UML modellering

Jamila & Julia

Plan for timen

- Hva og hvorfor modellering?
- Teori, notasjon og anvending i prosjektarbeidet
  - Use case diagram
  - Sekvensdiagram
  - Klassediagram
  - Aktivitetsdiagram

Hva er UML?

Unified Modeling Language (UML) er en industristandard for datarelatert modellering og er en metode for å visualisere og designe programvaresystemer.

Gir en standardisert måte å beskrive systemets struktur, oppførsel og interaksjoner mellom dets komponenter.

Utviklere og team kan kommunisere på en mer presis måte om design og implementasjon av systemet, og dermed redusere risikoen for feil og misforståelser.

Når bruker vi UML?

I kravanalysen

I designprosessen

Etter implementasjon

Når vi vil:

...ha felles forståelse

...beskrive tenkt system

...dokumentere eksisterende system

Hvorfor modellerer vi?

UML - hvorfor modellerer vi?

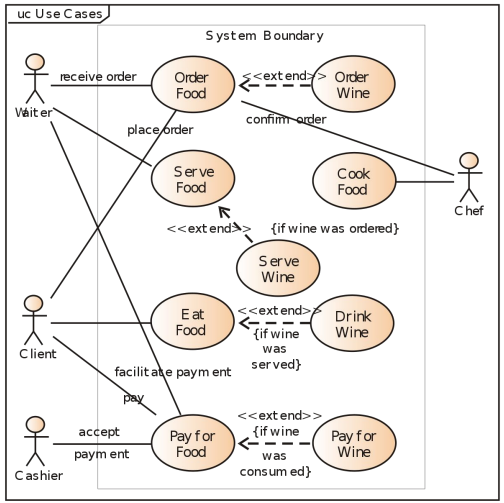
- I systemutvikling er modellering prosessen hvor man utvikler abstrakte modeller av et system
- Det er viktig å forstå at èn systemmodell ikke er en komplett representasjon av systemet, men at den kun viser ett perspektiv
- ...derfor trenger man flere typer modeller som synliggjør ulike aspekter ved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

Hvilke 4 diagram skal dere kunne?

Diagrammer for dette emnet

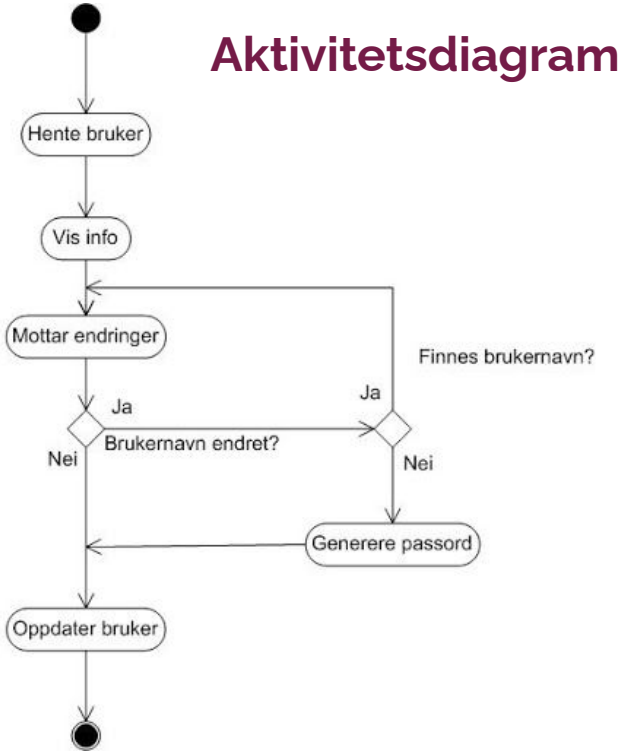
Use Case-diagram:

1.



Aktivitetsdiagram:

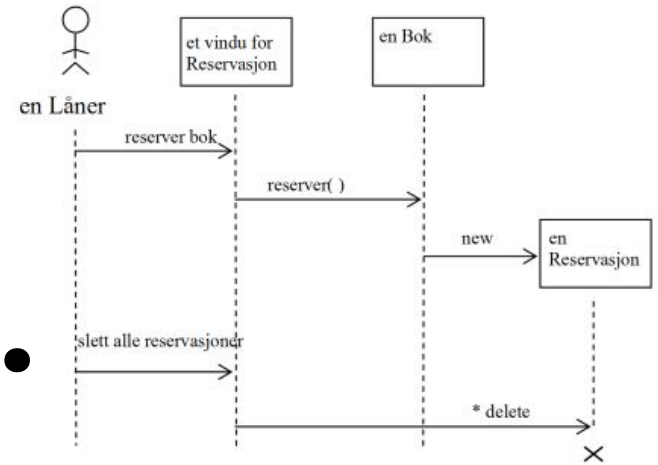
2.



