10. Nevn et eksempel på en tre-lags arkitektur.

- MVC (Model View Controller)

11. Hvorfor benyttes designmønstre (design patterns)?

- Mye av vårt daglige virke består i å lete etter strukturer (mønstre) i omgivelsene
- Mønstre er vanlige løsninger på vanlige problemer
- Mange systemer har grunnleggende fellestrekk

12. Hva er Kanban?

- Smidig prosessmodell
  - Bygd på Lean og ”just-in-time” prinsippet
  - Minst mulig WIP (Work in progress)

2 a – Use Case- tekstlig beskrivelse

- **Navn**: Beregn Mengde
- **Aktør**: Person (eller forbruker el.l.)
- **Pre-betingelse:** Ingen
- **Post-betingelse** (hovedflyt): Person har fått oversikt over sitt totalforbruk

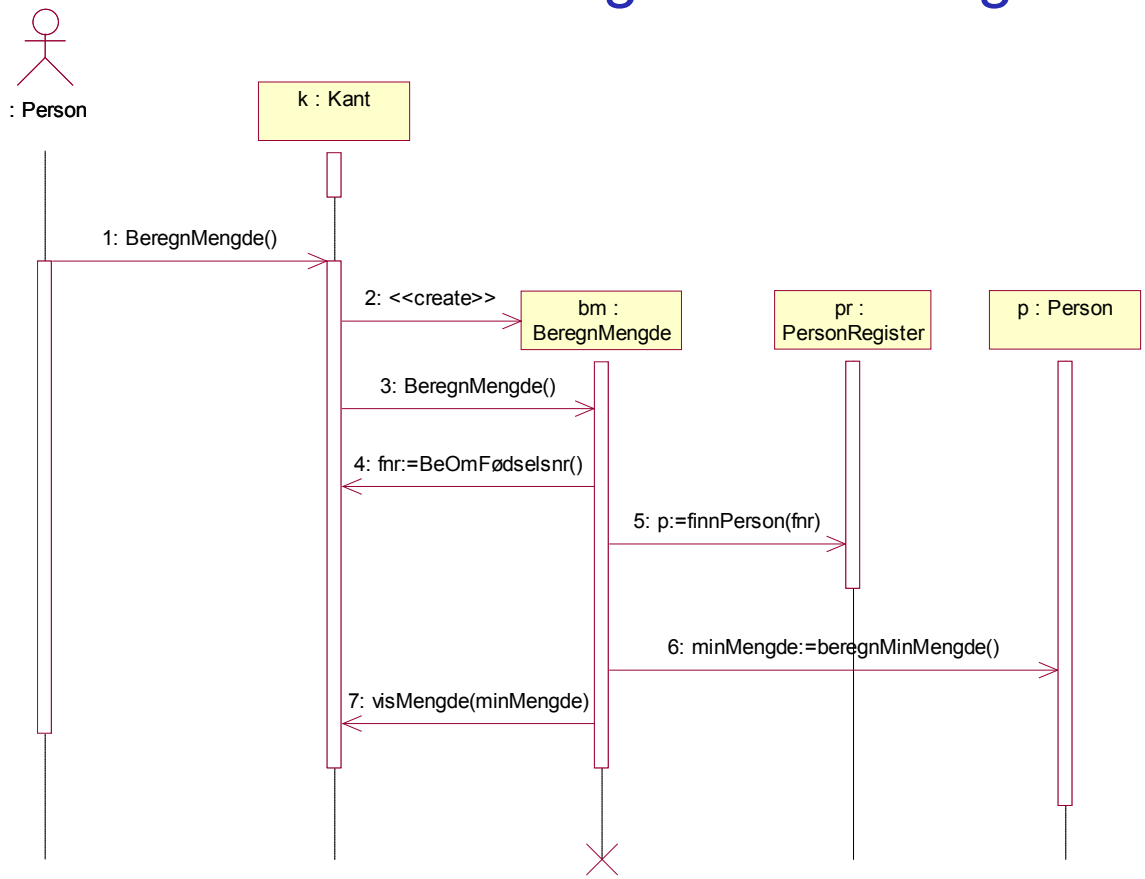
- Normal Hendelsesflyt:
  - 1. Personen oppgir sitt fødselsnummer
  - 2. Systemet finner personen i systemet
  - 3. Systemet beregner den totale klimagassmengden som personen gjennom sine kjøp har bidratt til
  - 4. Systemet viser klimagassmengden til personen
- Variasjoner:
  - 2a. Ugyldig fødselsnummer:
    - 1. Systemet gir feilmelding og avslutter
  - 2b. Personen finnes ikke i systemet:
    - 1. Systemet gir feilmelding og avslutter

