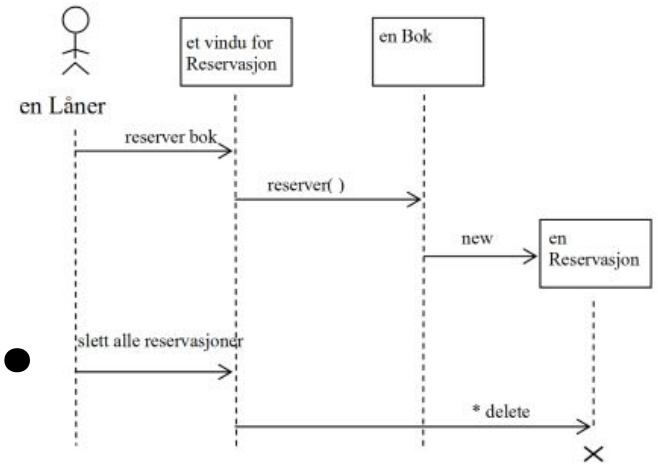
2.



Diagrammer for dette emnet

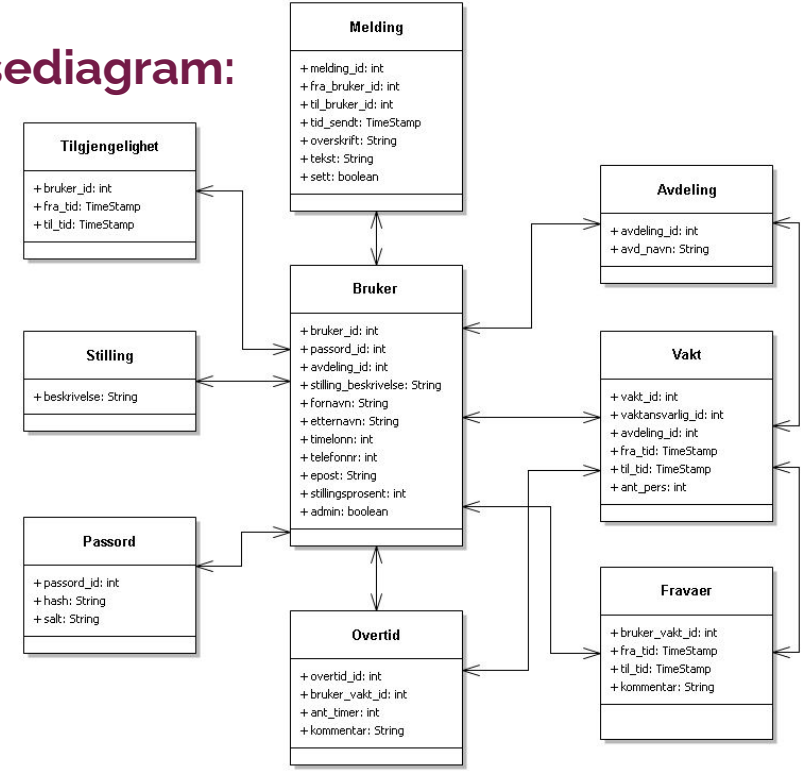
Sekvensdiagram:

3.