Diagrammer for dette emnet

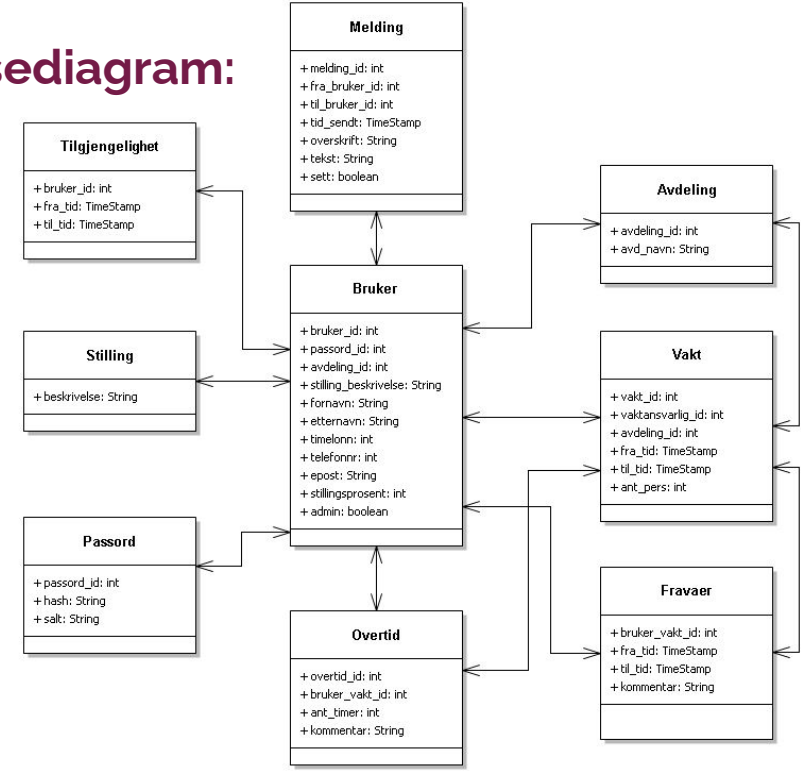
Sekvensdiagram:

3.



Klassediagram:

4.



Use case diagram: Viser interaksjon mellom et system og omgivelsene. Tar utgangspunkt i primæraktørs mål og hvordan sekundæraktører assisterer dette målet gjennom systemet.

Aktivitetsdiagram: Viser aktivitetsflyten i en prosess eller dataprosessering.

Sekvensdiagram: Viser interaksjon og informasjonsflyten mellom aktørene og systemet, og systemkomponentene i form av objektklasser.

Klassediagram: Viser struktur: objektklasser av et system, deres attributter og metoder, og assosiasjonene mellom klassene.

Hvorfor lager vi så mange diagram?

Vi ønsker å vite hva skal lage før vi lager det.

Vi må kommunisere med kunden hva vi skal lage.

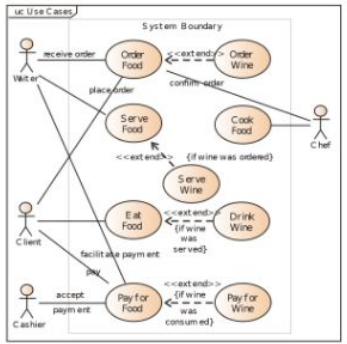
Vi ønsker å finne et *designpattern* som alle utviklerne forstår, fordi vi samarbeider jo om koden.

Vi ønsker at andre skal forstå systemet vi har laget, blant annet så de kan opprettholde det.

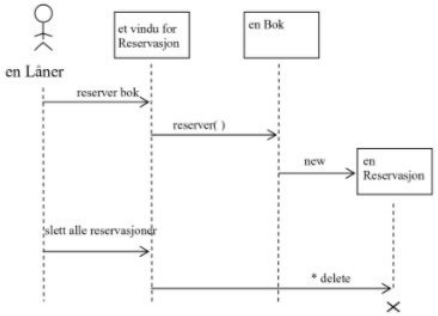
Designpattern?

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.

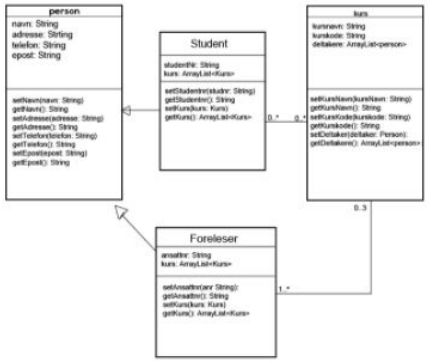
Use Case-diagram:



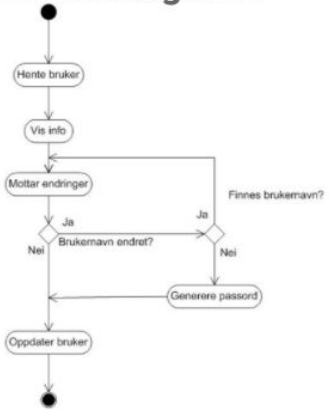
Sekvensdiagram:



Klassediagram:



Aktivitetsdiagram:



Interaksjonsmodeller

Strukturelle modeller

Atferdsmodeller

Interaksjonsmodeller

Use Case

Hvilke mål har primær aktøren? → Boble

Hvem hjelper til med å nå primær aktørs mål? → Sekundær aktør

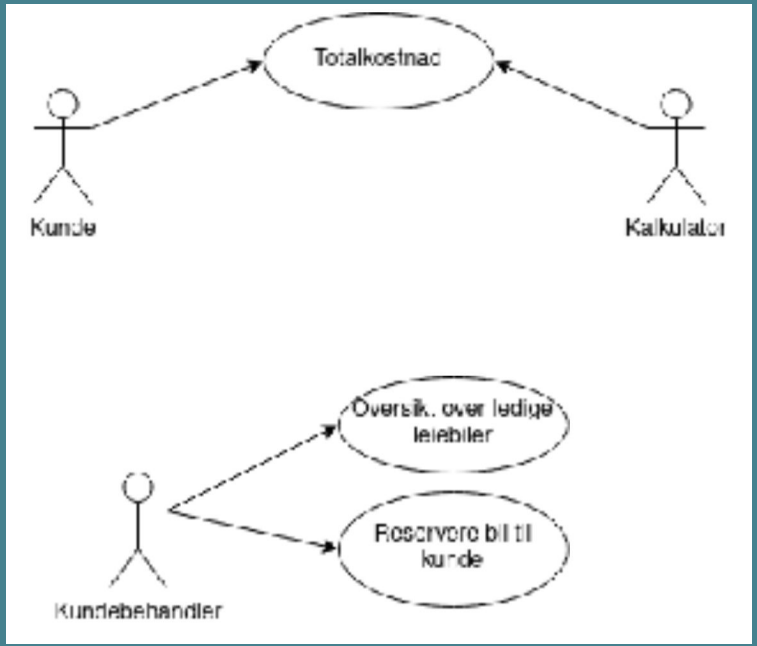
Fra brukerhistorie til Use Case diagram:

