UKE 10 - UML modellering

IN1030 - Gruppe 2

Hva skal vi i dag?

- **UML-diagram & krav**
- **Use case diagram**
- **Ukesoppgaver**

Praktisk info om grupper

- Hvis noen i gruppen ikke melder seg/dere får ikke kontakt med: send en mail (emmatv@uio.no)
- Det kan være smart av dere og gjennomføre obligen smidig
  - da får dere bedre forståelse av prosessmodellene
  - blir enklere å svare på obliger
  - kanskje det hjelper til et smidig samarbeid innad i gruppen
  - Kanban: tar på dere oppgaver og gjør dem til dere er ferdige
  - Scrum: setter små tidsfrister (sprinter) denne uken skal man bli ferdig med sånn og sånn
    - kan ha stå-opp møter hvor dere forteller hverandre hvordan det går
    - ha et retrospektivt møte hvor dere kan evaluere om samarbeidet går bra så langt eller om dere må endre strategi

Menti

Hva er UML?

Unified Modeling Language (UML) er en industristandard for datarelatert modellering, forvaltet av et internasjonalt konsortium kalt Object Management Group (OMG). (Wikipedia).

Når bruker vi UML?

	Når vi vil:
I kravanalysen	*...ha felles forståelse*
I designprosessen	*...beskrive tenkt system*
Etter implementasjon	*...dokumentere eksisterende system*

Hvorfor lager vi UML diagram?

Fordi....

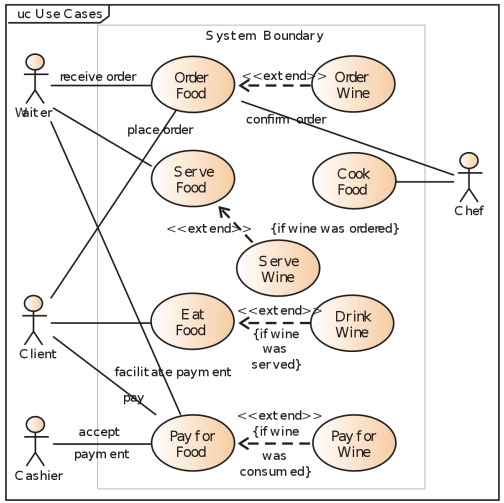
- ...vi skal identifisere krav
- ...vi skal være enige om hvordan endelig løsning skal bli
- ...vi skal kunne vedlikeholde systemet
- ...vi ønsker at andre skal kunne ta over vedlikehold av systemet
- ...vi skal kunne lage er godt system design

Hvilke 4 diagram skal dere kunne?

Hvilke diagram er dette?

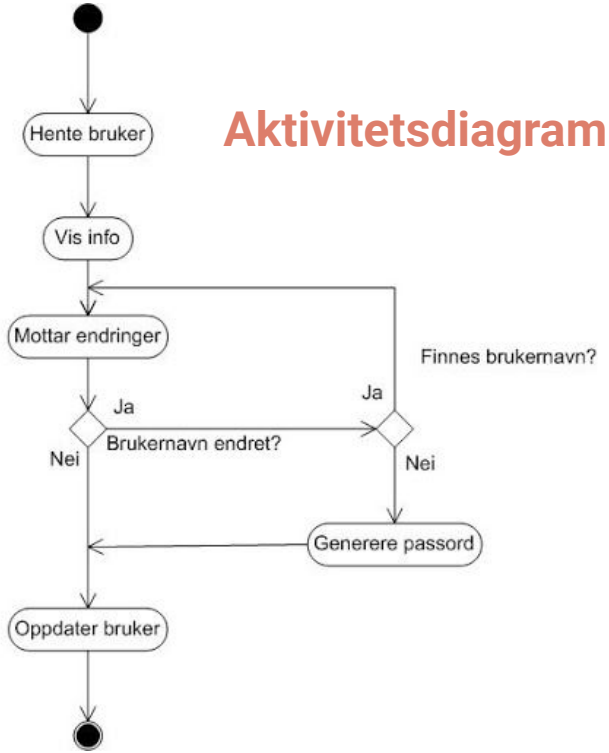
Use Case-diagram:

1.



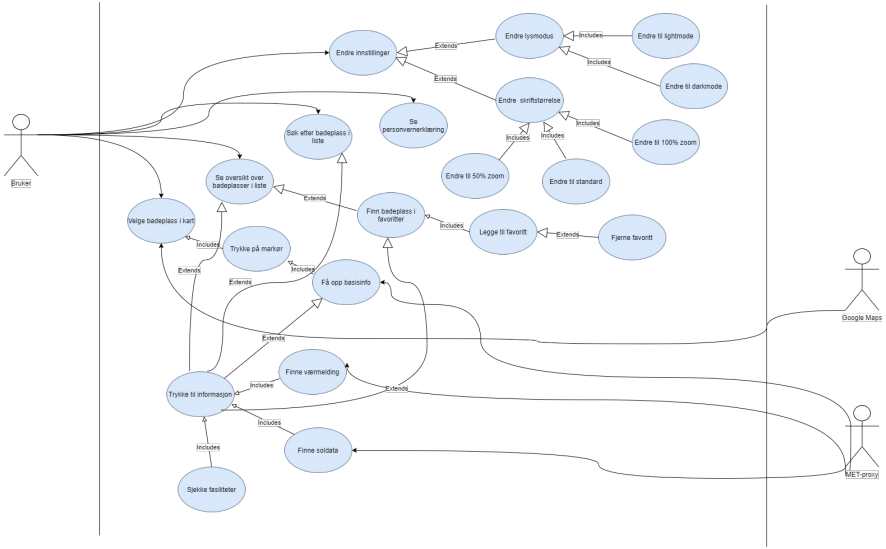
2.

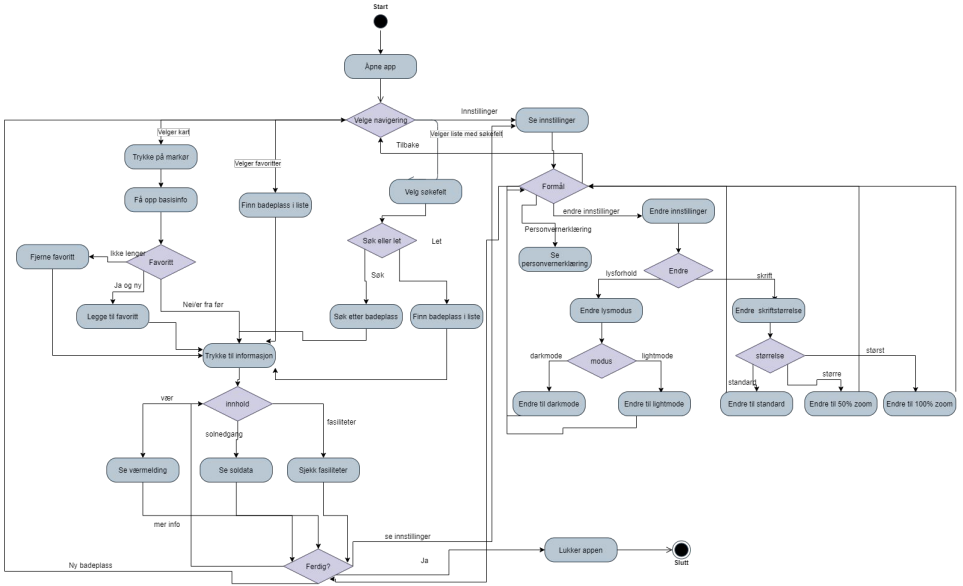
Aktivitetsdiagram:



Eksempel fra IN2000

OBS! Ikke bruk dette som en fasit, men som inspirasjon





Use case diagram

- Identifisere aktørens mål → et use case
- Interaksjonsperspektiv
- Hvordan systemet oppnår et mål av verdi for aktøren

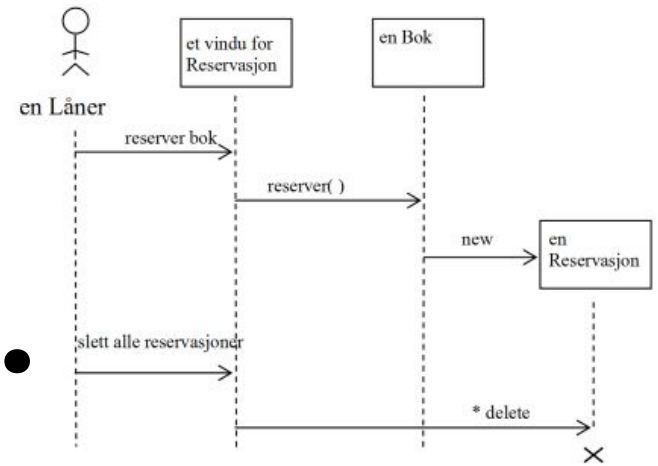
Aktivitetsdiagram

- Hendelsesflyten i et use case
- Flytskjema
- Viser aktiviteter og tilhørende handlinger

Hvilke diagram er dette?

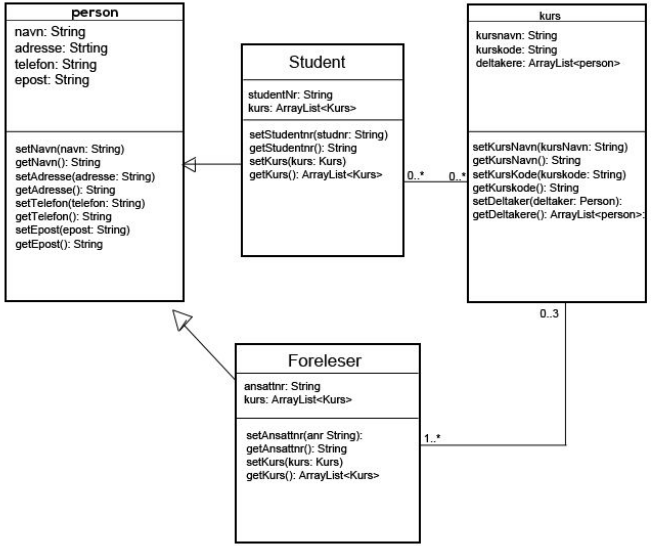
Sekvensdiagram:

3.