2 a – Use Case- tekstlig beskrivelse - Kommentar

Her finnes det mange varianter som er greie løsninger. Man kan godt si at hvis man oppgir et ugyldig fødselsnummer så vil man heller ikke finne personen, så hvis man ikke har med noe i likhet med 2a så er det også ok. Bruksmønsterer og sekvensdiagrammet i 2 b bør være ”rimelig” konsistente. Det gis trekk om bruksmønsteret uttrykker noe helt annet enn sekvensdiagrammet.

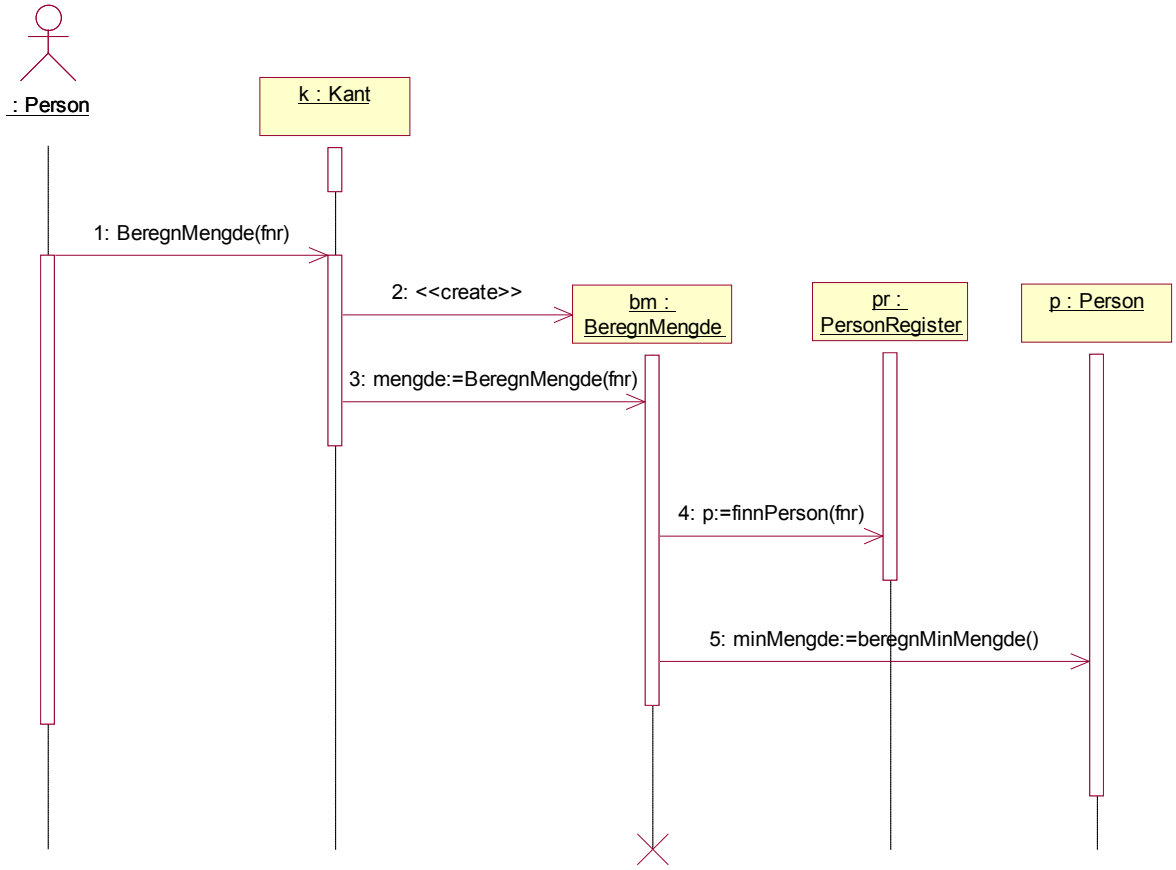
Pre-betingelse: I gjennomgangen av oppgaven ble det diskutert om ”Personen ønsker å se sin mengde” kunne være en pre-betingelse. Det er nok mest riktig å se det som en ”trigger” og ikke en pre-betingelse, slik at jeg velger her at det ikke er noen pre-betingelse. En pre-betingelse uttrykker mer noe som må være oppfylt for å sette i gang bruksmønsteret.