Format: Som [rolle] ønsker jeg [funksjon] for å oppnå [nytteverdi]

Som kunde ønsker jeg å vite totalkostnaden på billeien.

Som kundebehandler ønsker jeg å se hvilke biler jeg kan leie ut til kunden.

Som kundebehandler ønsker jeg å reservere bil til kunden.



<<include>> og <<extend>>

Include-relasjonen: Indikerer at et (sub) use case inneholder nødvendig funksjonalitet for gjennomførelsen av et annet basiscase.

Extend-relasjonen: Utvider oppførselen/funksjonalitet til et basiscase, som utføres under spesielle omstendigheter.

Aktør vs interessent...

Aktør: aktiv rolle, kommuniserer med systemet.

Interessent: kommuniserer ikke nødvendigvis med systemet.

Primær- vs sekundæraktør...

Primæraktør: eget ***mål*** i kommunikasjonen med systemet.

Sekundæraktør: trengs for at primær aktøren skal nå målet,
kommuniserer også aktivt med systemet.

Navn: Kjøp billett som eksisterende kunde (m.a.o. kunde med medlemskap)

Primæraktør: Kinokunde med medlemskap

Sekundæraktør: Betalingstjeneste (Nets)

Prebetingelse: Kunden har medlemsskap. Brukernavn er godkjent

Postbetingelser:

- PDF med billetter er generert, eller
- Use caset avlsuttes på annen måte

Hovedflyt:

1. Kinokunden skriver passord
2. Systemet sjekker om passordet er riktig
3. Passordet er korrekt, og systemet gir kunden tilgang til kontoen.
   (Om passord ikke godkjennes, trigges alternativ nr.1)
4. Systemet henter tilgjengelige filmvisninger
5. Systemet viser lista til kunden
6. Kinokunden velger filmvisning
7. Systemet finner ledige seter for visningen
8. Systemet viser de ledige setene på et kart til kunden, med prisoversikt
9. Kunden trykker på setene hen vil ha.
   (Om ønska sete(r) ikke er ledige, trigges alternativ flyt nr.2)
10. Kunden velger type billett for hvert sete, her også spesial eller ikke (alle seter har spesialfunkjson som kan aktiveres om det blir betalt for)
11. Systemet beregner totalpris for billettene (normal)
12. Kunden bekrefter og går videre
13. Systemet sjekker om kunden har poeng
14. Kunden har poeng. Regn (sum minus poengverdi) og lagre. (Differanse kan ikke være mindre enn 0, resterende poeng forblir på konto.)
15. Systemet sier “Prisen er x, men om du bruker poeng er prisen y. Vil du bruke poeng?” (Ja / nei valg)
16. Kunden velger ja
17. Pris settes til sum - poengverdi
18. Differanse > 0
    (Om sum == 0, trigges alternativ flyt nr.3)
19. Kinokunden velger betalingsmetode
20. Systemet sender kinokunden til betalingstjenesten
21. Betalingstjenesten sender beskjed til vårt system om at betalingen ble godkjent
    (Om ikke godkjent, trigges alternativ flyt nr.4)
22. Systemet gir medlemmet x antall poeng
23. Systemet genererer PDF med billettene