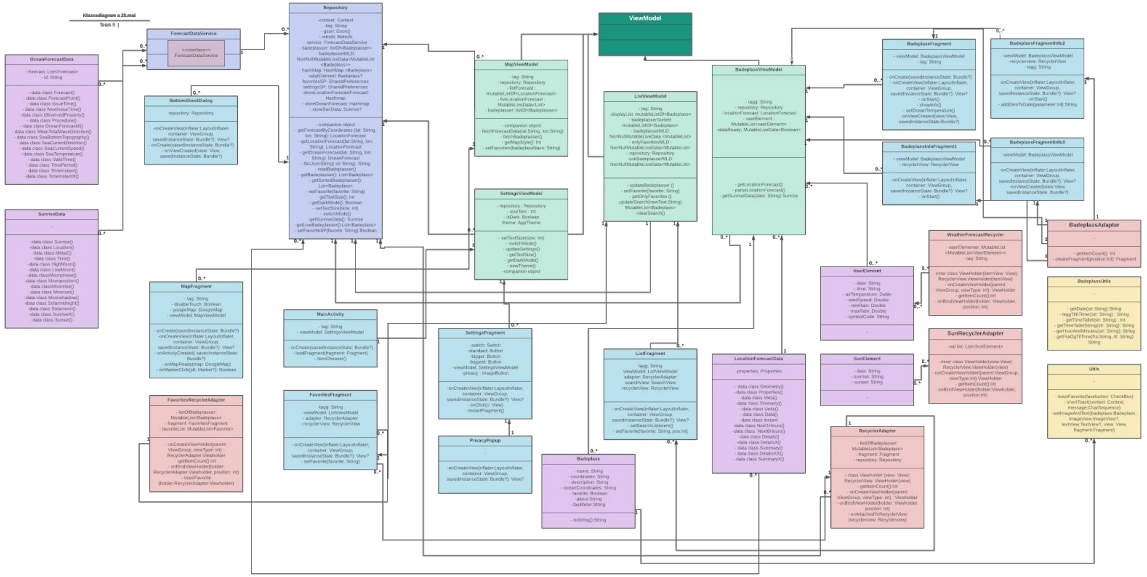
Klassediagram:

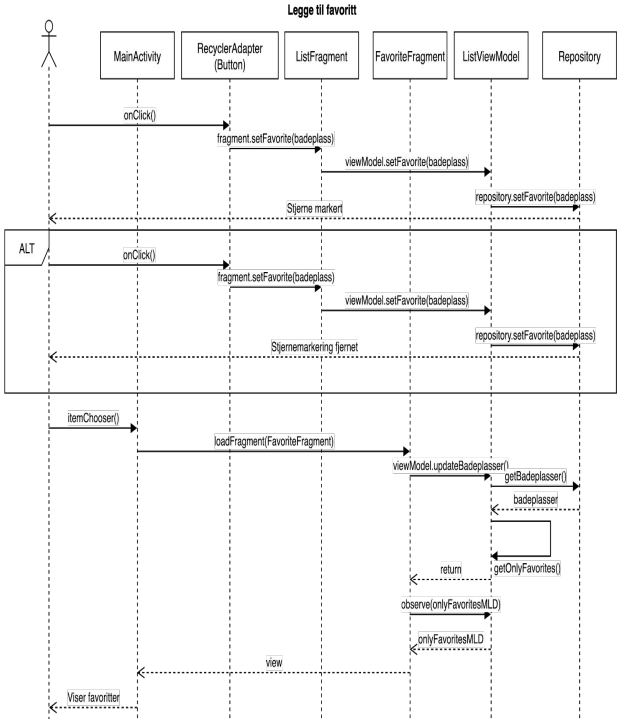
4.



OBS! Ikke bruk dette som en fasit, men som inspirasjon

Eksempel fra IN2000





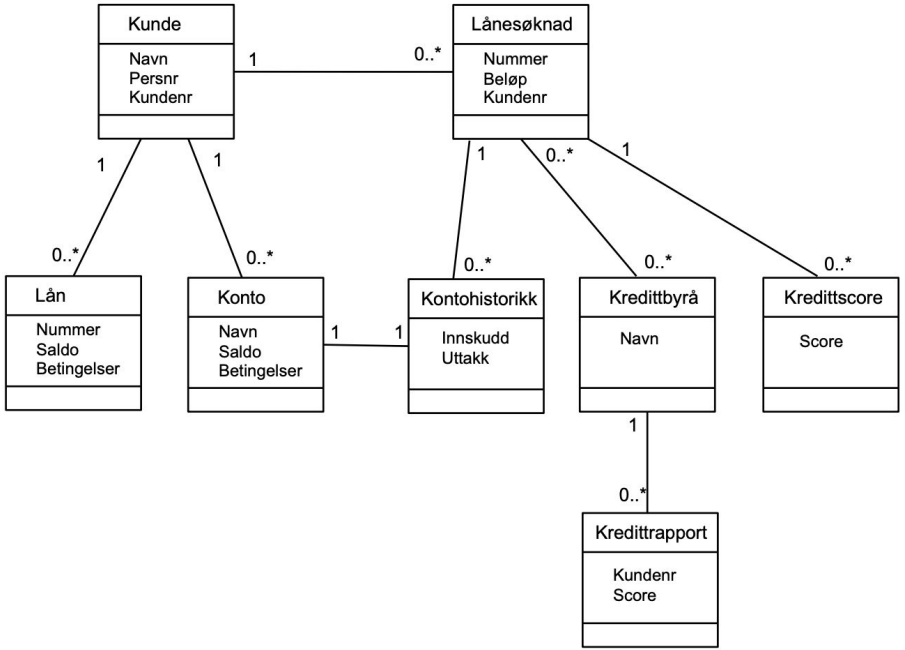
Klassediagram

- Kodenært
- Representasjon av objektene
- Se relasjoner mellom objektene
- Subklasser

Sekvensdiagram

- Flyt i et use case
- Kodenært
- Objekter
- Metodekall

Domenemodell - klassediagram uten metoder



Kort om “våre” modeller

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

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.

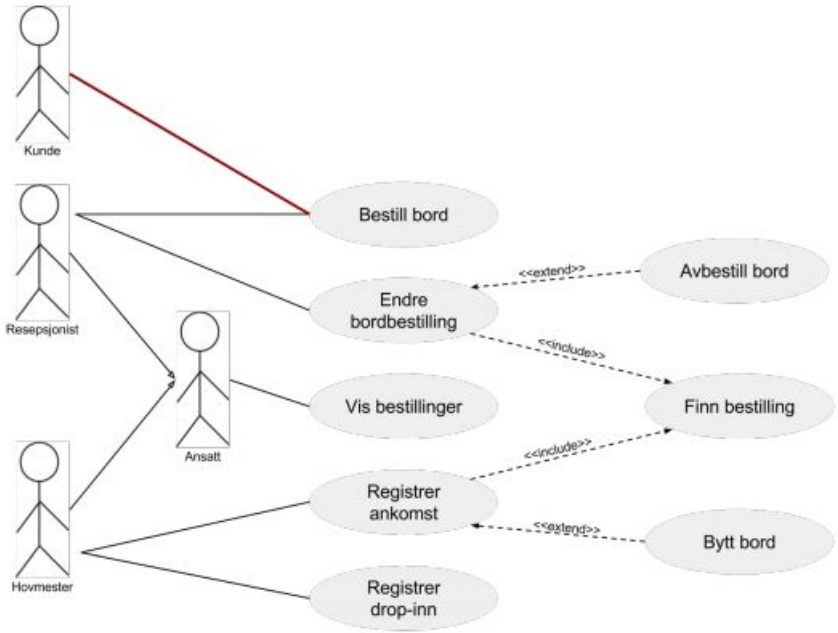
Aktivitetsdiagram: Viser aktivitetsflyten i en prosess eller dataprosessering.

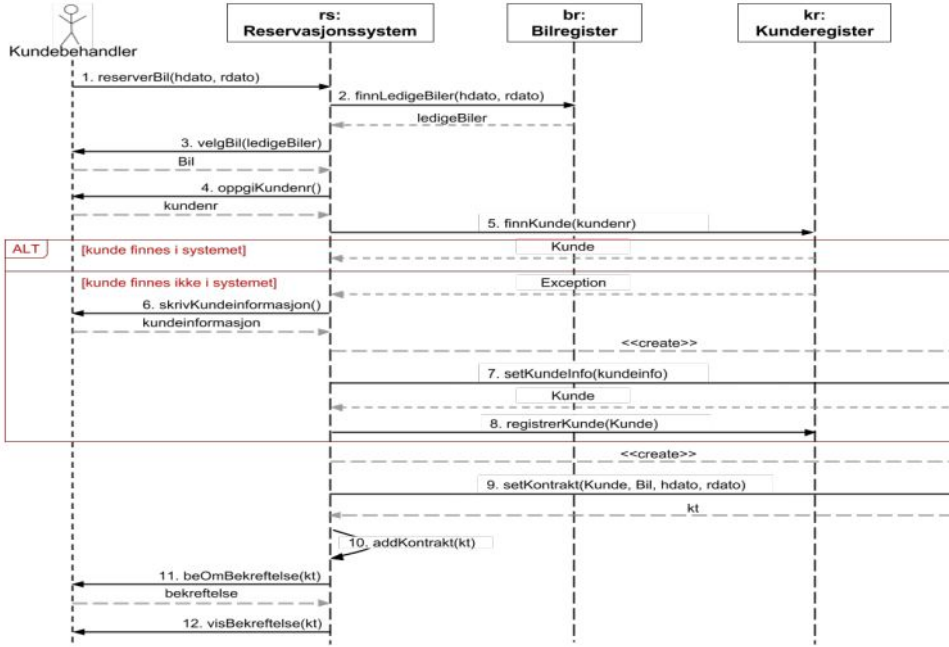
OBS: Det er viktig at alle modeller for et og samme system samsvarer !
En metode som blir brukt i et sekvensdiagram må også være med i et klassediagram.

Spørsmål?

Interaksjonsmodeller

Use Case diagram & Sekvensdiagram





Use-case

Aktør vs. interessent...

Aktør: aktiv rolle, kommuniserer med systemet.

Interessent: kommuniserer ikke nødvendigvis med systemet.

Primær vs sekundær aktør...

Primær aktør: eget **mål** i kommunikasjonen med systemet.