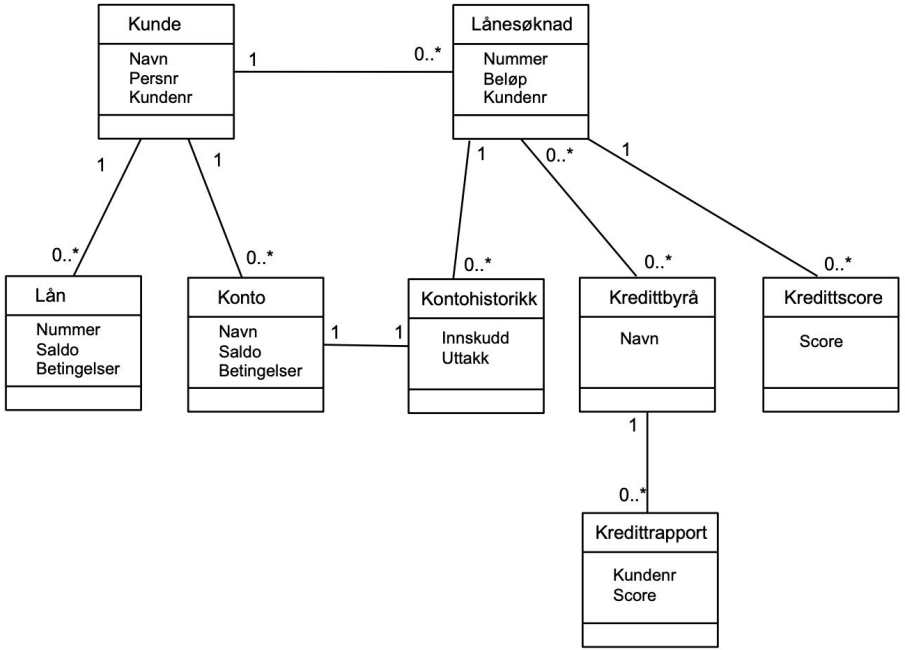


Klassediagram:

4.



Domenemodell - klassediagram uten metoder



Hvorfor lager vi så mange diagrammer?

- Vi ønsker å vite hva skal lage før vi lager det.
- Vi må kommunisere med kunden hva vi skal lage.
- Vi ønsker å finne et designpattern som alle utviklerne forstår, fordi vi samarbeider jo om koden.
- Vi ønsker at andre skal forstå systemet vi har laget, blant annet så de kan opprettholde det.

Designpattern

Modeller brukes (blant annet) for å skape felles forståelse for arkitekturen, slik at utviklere kan jobbe på hver sin enhet etter samme mønster - for så å "merge" enhetene sammen til et stort system.

Data-motiverte modeller:

Viser input data’s prosess mot output. Hjelper oss å forstå hva som skjer i prosessen fra input til output.

Hendelses-motiverte modeller:

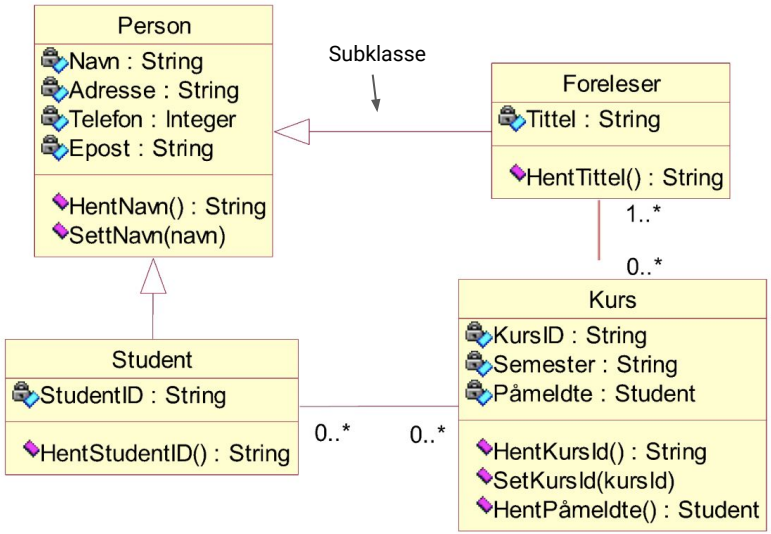
Viser systemets respons til eksterne og interne hendelser. Antagelse: systemet har et endelig antall “status-steg” som aktiveres av gitte hendelser, noe som kan skape overgang fra et status-steg til neste.