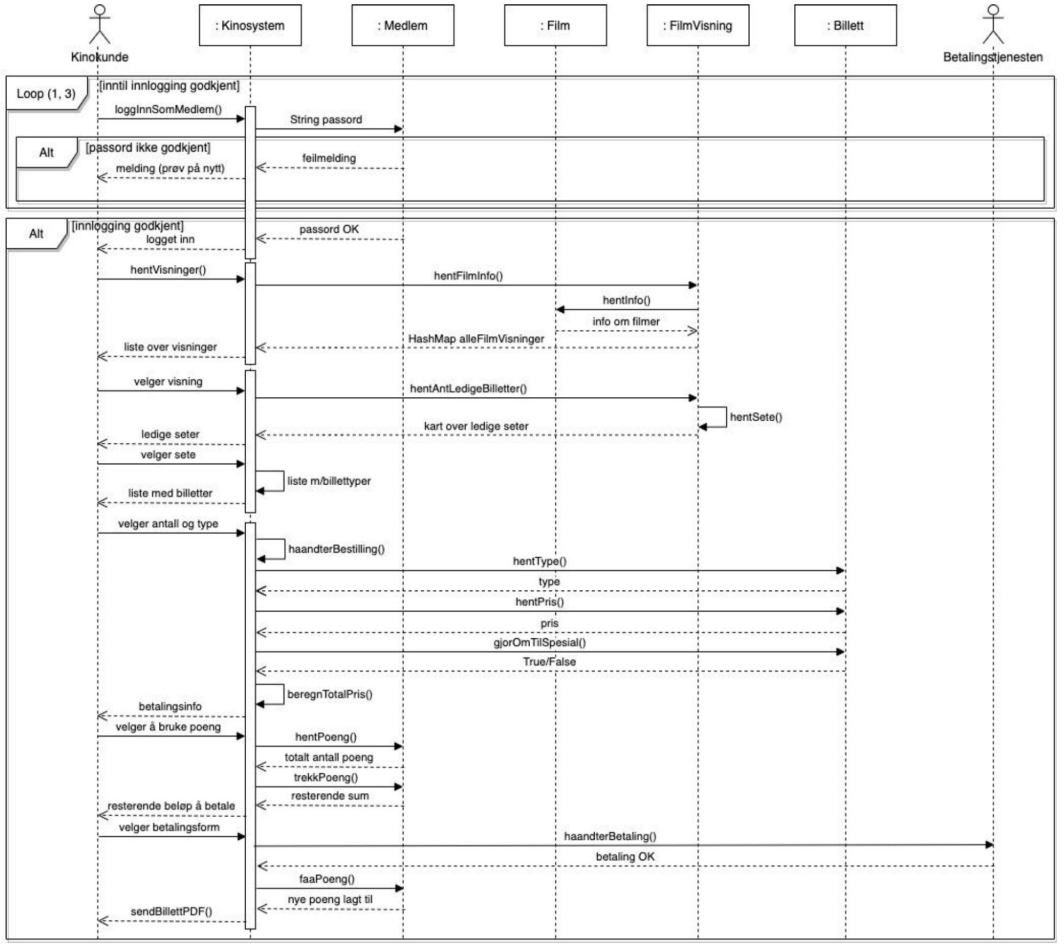
Alternativ flyt 1, steg 3:

3.1 Systemet godkjenner ikke passordet.

3.2.A1 Kinokunde sendes til steg 1, med en feilmelding, og får en ny sjanse til å logge inn.

3.2.A2.1 Kinokunden har brukt opp innloggingssjansene sine.

3.2.A2.2 Use caset (som innebærer at kinokunden logger inn) avsluttes.

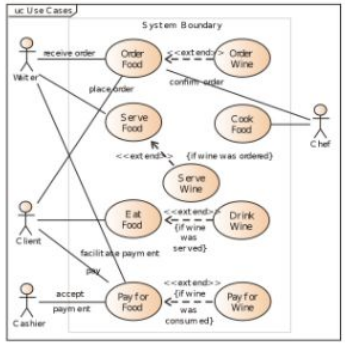


Tips til modellering av sekvensdiagram

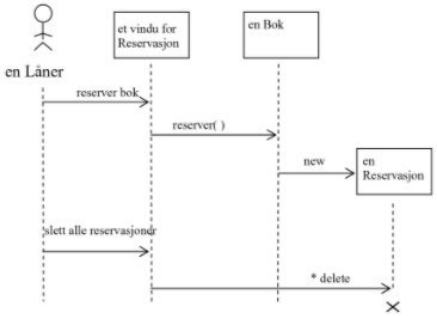
1. Identifiser de ulike aktørene/objektene.
2. Lag et tenkt, tekstlig oppsett basert på hovedflyt.
3. Modeller steg for steg, basert på stegene i hovedflyten.
4. Inkluder alternativ flyt etter at du har laget en modell for hovedflyten.

Strukturelle modeller

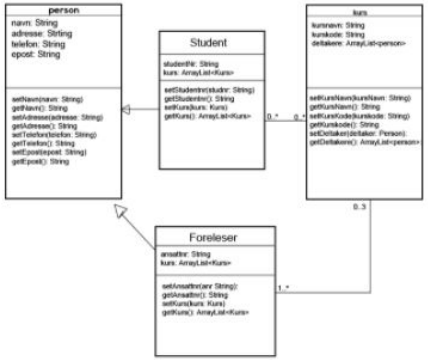
Use Case-diagram:



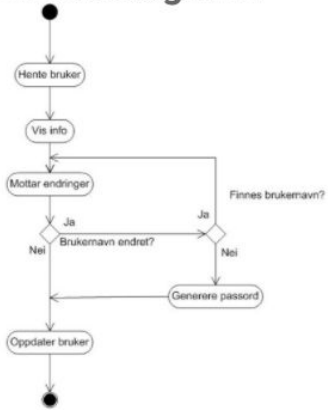
Sekvensdiagram:



Klassediagram:



Aktivitetsdiagram:



Interaksjonsmodeller

Strukturelle modeller

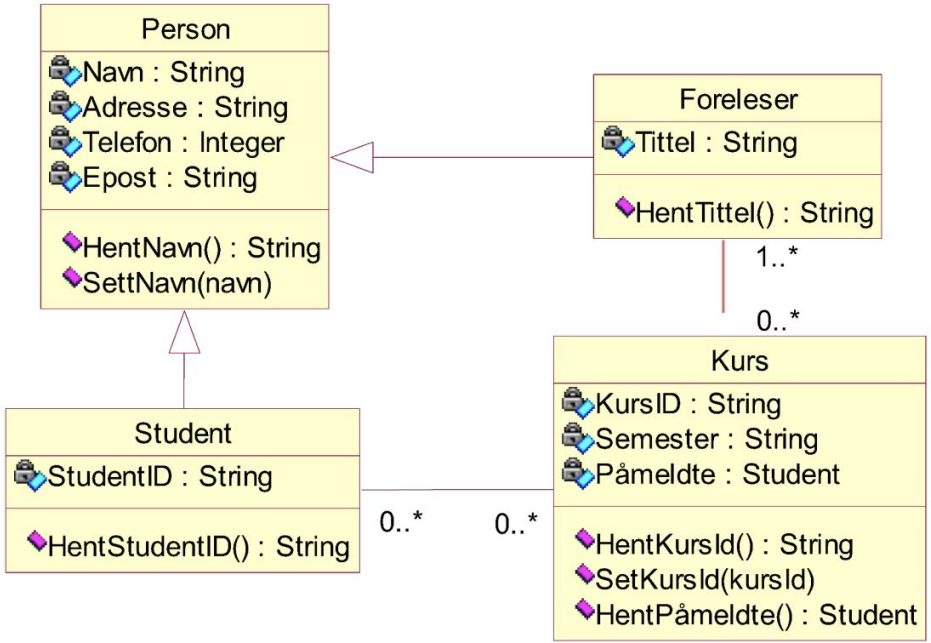
Atferdsmodeller

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 viser:

- Forhold mellom objektene
  - Én-til-én
  - Én-til-mange
  - Mange-til-én
  - Mange-til-mange
- Sub-klasser