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

Use Case diagram

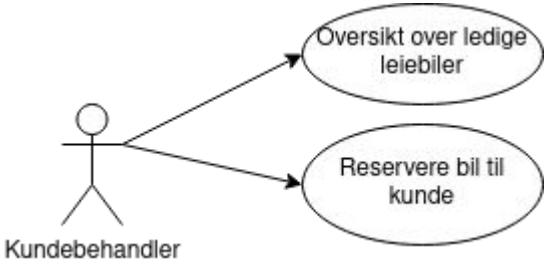
Hvilke mål har primær aktøren? → Boble

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

Fra brukerhistorie til Use Case diagram:
Som kunde ønsker jeg å vite totalkostnaden på å leie bilen slik at jeg vet om jeg har råd.

Som kundebehandler ønsker jeg å se hvilke biler jeg kan leie ut til kunden slik at jeg ikke dobbeltbooker.

Som kundebehandler ønsker jeg å reservere bil til kunden slik at jeg kan yte god kundeservice



Tekstlig beskrivelse av use case

Navn: Reserver bil
Primæraktør: Kundebehandler
Sekundæraktør: -
Prebetingelse: Ingen
Postbetingelse: Leiekontrakt for spesifisert bil og kunde med gitte utleiedatoer er opprettet

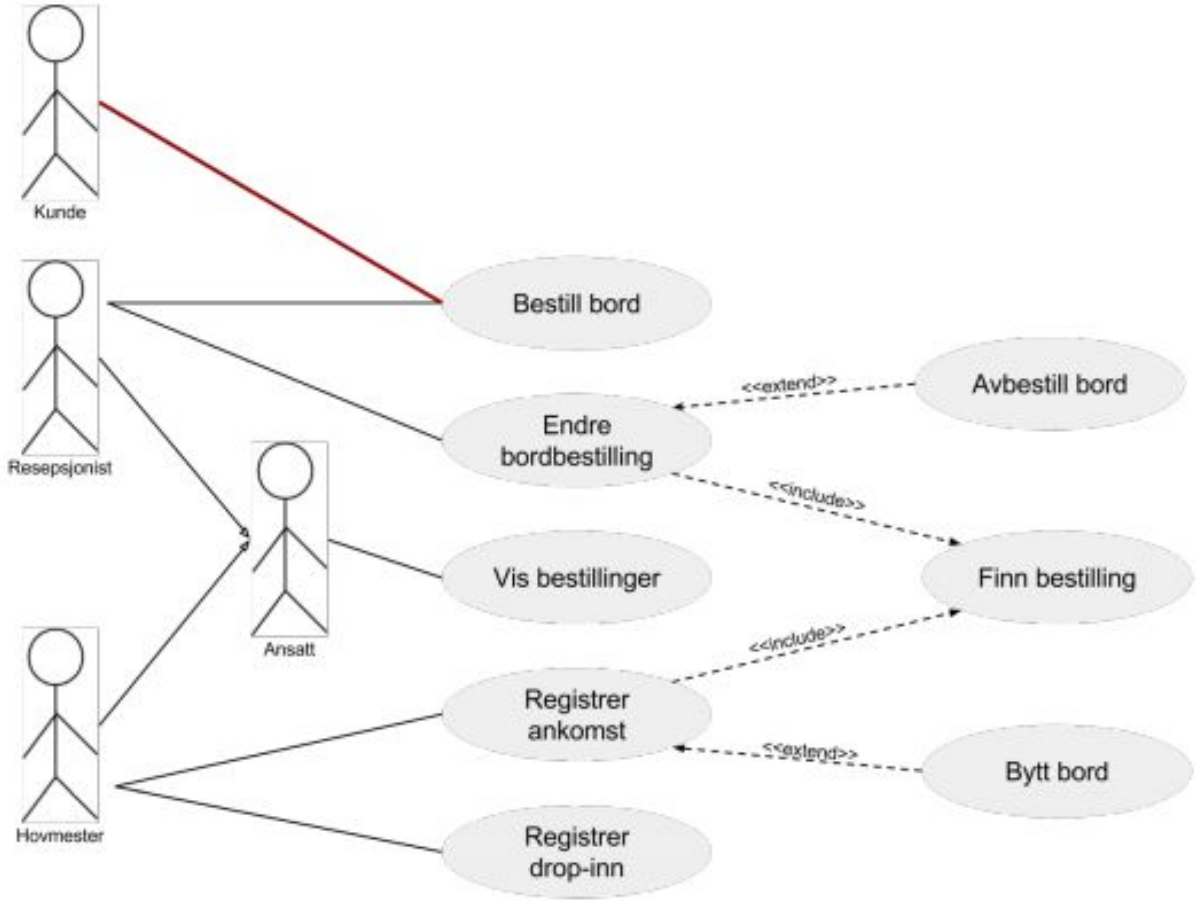
Hovedflyt:
1. Kundebehandler velger tidsintervall (hentedato og returdato)
2. Systemet returnerer en liste over tilgjengelige biler innenfor de spesifiserte datoene
3. Kundebehandler velger én av bilene.
4. Systemet ber om kundenr og finner kunden i systemet
5. Systemet oppretter ny kontrakt med kunde, og gitt periode
6. Systemet bekrefter reservasjonen

Alternativ flyt punkt :
4.1: Kunde finnes ikke i systemet
4.2: Systemet ber om kundeinformasjon
4.3: Kundebehandler skriver inn kundeinformasjon
4.4: Systemet oppretter en ny kunde med gitt informasjon
4.5: Systemet registrerer kunde i kunderegister og returnerer til steg 5

Include og extend relasjoner

- **Include relasjonen**
  - Et use case kan være en del av ett eller flere andre use case
  - Indikerer at et sub use case inneholder nødvendig funksjonalitet for gjennomførelsen av et annet basiscase
  - Holder seg til **hovedflyten**, men er avhengig av et annet use case sin funksjonalitet for å gjennomføre sin egen oppgave

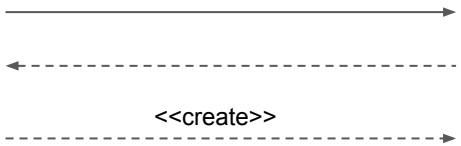
- **Extend relasjonen**
  - Et use case beskriver tilleggs oppførsel som utføres under gitte omstendigheter
  - Utvider oppførselen/funksjonaliteten til et basiscase, som utføres under spesielle omstendigheter
  - **Tenkes som alternativflyt**


Kunde
Resepsjonist
Hovmester
Ansatt
Bestill bord
Endre bordbestilling
Vis bestillinger
Registrer ankomst
Registrer drop-inn
Avbestill bord
Finn bestilling
Bytt bord
<<extend>>
<<include>>
<<include>>
<<extend>>

Sekvensdiagram

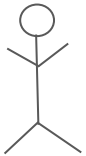
Sekvensdiagram notasjon

- **Metodekall** (med parametre)**:** heltrukket pil
- **Returverdi:** stiplet pil
- **Create** (for å opprette objekter): stiplet pil



- **Alternativ flyt:** settes i en “boks”

- **Aktører:** Strekmennesker



- **Klasser/Objekter:** Bokser

Tips til modellering av sekvensdiagram

1. **Identifiser de ulike aktørene/objektene.**
2. **Lag et tenkt, tekstlig oppsett basert på hovedflyt.**
    a. Jo mer detaljert, desto enklere blir det å modellere
3. **Modeller steg for steg, basert på stegene i hovedflyten.**
4. **Inkluder alternativ flyt etter at du har laget en modell for hovedflyten.**

Tips til modellering av sekvensdiagram

1: Identifiser de ulike **aktørene** & **objektene**:

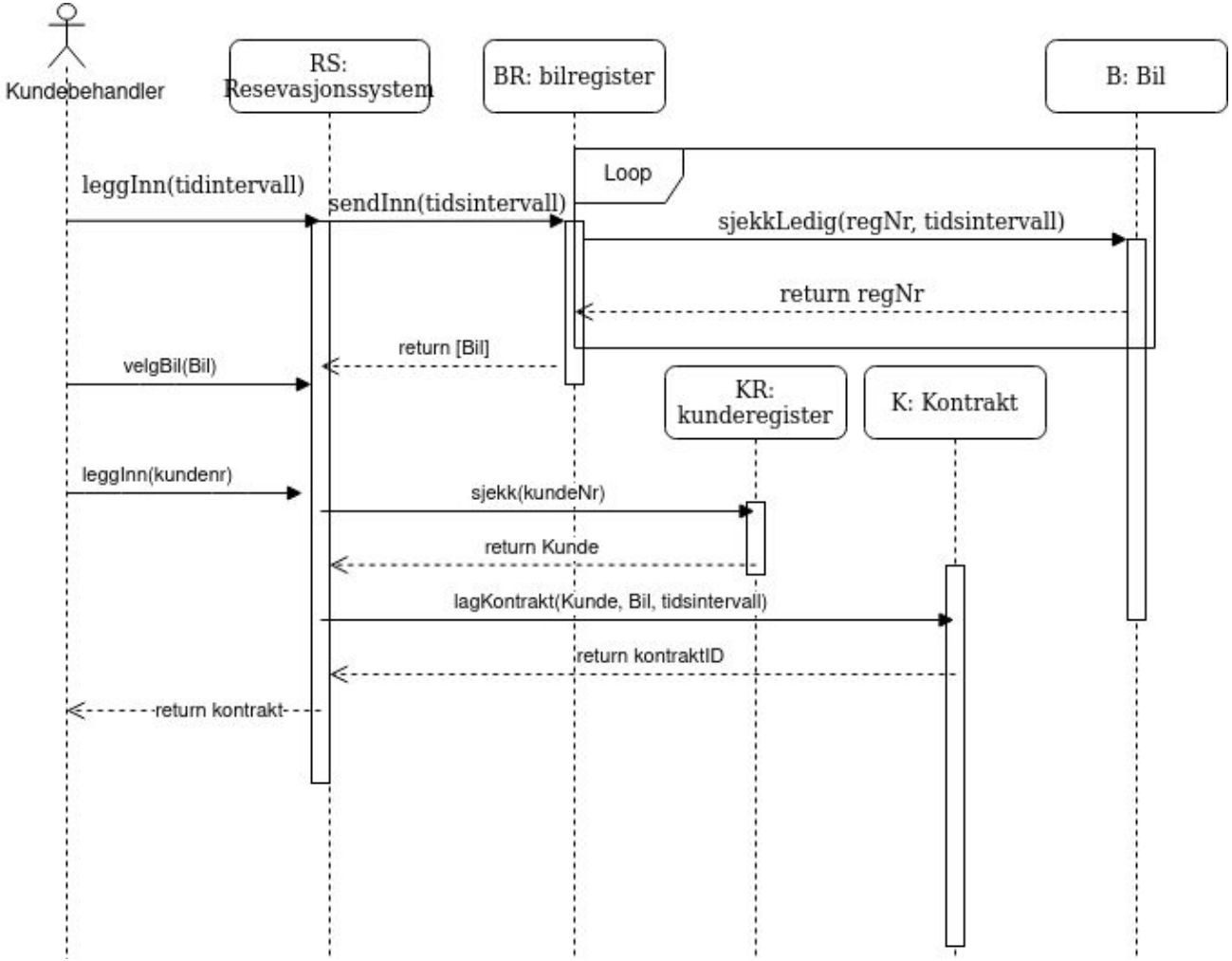
Hvilket system er det snakk om? – ***Reservasjonssystem*** *for biler*
Har vi eventuelle undersystemer? – ***Bilregister/Kunderegister*** *(avhengig av hvordan systemet er implementert)*
Har vi eventuelle objekter? – ***Kunde*** *(objekter for de ulike kundene)****/Kontrakt*** *(objekt for utleiekontrakt)*
Hvem skal interagere med systemet? – ***Kundebehandler***

Aktør: **Kundebehandler.**
Klasser/Objekter: **Reservasjonssystem, Bilregister, Kunderegister, Kunde, Kontrakt**.