Strukturmodeller

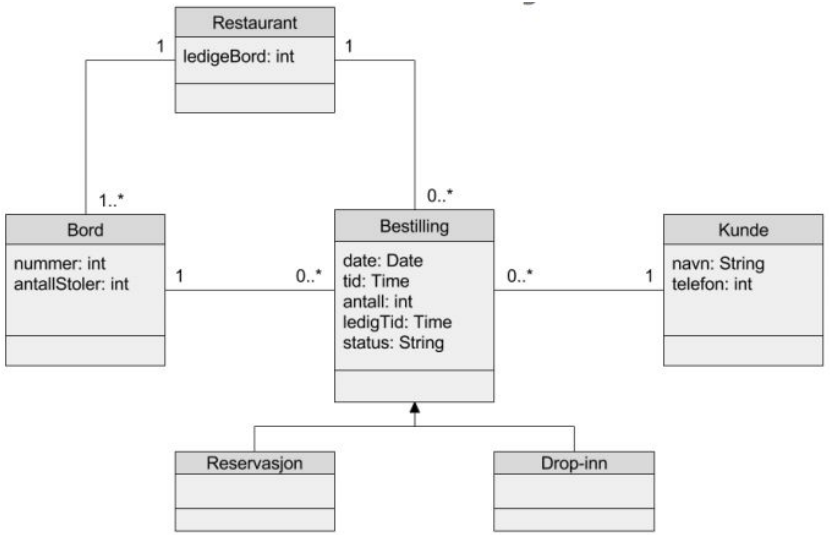
Klassediagram

- Strukturmodell som viser **strukturen i et system**
- Klassediagram → viser **objektklassene** i systemet og **assosiasjonene** mellom disse klassene
- Objektklasse: kan tenkes som en generell definisjon av et systemobjekt
    - Illustreres som en boks som inneholder navn, variabler og metoder.
- Assosiasjon: en link mellom klasser som indikerer at det er en relasjon mellom dem
    - Multiplisitet
        - ➢ 1 nøyaktig én
        - ➢ 0 .. 1 null eller én
        - ➢ * eller 0 .. * null eller mer
        - ➢ 1 ..* én eller mer ➢ n nøyaktig én

Klassediagram



Domenemodell

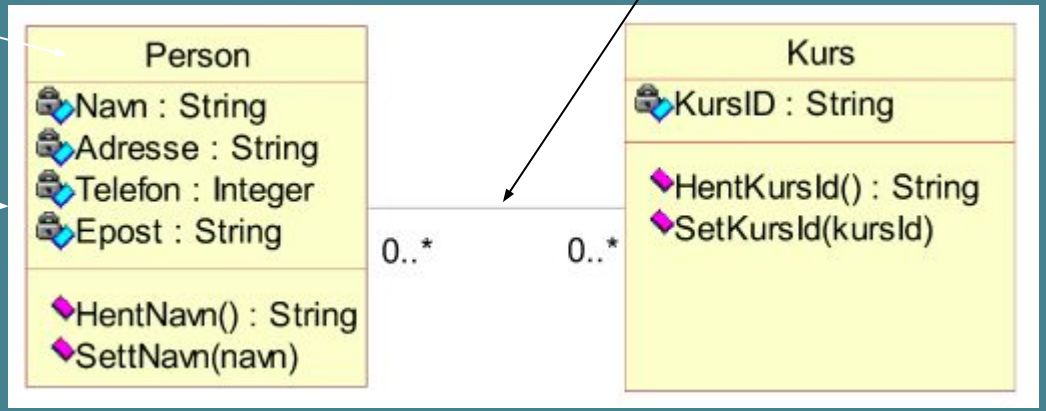


Notasjon

Relasjon mellom objekter

Navn på klasse

Attributter



Funksjoner og metoder

Modularisering

Arkitekturprinsipp!

- **Kohesjon** er et mål på hva slags ansvar et objekt har og hvor fokusert ansvaret er.
  - Mål: objekter som har moderat ansvar og utfører et begrenset antall oppgaver innenfor ett funksjonelt område.

- **Kobling** er et mål på hvor sterkt et objekt er knyttet til andre objekter.
  - Mål: Objekter med begrenset antall avhengigheter.

Eksempel

Høy kohesjon:
Et objekt skal kun ha ansvar for relaterte ting til det objektet

Lav kobling:
Et objekt skal samarbeide med et begrenset antall andre objekter

Atferdsmodeller

Aktivitetsdiagram

- Grafisk representasjon av **arbeidsflyt**
- **Aktiviteter** og tilhørende **handlinger** (actions)
- Viser overordnet **kontrollflyt**
- Beskriver hvordan mulige **utfall** av en **aktivitet påvirker flyten**
- Viser hvilke **aktiviteter** som kan utføres **parallelt**

OBS: Boken nevner aktivitetsdiagram som både Kontekstmodell og Adferdsmodell

Notasjon

Start: angir hvor flyten starter

Full sirkel

Slutt: angir hvor flyten ender

Full sirkel med ring rundt

Aktiviteter: angir ulike aktiviteter som inngår i arbeidsflyten

- Representeres med navngitte, avrundede rektangler
- Kan være fysisk (“godkjenn søknad”) eller elektronisk (“vis kjøpshistorikk”)