2 b – Sekvensdiagram – ”Beregn mengde”



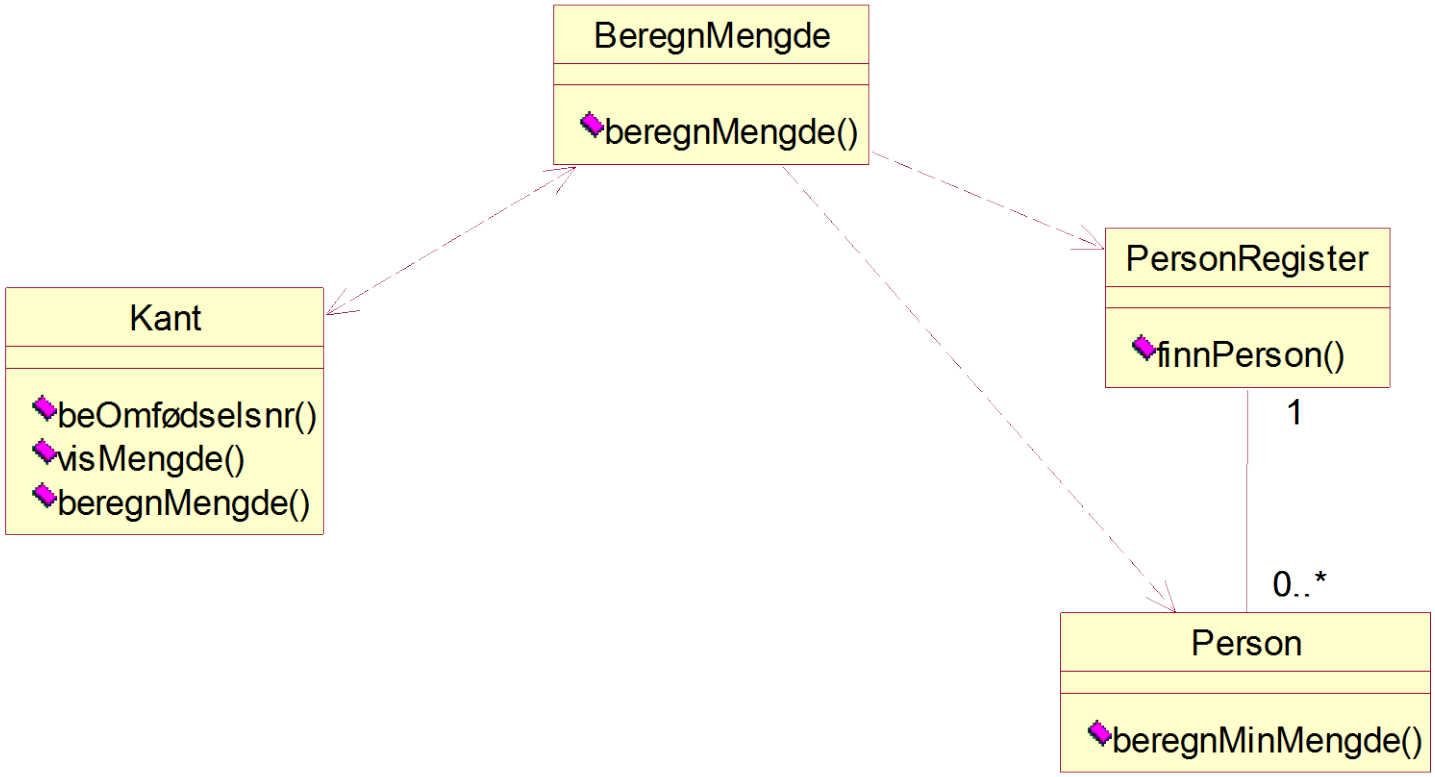
Kommentar til diskusjonen på gjennomgangen om <<Create>> og kallet på BeregnMengde() . I programmering kalles konstruktøren automatisk. Men vi har med <<create>> for å modellere at vi oppretter en ny sesjon. Det holder ikke å bare lage objektet, vi må også "starte" det opp, og det gjør vi med kallet på BergenMengde(). <<create>> er bare en måte å modellere på, det initierer ingenting.