Tips til modellering av sekvensdiagram

2: Lag et tenkt, tekstlig oppsett basert på hovedflyt:

- **Aktør (Kundebehandler) til venstre.**
- **Interagerer med Reservasjonssystemet når hen velger tidsintervall.**
- **Da interagerer Reservasjonssystemet med Bilregister og henter ut tilgjengelige biler.**
- **«Viser» tilgjengelige biler til kunde.**
- **Kundebehandler velger bil og «sender» dette til Reservasjonssystem.**
- **Reservasjonssystem ber om kundenr.**
- **Kundebehandler gir kundenr.**
- **Reservasjonssystem finner kunde i Kunderegister.**
- ***Kundeobjekt på plass eller være med i Sekvensdiagrammet? Må det registreres som alternativ flyt?***
- **Kontrakt blir laget med metode fra Kundeobjekt**


Kundebehandler
RS:
Resevasjonssystem
BR: bilregister
B: Bil
Loop
leggInn(tidintervall)
sendInn(tidsintervall)
sjekkLedig(regNr, tidsintervall)
return regNr
return [Bil]
velgBil(Bil)
KR:
kunderegister
K: Kontrakt
leggInn(kundenr)
sjekk(kundeNr)
return Kunde
lagKontrakt(Kunde, Bil, tidsintervall)
return kontraktID
return kontrakt

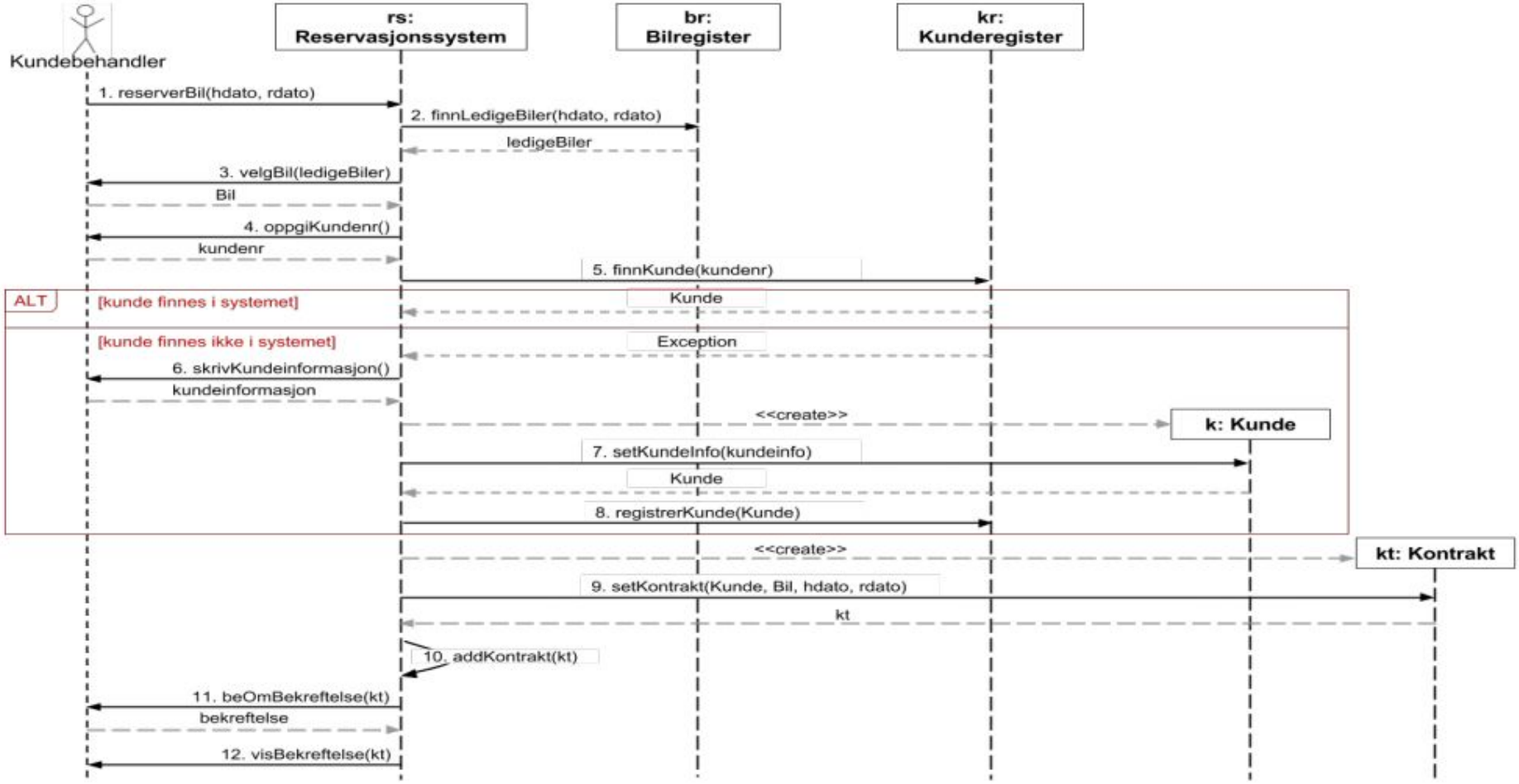
Tekstlig beskrivelse av use case

Navn: Reserver bil
Primæraktør: Kundebehandler
Sekundæraktør: -
Prebetingelse: Ingen
Postbetingelse: Leiekontrakt for spesifisert bil og kunde med gitte utleiedatoer er opprettet

Hovedflyt:
1. Kundebehandler velger tidsintervall (hentedato og returdato)
2. Systemet returnerer en liste over tilgjengelige biler innenfor de spesifiserte datoene
3. Kundebehandler velger én av bilene.
4. Systemet ber om kundenr og finner kunden i systemet
5. Systemet oppretter ny kontrakt med kunde, og gitt periode
6. Systemet bekrefter reservasjonen

Alternativ flyt punkt 2:
4.1: Kunde finnes ikke i systemet
4.2: Systemet ber om kundeinformasjon
4.3: Kundebehandler skriver inn kundeinformasjon
4.4: Systemet oppretter en ny kunde med gitt informasjon
4.5: Systemet registrerer kunde i kunderegister og returnerer til steg 5

Sekvensdiagram



Andre fine videoer:

Aktivitetsdiagram

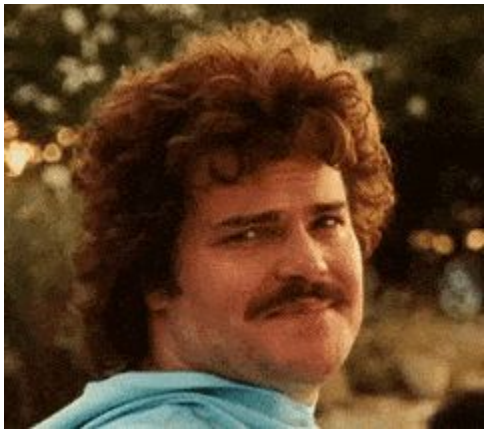
Sekvensdiagram

Use Case-diagram

Oblig 4

Ukesoppgaver

Talk but write



UKESOPPGAVER

Fasit

Spørsmål 1

Spørsmål: Hva er et use case, og hvorfor er det nyttig å lage use cases?

Svar: Kan ses på som en slags **historie**, som beskriver hvordan systemet **oppnår** et mål av verdi for en aktør. Viktig at et use case beskriver en komplett funksjonell enhet som kan testes. Vi skiller mellom use case som tekstlig beskrivelse og use case diagram. Vi har med både **hovedflyt**, **alternativ flyt** og **primær aktør** og eventuelt **sekundær aktør** i et tekstlig beskrivelses use case.

Det er nyttig å lage use case fordi det hjelper oss å se hvilke **steg/sekundær aktører** som må til for at primær aktør skal nå **målet** sitt, og hvor det kan oppstå **alternativ flyt**.

Use case kommuniserer tydelig:

- Hva de forskjellige **målene** med systemet er
- **Hvem** som bruker systemet, og til **hva** systemet skal brukes
- Hvilke andre aktører systemet interagerer med

Spørsmål 2

Spørsmål: Hva skiller aktører fra interessenter?

Svar: Skillet mellom aktør og interessent omhandler **rolle/tilknytning** til bruken og utviklingen av systemet.

- En aktør er en samlebetegnelse for å angi alle grupper av personer som **anvender** systemet, eller **øvrige systemer** som blir **anvendt** av systemet
  - Aktører må være enten:
    - **brukere** av systemet
    - **andre systemer** som brukes av/bruker systemet
- En interessent er en betegnelse for alle personer, grupper eller organer, som **påvirkes** av eller **påvirker** systemets utvikling/bruk. Både direkte og indirekte.
  - Interessenter kan være:
    - **brukere** av systemet
    - inkluderer **alle** som påvirker/påvirkes av systemets kravspesifikasjon og utvikling

Spørsmål 3

Spørsmål: Hva er en aktør i et use case diagram, og hva er forskjellen på en primær og sekundær aktør?

Svar: I et use case diagram er aktøren **strekfigurene**. Disse må ha et **rollenavn**. Primæraktører har et **mål** av verdi i systemet. Sekundæraktører **realiserer** målene til primæraktørene.