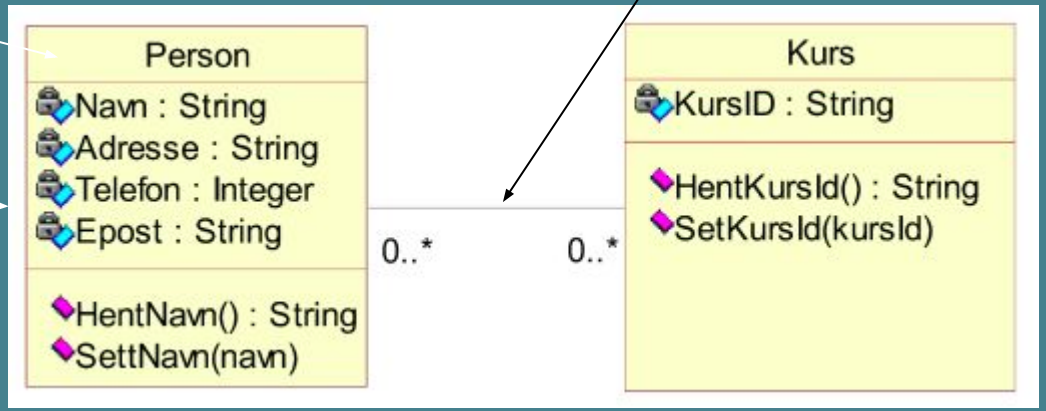


Notasjon

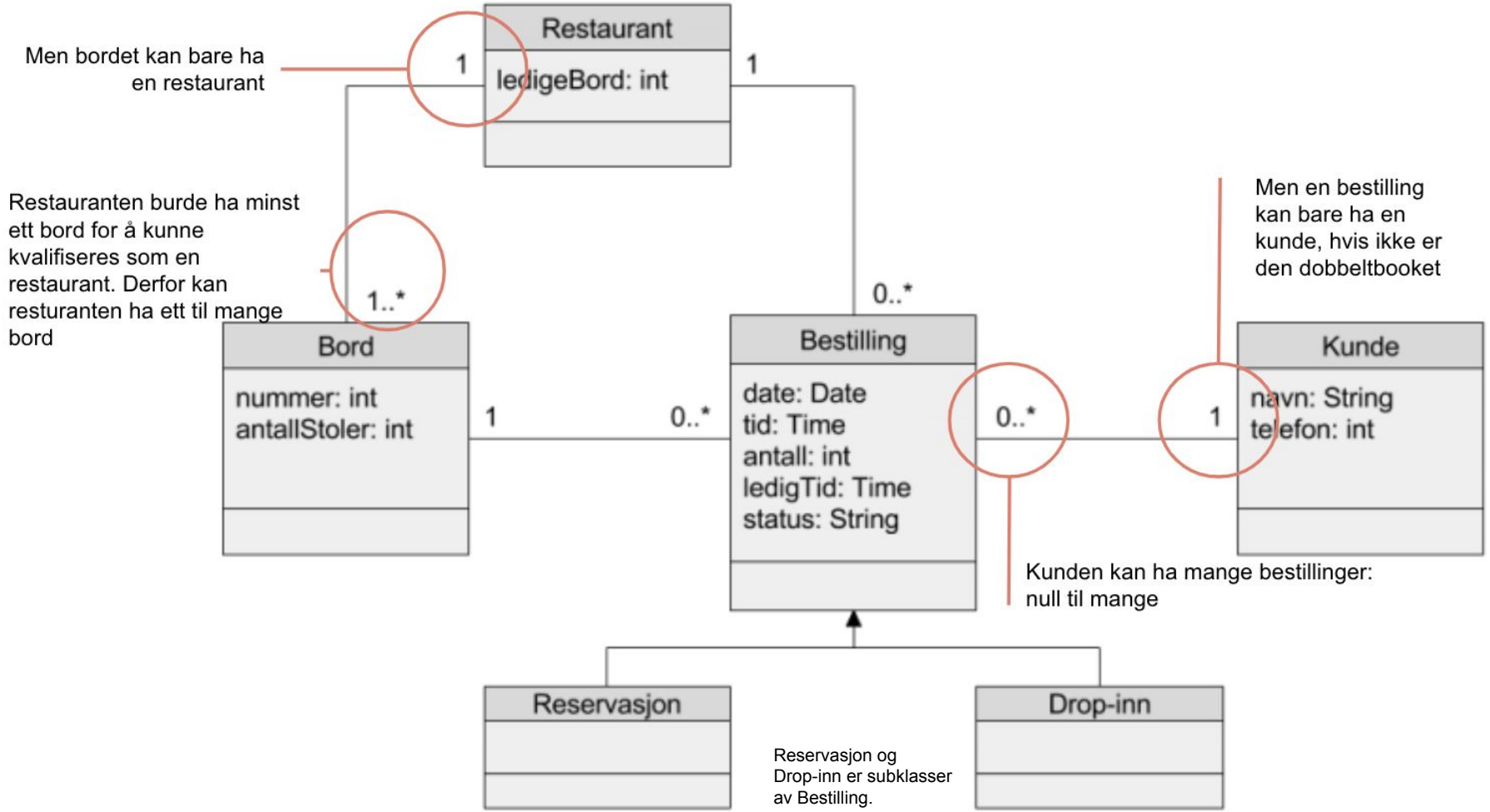
Relasjon mellom objekter

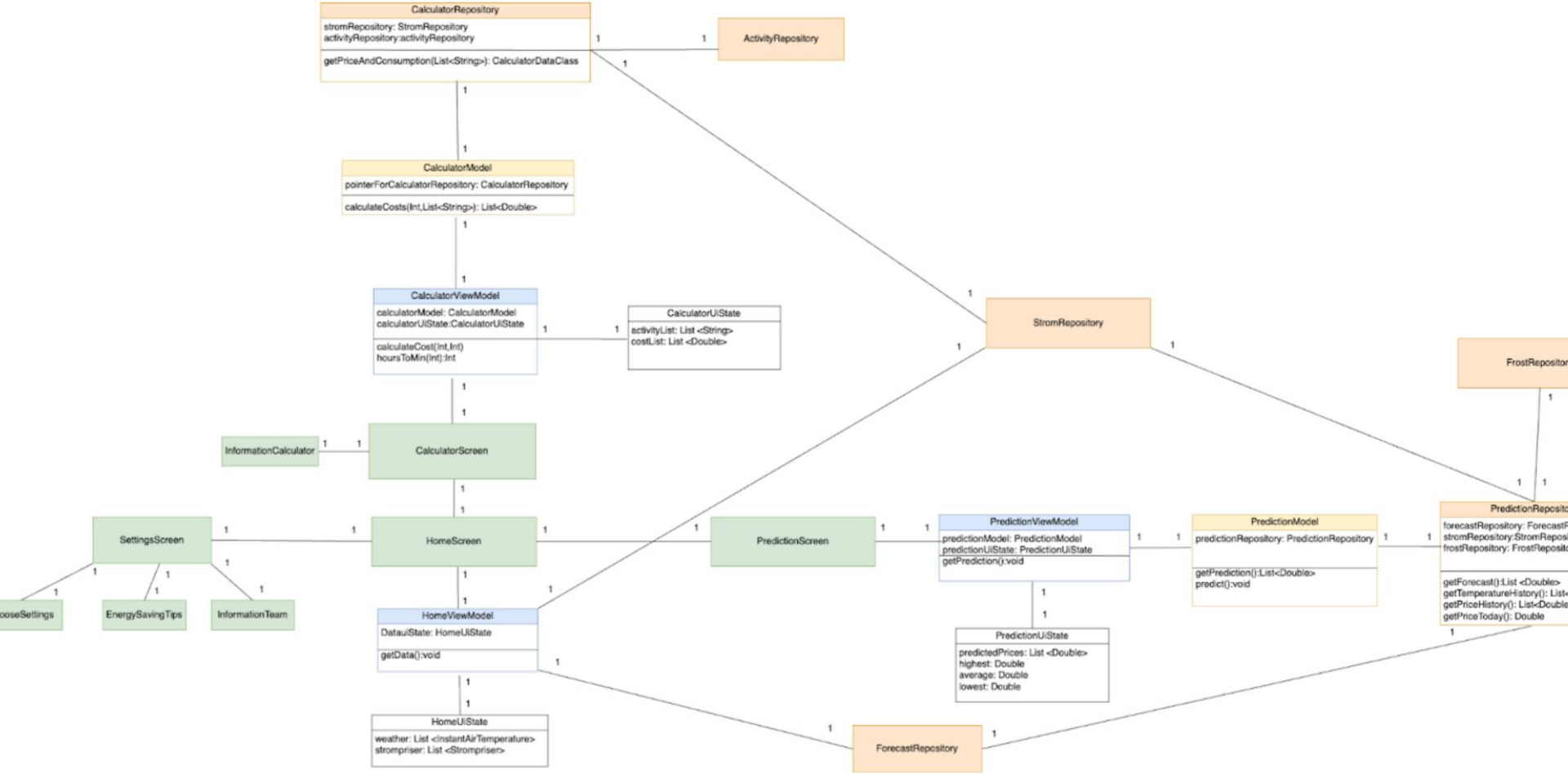
Navn på klasse

Attributter



Funksjoner og metoder


Men bordet kan bare ha en restaurant
Restaurant
ledigeBord: int
1
1
Restauranten burde ha minst ett bord for å kunne kvalifiseres som en restaurant. Derfor kan resturanten ha ett til mange bord
1..*
Bord
nummer: int
antallStoler: int
1
0..*
Bestilling
date: Date
tid: Time
antall: int
ledigTid: Time
status: String
0..*
Men en bestilling kan bare ha en kunde, hvis ikke er den dobbeltbooket
0..*
1
Kunde
navn: String
telefon: int
Kunden kan ha mange bestillinger: null til mange
Reservasjon
Drop-inn
Reservasjon og Drop-inn er subklasser av Bestilling.