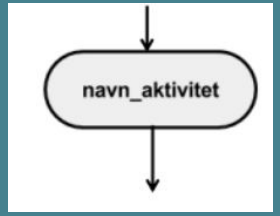
Valg: angir at man står ovenfor et valg (decision)

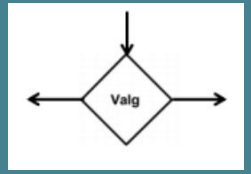
Eksempel: IF, IF-ELSE, CASE, osv.

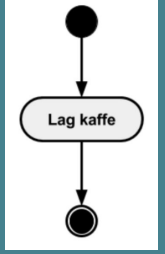
Valgdiamant

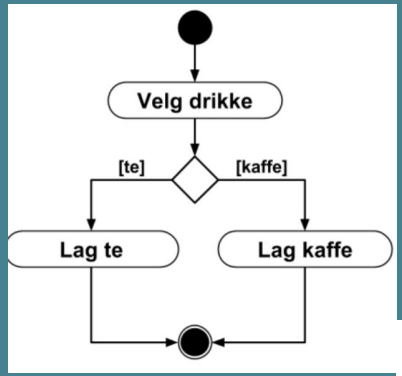
Blokkeringer (bar):

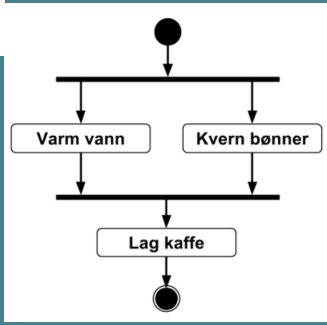
- Representerer start (split) og slutt (join) for parallelle prosesser
- Viser hvilke prosesser man må vente på, før man kan gå videre