Eks:

UML: **Kunde** (primær aktør) setter inn penger

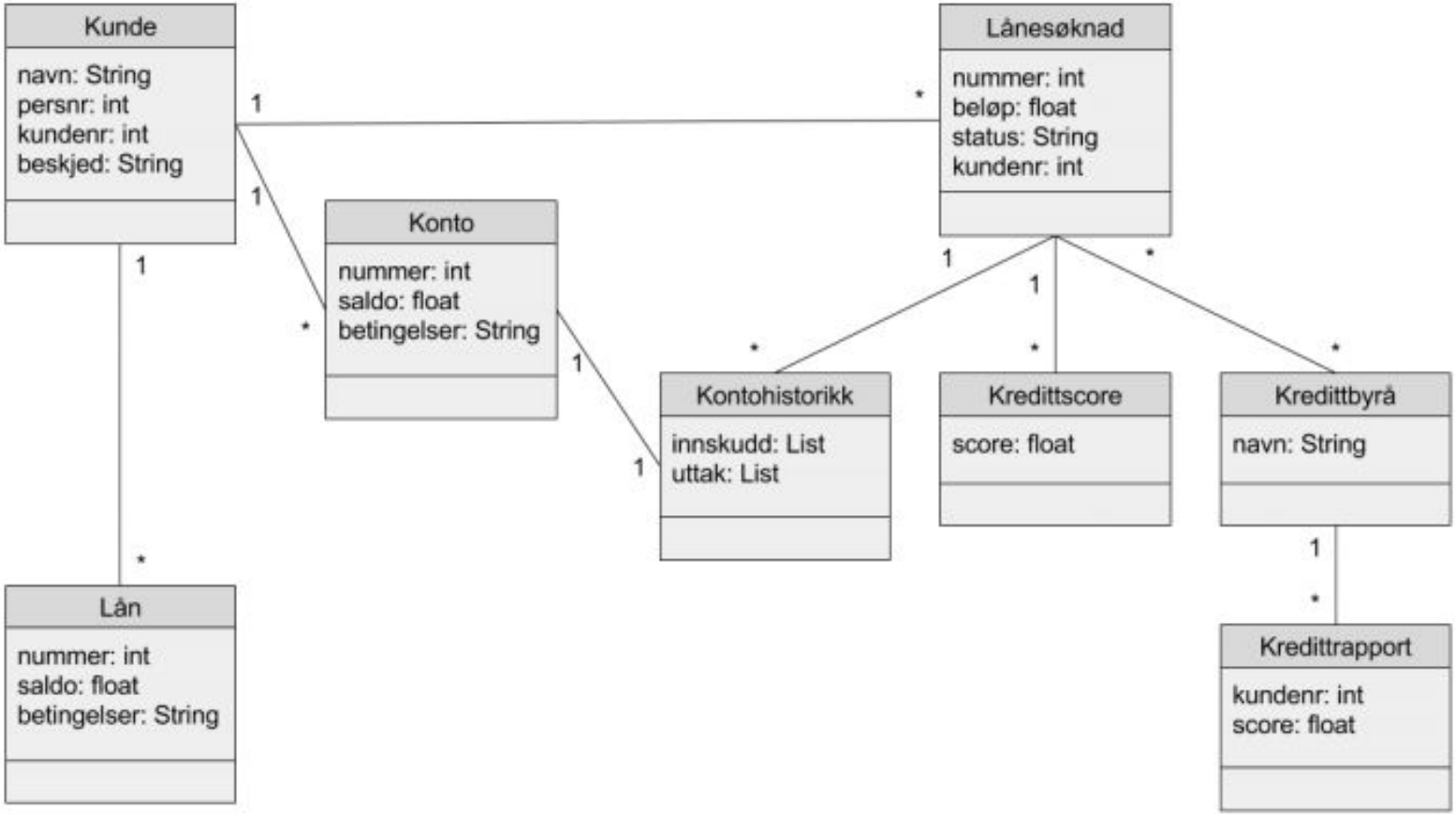
UML: **Bank** (sekundær aktør) sørger for at pengene blir satt inn

Spørsmål 4

Spørsmål: Hva er en domenemodell, og hvorfor er det nyttig å lage en domenemodell for et gitt system?

Svar: En domenemodell er en **representasjon** av de ulike **objektene** (i domenet) et system består av. Modelleres på samme måte som et **klassediagram**, men **uten** metoder.

Det er nyttig for å: **kartlegge** hvilke objekt som man bør ta hensyn til; **kommunisere** at man har forstått domenet; se på **relasjoner** mellom objektene.


Kunde
navn: String
persnr: int
kundenr: int
beskjed: String
Lånesøknad
nummer: int
beløp: float
status: String
kundenr: int
Konto
nummer: int
saldo: float
betingelser: String
Kontohistorikk
innskudd: List
uttak: List
Kredittscore
score: float
Kredittbyrå
navn: String
Lån
nummer: int
saldo: float
betingelser: String
Kredittrapport
kundenr: int
score: float
1
*

Spørsmål 5

Anta følgende beskrivelse av et system som skal håndtere bord og bordbestillinger på en restaurant: Systemet skal støtte **bordreservasjoner** og **bordplassering** i en restaurant. Kunder **kontakter** restauranten for å **bestille** eller **avbestille** bord. En **resepsjonist mottar** samtalene. Bestillinger legges inn for et **bestemt bord** sammen med **antall** personer. For hver bestilling **registreres** en **kontaktperson** med navn og telefonnummer.

Når gjester ankommer, blir de plassert ved sitt bord av **hovmesteren**, og deres bestilling **markeres** med “ankommet”. Hvis gjestene plasseres ved et annet bord enn det som var registrert med bestillingen, så **registreres** bordbyttet i bestillingen. Tidspunktet da et gitt bord må være ledig igjen kan også **registreres**. Kunder kan **endre** bestilling eller avbestille bord på forhånd.

Det er selvfølgelig mulig å spise uten å ha bestilt på forhånd hvis det er ledige bord. Når gjester får bord **uten** å ha bestilt dette, **markeres** det i systemet med **tidspunkt, bord og antall**, men **uten** navn og telefonnummer.

Når nye bestillinger **registreres** i systemet, eller eksisterende bestillinger **endres**, skal skjermbildet **umiddelbart** oppdateres, slik at de ansatte på restauranten alltid har oppdatert informasjon tilgjengelig.

SPØRSMÅL 5A

Spørsmål: Analyser beskrivelsen av et system som skal håndtere bord og bordbestillinger på en restaurant. **Finn aktører for systemet.**

Svar:

- **Resepsjonist:** motta samtaler og håndtere bestillinger
- **Hovmester:** plassere gjester
- Videre kan man *generalisere* disse to til **“Ansatt”**, med samme mål: se oversikt over bestillinger/plasseringer.

Marker de du ser har et mål / hjelper til med å nå mål i teksten

Systemet skal støtte bordreservasjoner og plassering i en restaurant. Kunder kontakter restauranten for å bestille / avbestille bord. En resepsjonist mottar samtalene. Bestillinger legges inn for et bestemt bord sammen med antall personer. For hver bestilling registreres kontaktperson med navn og telefonnummer.

Når gjester ankommer, blir de plassert ved sitt bord av hovmesteren, og bestillingen markeres med "ankommet". Hvis gjestene plasseres ved et annet bord, registreres dette bordbyttet. Tidspunktet et gitt bort må være ledig igjen kan også registreres. Kunder kan endre bestilling / avbestille på forhånd.

Det er mulig å få bord uten å ha bestilt på forhånd, gitt at ledige bord finnes. Når gjester får bord uten å ha bestilt dette, markeres tidspunkt, bord og antall i systemet, men ikke navn og telefonnummer.

Når nye bestillinger registreres i systemet, eller eksisterende bestillinger endres, skal skjermbildet umiddelbart oppdateres, slik at ansatte på restauranten alltid har oppdatert informasjon tilgjengelig.

SPØRSMÅL 5B

Spørsmål: Analyser beskrivelsen av et system som skal håndtere bord og bordbestillinger på en restaurant. **Finn use cases for systemet.**

Analyse:
Hva skal aktørene kunne gjøre? Hvilke resultater vil aktøren oppnå?
Identifiser de viktigste oppgavene (se etter verb)
- Skape / Lagre / Endre / Lese / Slette

Notasjon for use case-funksjoner:
Figur: Oval
Merkelapp: Navn på use case → verbfrase

SPØRSMÅL 5C

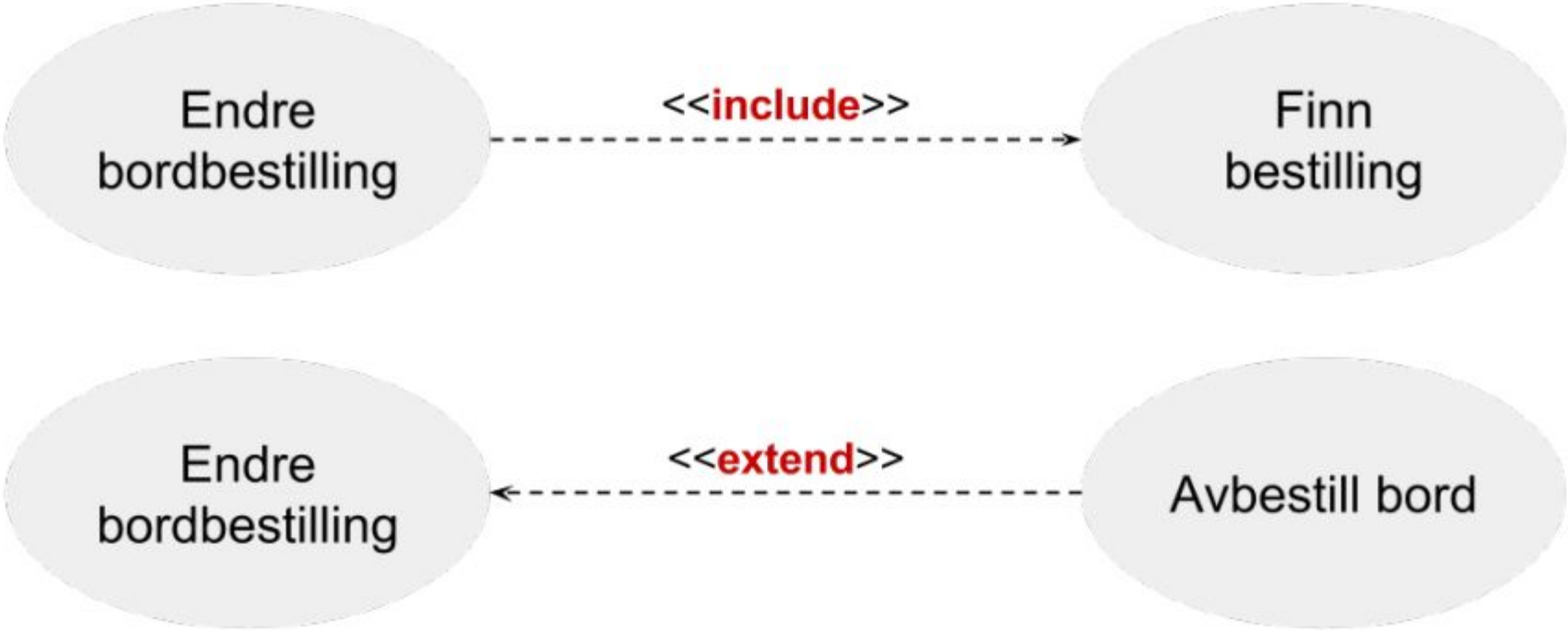
Spørsmål: Analyser beskrivelsen av et system som skal håndtere bord og bordbestillinger på en restaurant. **Lag et use case diagram for systemet.**