Figur 19: Bilde av klassediagram. (Vedlagt er en større illustrasjon av diagrammet.)

Modularisering

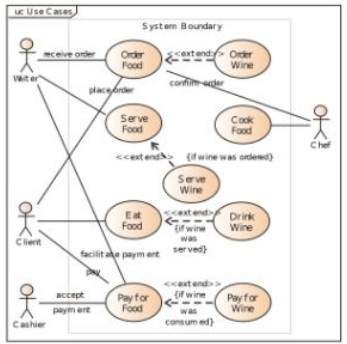
Arkitekturprinsipp!

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

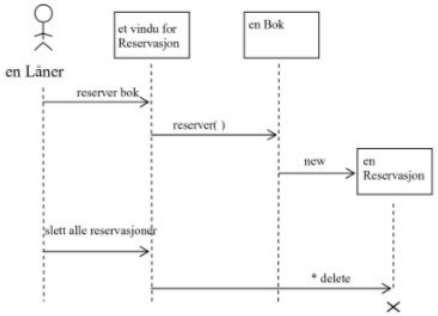
Lav kobling:
Et objekt skal samarbeide med et begrenset antall andre objekter

Atferdsmodeller

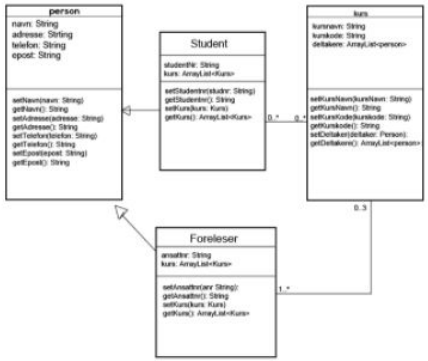
Use Case-diagram:



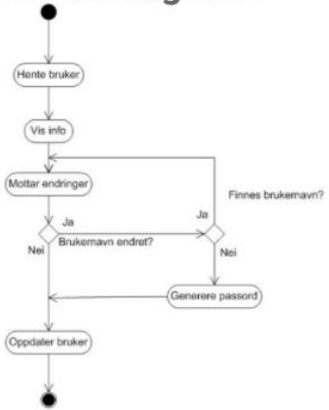
Sekvensdiagram:



Klassediagram:



Aktivitetsdiagram:



Interaksjonsmodeller

Strukturelle modeller

Atferdsmodeller

Aktivitetsdiagram

Grafisk representasjon av **arbeidsflyt**

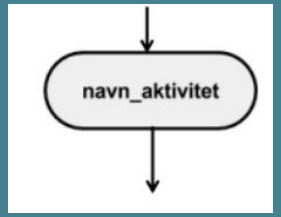
Aktiviteter og tilhørende **handlinger** (actions)

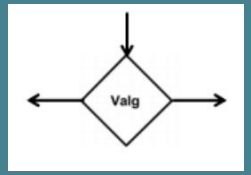
Viser overordnet **kontrollflyt**

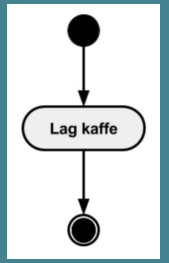
Beskriver hvordan mulige **utfall** av en **aktivitet**
påvirker flyten

Viser hvilke **aktiviteter** som kan utføres **parallelt**

Notasjon







Start: angir hvor flyten starter

Full sirkel

Slutt: angir hvor flyten ender

Full sirkel med ring rundt

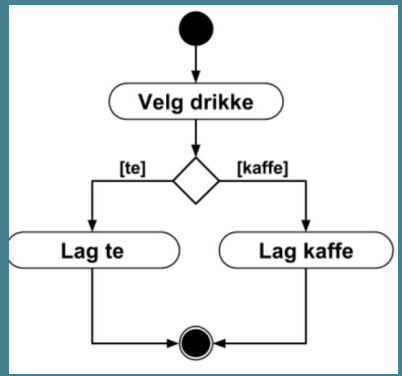
Aktiviteter: angir ulike aktiviteter som inngår i arbeidsflyten

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

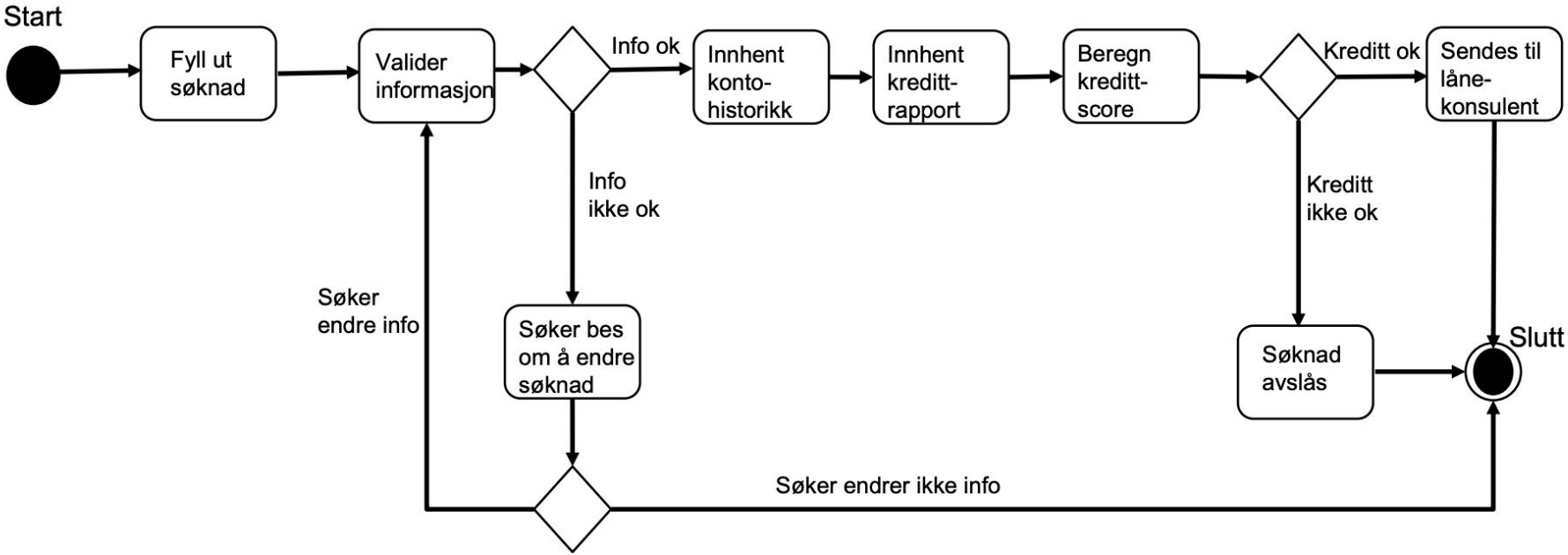
Valg: angir at man står ovenfor et valg (decision)