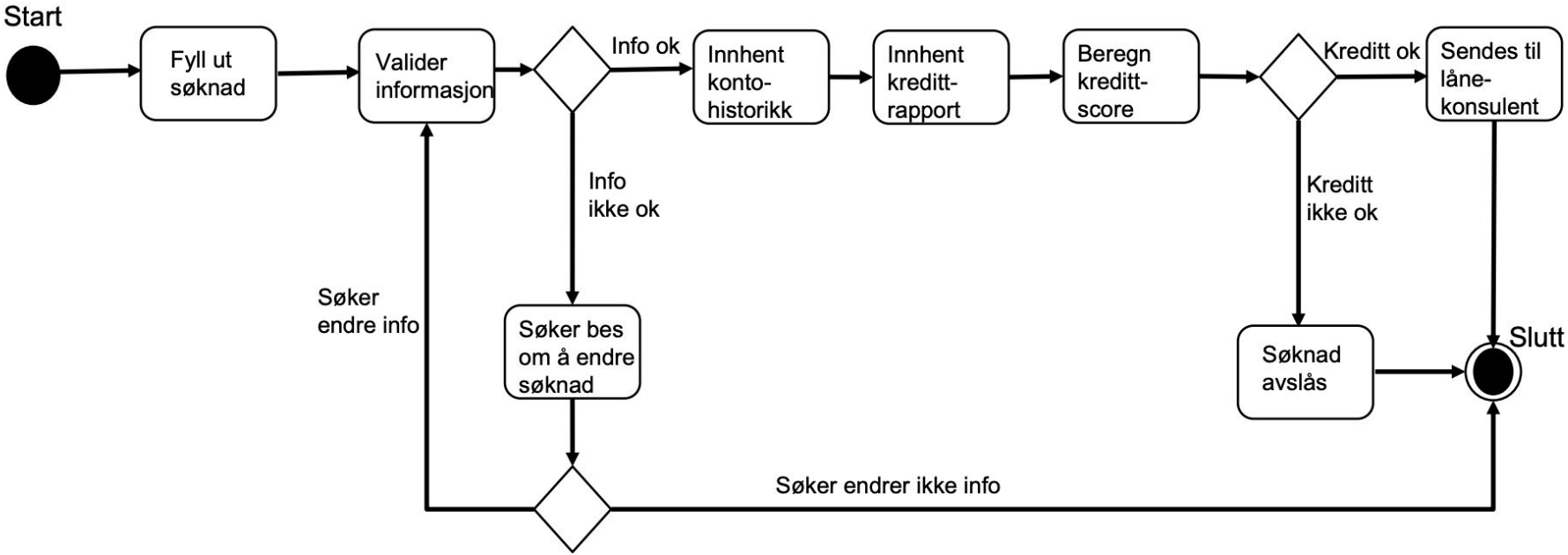




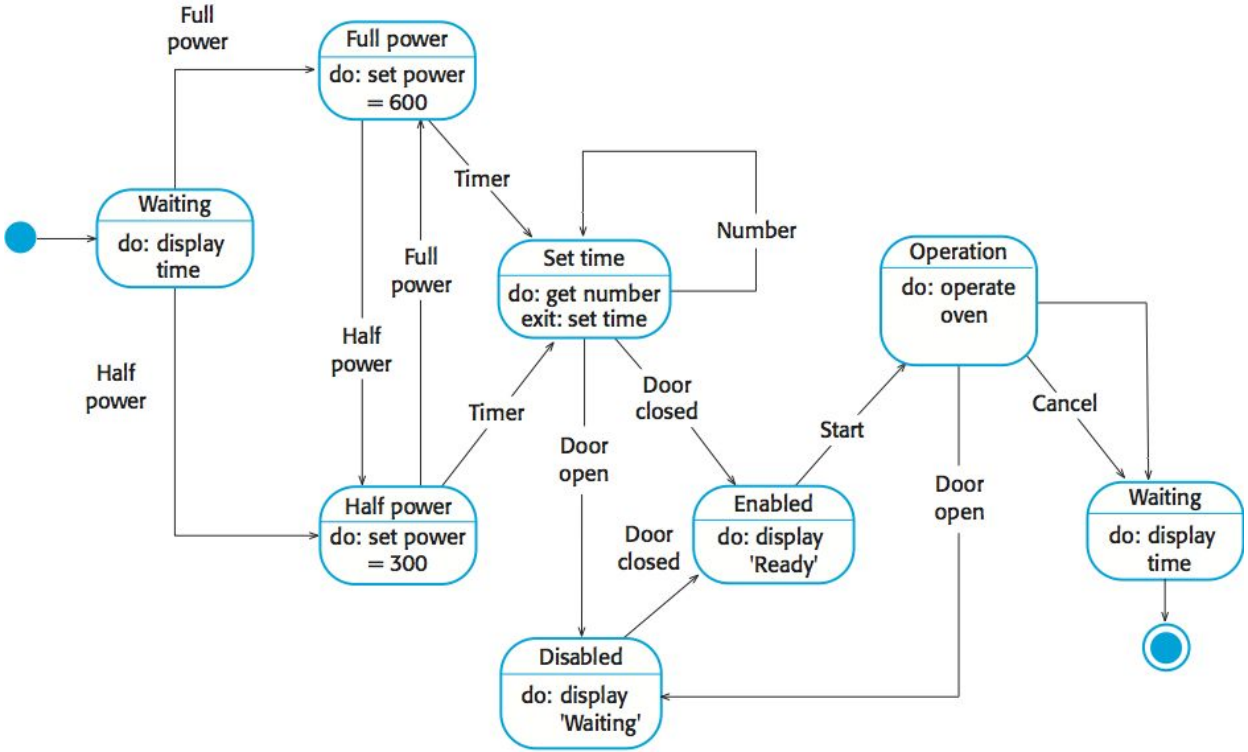




Aktivitetsdiagram



Aktivitetsdiagram



Spørsmål om UML diagrammer?

Ukesoppgaver

Link til oppgaver

Pensum for denne uken

Kapittel 5: System Modeling

Kapittel 7: Design and Implementation

Etter påske - Informasjonssikkerhet

Foilere

Delkapittel 8.1-8.2: Dev testing, Test-driven development

Delkapittel 22.1: Risikohåndtering

Oblig 5 frist: fredag, 6.mai, kl. 23:59.

Andre fine videoer:

Aktivitetsdiagram

Sekvensdiagram

Use Case-diagram

Takk for i dag!

Har du spørsmål, så send endelig en mail på: nhmoller@uio.no