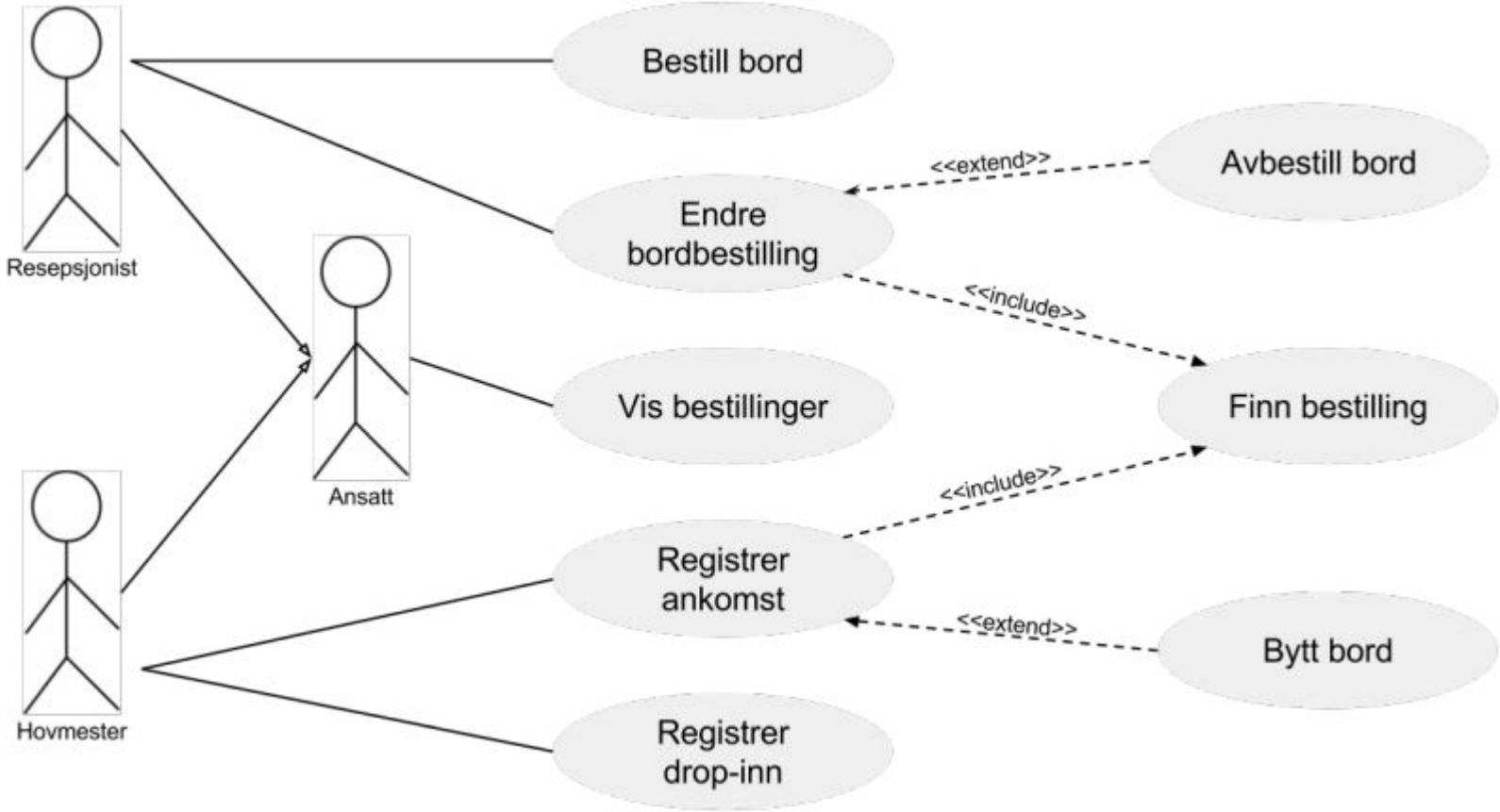
Svar:
Include-relasjonen: Indikerer at et **(sub)** use case inneholder nødvendig funksjonalitet for gjennomførelsen av et annet basiscase.

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


Basiscase
<<include>>
Subcase


Basiscase
<<extend>>
Subcase


Endre bordbestilling
<<include>>
Finn bestilling
Endre bordbestilling
<<extend>>
Avbestill bord


Resepsjonist
Ansatt
Hovmester
Bestill bord
Endre bordbestilling
Vis bestillinger
Registrer ankomst
Registrer drop-inn
Avbestill bord
Finn bestilling
Bytt bord
<<extend>>
<<include>>
<<include>>
<<extend>>