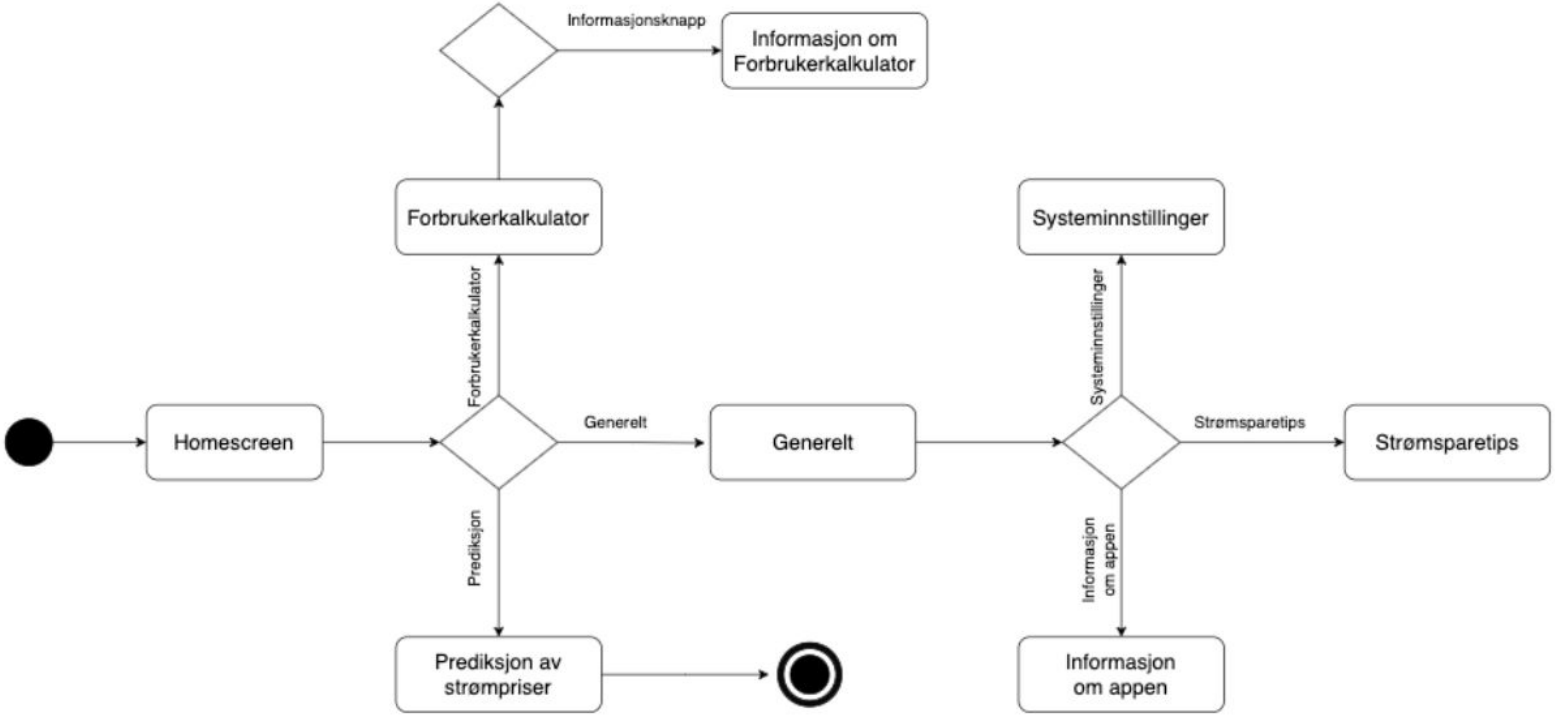
Eksempel: IF, IF-ELSE, CASE, osv.

Valgdiamant



Aktivitetsdiagram





Figur 11: Aktivitetsdiagram som viser navigasjon til ulike sider i applikasjonen, deriblant "Prediksjon av strømpriser". (Vedlagt er en større illustrasjon av aktivitetsdiagrammet)

Spørsmål om UML diagram?

Andre fine videoer:

Aktivitetsdiagram

Sekvensdiagram

Use Case-diagram

julialun@uio.no

jamilakm@uio.no