2 b – Sekvensdiagram – alternativ versjon



Kommentar: Dette er en enklere løsning der fnr er med som parameter og jeg antar at kontrollobjektet returnerer mengden i metoden ”BeregnMengde(fnr)". Det er strengt tatt ikke nødvendig å kalle ”k.vismengde()”, da vi kan anta at dette skjer bak ”kulissene”

2 c – Klassediagram – basert på første versjon av sekvensdiagrammet



Kommentar: Personregisteret kan ha 0 eller flere personer. Det er ”løse relasjoner” mellom Kant og BeregnMengde (piler i begge retninger), mens pilene går fra BeregnMengde til Personregister og Person (kan leses fra sekvensdiagrammet).

Oppgave 4 – Fossefall versus smidig

- **Tunge versus lette prosesser**
  - En tung prosess inkluderer mange aktiviteter, og ofte roller og krever formelle, detaljerte og konsistente prosjektdokumenter.
  - Tunge prosesser er ofte “for-tunge”, dvs. vektlegger aktiviteter som vanligvis gjøres tidlig i prosessen (planlegging, analyse & design)
  - Lette prosesser fokuserer mer på fundamentale prosesser (“f.eks. kontinuerlig testing”), har færre formelle dokumenter og er ofte mer iterative

Oppgave 4 – Fossefall versus smidig

- Egenskaper ved Fossefall
  - Vanskelig å tilpasse endringer i krav underveis i prosjektet
  - Egner seg best når kravene er godt forståtte og lite sannsynlig med mye endringer underveis
  - Men få systemer har stabile krav …
  - Fossefallsmodellen brukes mest i store prosjekter som gjerne utvikles på ulike steder. Den plandrevne utviklingen gjør det enklere å koordinere arbeidet
  - Brukes også i små, godt forståtte prosjekter

Oppgave 4 – Fossefall versus smidig

Egenskaper ved Smidig

- Jobber i team med en flat struktur
- Teamleder er fasilitator, ikke tradisjonell prosjektleder
- Utviklingen foregår gjennom iterasjoner, i scrum kalt **sprinter**, av varighet 2-4 uker.
- Sprintplanlegging
- Backlog og produkteier (PO)
- User stories
- Estimering : Planning Poker
- ”**stand-up**”-møter (ofte daglige møter)
- Kanban – ”Just in time prinsippet”, ikke tidsboks som i Scrum