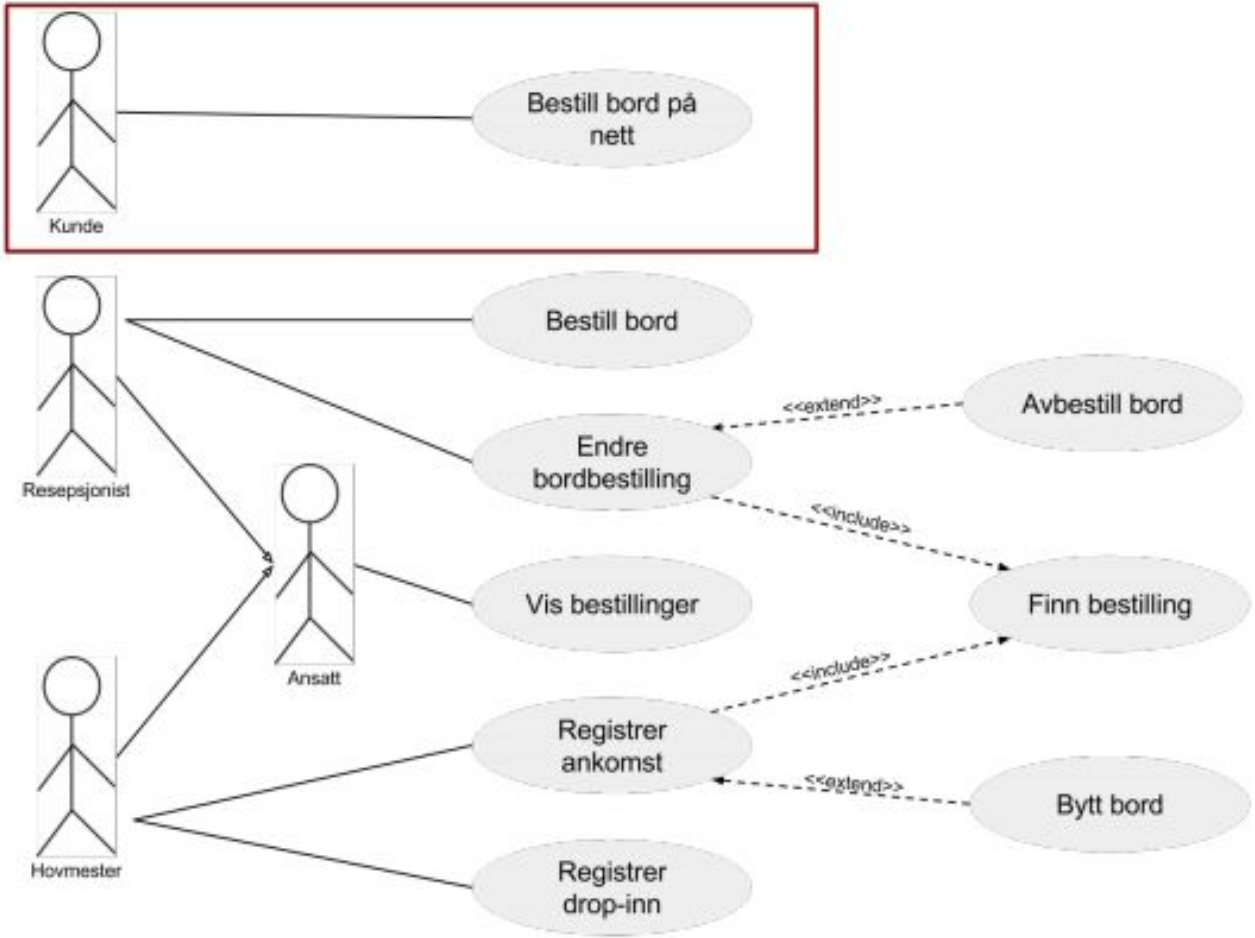
SPØRSMÅL 5D

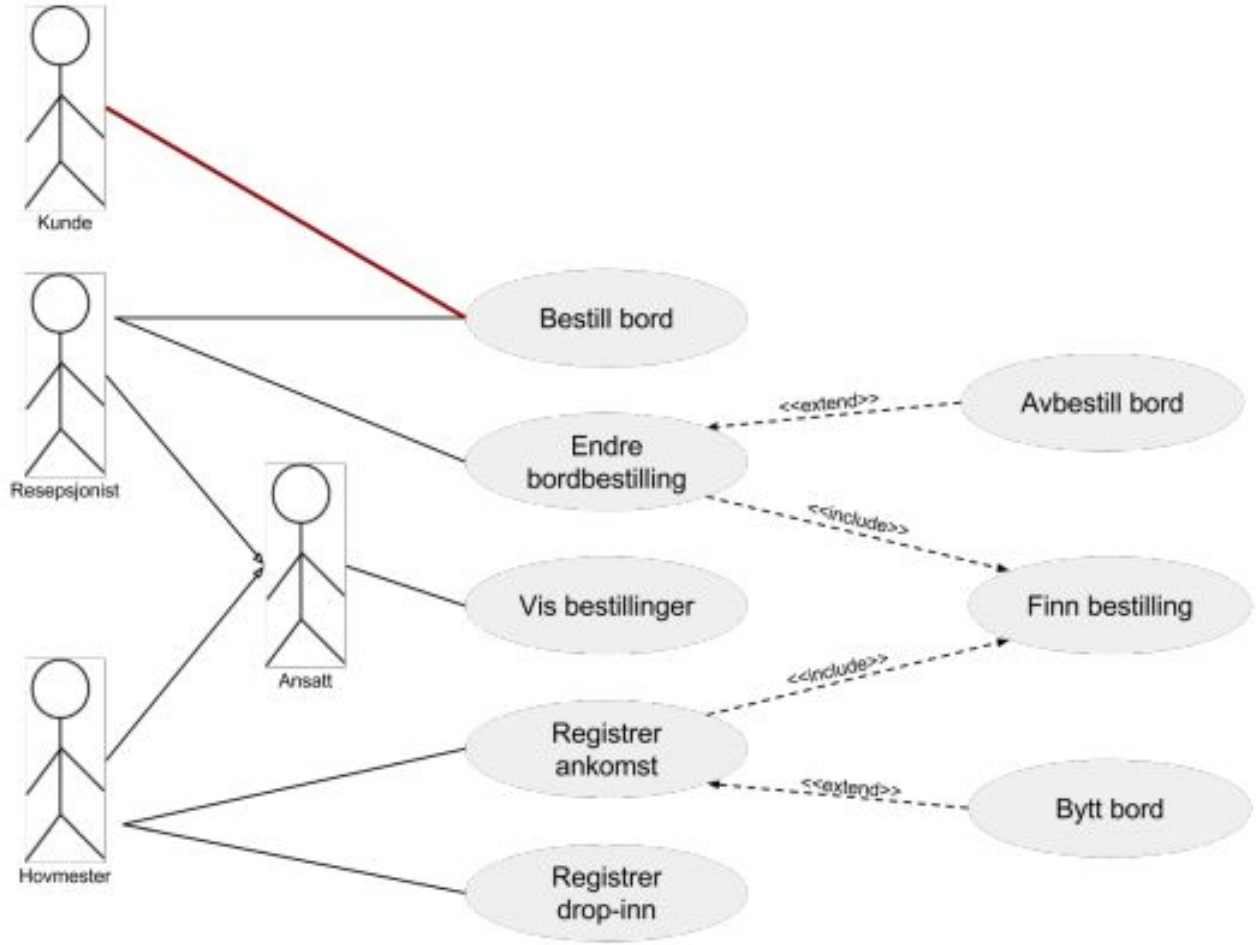
Spørsmål: Anta at en kunde kan bestille bord på nett i tillegg til over telefon. Gjør eventuelle modifikasjoner i løsningene i oppgave a, b og c.

Svar:

Analyse:

A. Ny aktør **Kunde,** med mål: **Bestille bord på nett**
B. Nytt use case?
    a. Skal man opprette et nytt use case “Bestill bord på nett?”
    b. Skal man beholde “Bestill bord”?
C. Oppdater use case-diagrammet


Kunde
Bestill bord på nett
Resepsjonist
Ansatt
Hovmester
Bestill bord
Endre bordbestilling
Vis bestillinger
Registrer ankomst
Registrer drop-inn
<<extend>>
Avbestill bord
<<include>>
Finn bestilling
<<include>>
<<extend>>
Bytt bord


Kunde
Resepsjonist
Ansatt
Hovmester
Bestill bord
Endre bordbestilling
Avbestill bord
<<extend>>
<<include>>
Vis bestillinger
Finn bestilling
<<include>>
Registrer ankomst
<<extend>>
Bytt bord
Registrer drop-inn

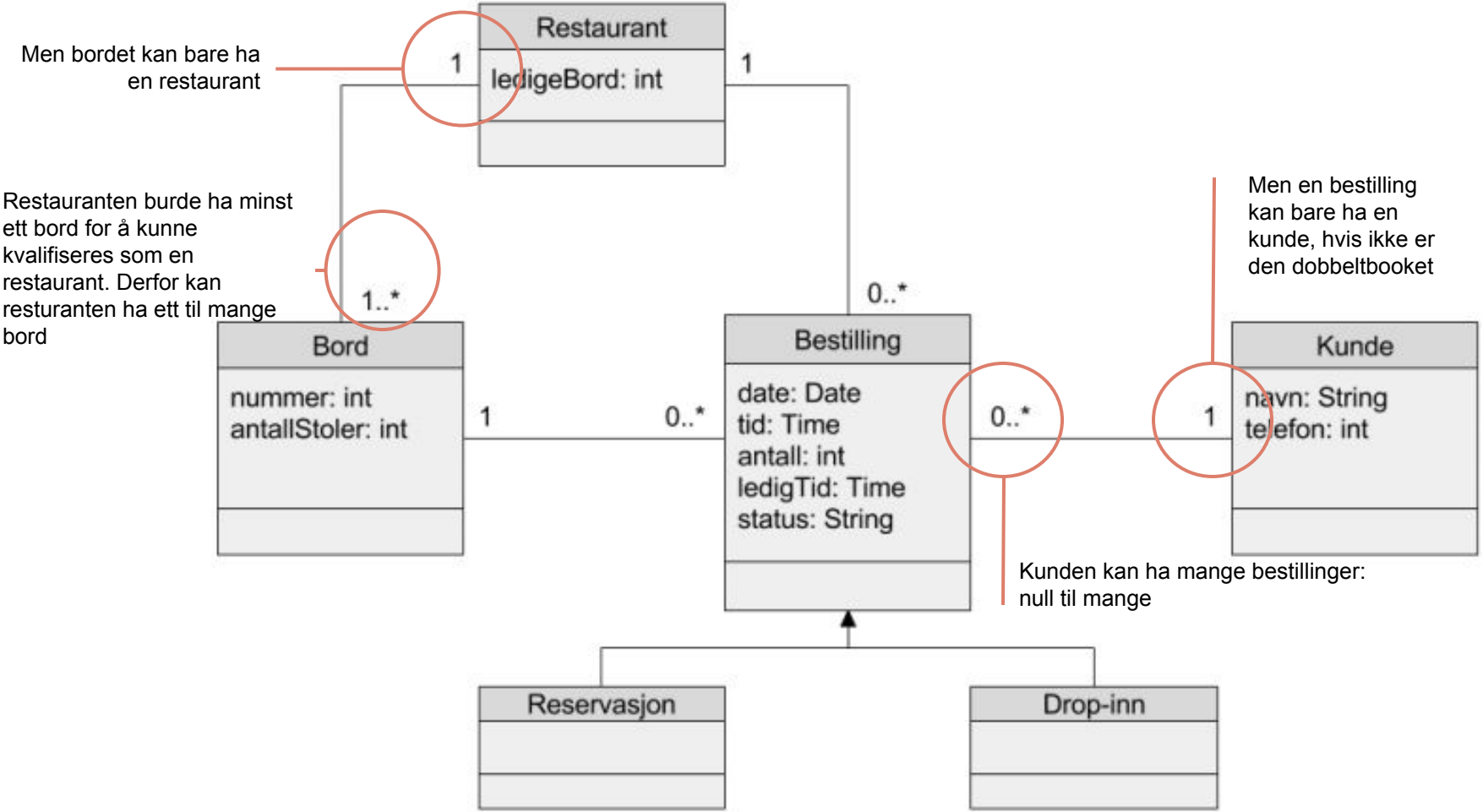
Spørsmål 5E

Spørsmål: Lag en enkel domenemodell for systemet.

Svar: Domenemodell: klassediagram **uten** metoder
Hvilke klasser er aktuelle å modellere?

For hver klasse:

- Hvilke attributter (**variabler**) er naturlig å inkludere?
- Hvilket **forhold** er det mellom klassene? Én-til-én, Én-til-mange, Mange-til-én, Mange-til-mange


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..*
0..*
Men en bestilling kan bare ha en kunde, hvis ikke er den dobbeltbooket
Bord
nummer: int
antallStoler: int
1
0..*
Bestilling
date: Date
tid: Time
antall: int
ledigTid: Time
status: String
0..*
1
Kunde
navn: String
telefon: int
Kunden kan ha mange bestillinger: null til mange
Reservasjon
Drop-inn

Spørsmål 6A

Spørsmål: Hva er en tekstlig beskrivelse av et use case?

Svar: En tekstlig beskrivelse av en use case tar for seg **interaksjonen mellom systemet og bruker.** Det er en **nummerert liste** som beskriver **interaksjonen** steg for steg. En tekstlig beskrivelse skal inneholder følgende:

Navn på use case → Hvilken funksjonalitet dreier det seg om?
Aktør → hvem/hva er det som interagerer med systemet?
Prebetingelse(r) → Hva skal til for å starte use caset?
Postbetingelse(r) → Hva skal til for å avslutte use caset?
Hovedflyt → Hvilke steg inngår i gjennomførelsen av use caset?
Alternativ flyt → Hvilke steg inntreffer ved avvik i hovedflyten?

Obs: viktig at alternativ flyt også **termineres**, enten ved å gå tilbake til hovedflyten eller ved å avslutte.

Spørsmål 6B

Spørsmål: Lag én tekstlig beskrivelse av ett av use casene du fant i 5 b. Ha med eventuelle pre- og postbetingelser og få med minst én alternativ flyt.

Svar del 1:
Navn: Bestill bord

Aktører: Resepsjonist

Prebetingelse: ingen

Postbetingelser (mål): Bord er reservert på kunde

Spørsmål 6B

Spørsmål: Lag én tekstlig beskrivelse av ett av use casene du fant i 5 b. Ha med eventuelle pre- og postbetingelser og få med minst én alternativ flyt.

Svar del 2:
Hovedflyt:

1. Kundebehandler ber systemet om å finne et ledig bord på en gitt tid og dato
2. Systemet finner ledige bord
3. Kundebehandler registrerer kundens navn, adresse og telefonnummer
4. Systemet lagrer kundens navn, adresse, og telefonnummer
5. Systemet ber om bekreftelse på reservasjon om bord
6. Kundebehandler bekrefter reservasjon av bord
7. Systemet registrerer reservering av bord på kunde

Alternativ flyt:

2.1. Systemet finner ingen ledige bord
2.2. Returnerer til hovedflyt, steg 1.

Spørsmål 6B

Spørsmål: Lag én tekstlig beskrivelse av ett av use casene du fant i 5 b. Ha med eventuelle pre- og postbetingelser og få med minst én alternativ flyt.

Svar del 3:
Modellering av alternativ flyt – Identifiser avvik i hovedflyten; der hovedflyten har alternativ.
Hva skjer i slike tilfeller? Hvordan responderer systemet?

Alternativ til alternativ flyt i oppgaven:
Bruker ber systemet om å finne et ledig bord til en gitt tid og dato:

Systemet returnerer ledig bord → INGEN LEDIGE BORD

Bruker registrerer kundens navn og telefonnummer:

Systemet lagrer kundens navn og telefonnummer → UGYLDIG NAVN/TELEFONNUMMER

Systemet ber om bekreftelse på reservasjon av bord

Bruker bekrefter reservasjon av bord → BRUKER AVKREFTER RESERVASJON

Spørsmål 7

Spørsmål: Hva er et sekvensdiagram?

Svar: Et sekvensdiagram er et **interaksjonsdiagram**
Modellen viser en interaksjonssekvens mellom **objektene** som finner sted under et bestemt use case.

Sekvensdiagrammer viser:

- **Hvilke** objekter som inngår i et bruksmønster
- **Interaksjonen** mellom objektene/deres rekkefølge
- **Data/informasjon** som sendes mellom objektene

Spørsmål 8

Spørsmål: Hvorfor er det nyttig å benytte sekvensdiagram?

Svar: Det er viktig å kunne vise **hva** som skjer/burde skje ved **kjøretid** av systemet.
Altså: Vise den **dynamiske** oppførselen til et program. Sekvensdiagram gir en **oversikt** over nødvendige **klasser og objekter** for å gjennomføre et **bruksmønster**. Det kan gjøre det **enklere** å **implementere** et system. Det er et svært **kodenært** diagram, som gjør det mulig å generere kode automatisk fra et sekvensdiagram.

Det viser også hvordan objektene **kommuniserer**, og gir **oversikt** over data som **sendes mellom** objektene, og **rekkefølgen** på dataoverføringen.

Spørsmål 9

Spørsmål: Et bilutleiefirma ønsker et informasjonssystem som kundebehandlerne kan benytte for utleie av biler. Under er det et forslag til en tekstlig beskrivelse av bruksmønsteret Reserver bil. Lag et sekvensdiagram for hovedflyten og én av de alternative flytene i den tekstlige beskrivelse.

- Tekstlig beskrivelse av use case 'Reserver bil'
  - **Navn:** Reserver bil
  - **Aktør:** Kundebehandler
  - **Prebetingelse:** Ingen
  - **Postbetingelse:** Leiekontrakt for spesifisert bil og kunde med gitte utleiedatoer er opprettet
- **Hovedflyt:**
  - 1. Kundebehandler velger tidsintervall (hentedato og returdato)
  - 2. Systemet returnerer en liste over tilgjengelige biler innenfor de spesifiserte datoene
  - 3. Kundebehandler velger én av bilene.
  - 4. Systemet ber om kundenr og finner kunden i systemet
  - 5. Systemet bekrefter at bilen er reservert for den gitte perioden
- **Alternativ flyt 2.1:** Det finnes ingen tilgjengelige biler i valgt tidsintervall.
  - 2.2. Systemet opplyser om at det ikke er tilgjengelige biler innenfor oppgitt tidsintervall.
  - 2.3. Kundebehandler oppgir et nytt tidsintervall (steg 1) eller avslutter bruksmønsteret.
- **Alternativ flyt 4.1:** Kunden finnes ikke.
  - 4.2. Systemet oppretter ny kunde og returnerer til steg 3.

Spørsmål 9

Svar del 1: Analyse av den tekstlige beskrivelsen:

1. Identifiser de aktuelle objektene/aktørene:
   - Hvilket system er det snakk om? -- Reservasjonssystem for utleie av biler
   - Har vi eventuelle undersystemer? -- Bilregister/Kunderegister (avhengig av hvordan systemet er implementert)
   - Har vi eventuelle objekter? -- Kunde (objekter for de ulike kundene)/Kontrakt (objekt for utleiekontrakt)
   - Hvem skal interagere med systemet? → Kundebehandler
2. Oppsett (rekkefølgen gis av flyten)
   - Aktøren (Kundebehandler) plasseres til venstre i sekvensdiagrammet
   - Aktøren interagerer med Reservasjonssystemet
     1. Kundebehandler velger tidsintervall (hentedato og returdato)
        Reservasjonssystemet interagerer…
        Først med Bilregisteret
     2. Returnerer liste over tilgjengelige biler
        Deretter med Kunderegisteret
     4. Finner kunden i systemet
        Kundeobjekt må være på plass før avtalen tegnes
        Kontrakten opprettes til slutt
     5. Systemet bekrefter at bilen er reservert for den gitte perioden

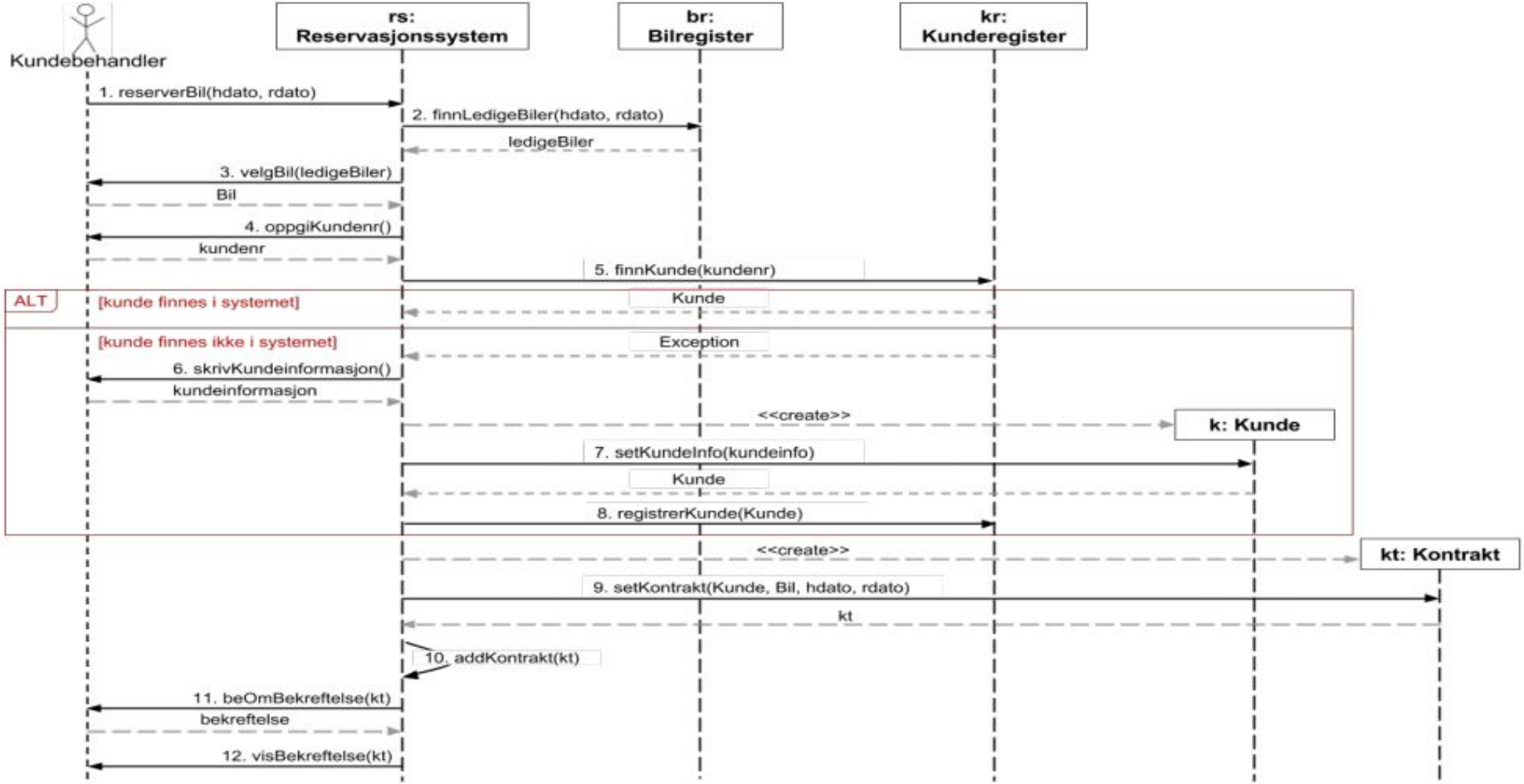
Foreløpig oppsett → Har nå et skjelett for hvordan sekvensdiagrammet skal se ut

Kan starte å modellere den tekstlige beskrivelsen

Spørsmål 9

Svar del 2:

3. Modeller hendelsesforløpet ved å følge den tekstlige beskrivelsen:
    - Rekkefølgen fra hoved- og alternativ flyt bestemmer rekkefølgen i diagrammet
    - Hvert steg i den tekstlige beskrivelsen er tilnærmet lik en pil i diagrammet
    - Data som sendes mellom objektene reflekteres i diagrammet
    - Metodekall (med parametere) → Heltrukket pil
    - Returverdier → Stiplet pil
    - Create (for å opprette objekter) → Stiplet pil


Kundebehandler
rs: Reservasjonssystem
br: Bilregister
kr: Kunderegister
1. reserverBil(hdato, rdato)
2. finnLedigeBiler(hdato, rdato)
ledigeBiler
3. velgBil(ledigeBiler)
Bil
4. oppgiKundenr()
kundenr
5. finnKunde(kundenr)
ALT
[kunde finnes i systemet]
Kunde
[kunde finnes ikke i systemet]
Exception
6. skrivKundeinformasjon()
kundeinformasjon
<<create>>
k: Kunde
7. setKundeInfo(kundeinfo)
Kunde
8. registrerKunde(Kunde)
<<create>>
kt: Kontrakt
9. setKontrakt(Kunde, Bil, hdato, rdato)
kt
10. addKontrakt(kt)
11. beOmBekreftelse(kt)
bekreftelse
12. visBekreftelse(kt)

emmatv@uio.no