

# Domenestruktur og medfølgende egenskaper - et epos

La oss se på hvordan vi modellerer en adresse.



Hensikten med denne analysen er ikke å si at alternativ a er bedre enn alternativ b, men å gi en forståelse på at det finnes flere alternativer og at man kan velge det alternativet som passer den oppgaven man skal løse.

## Domenestruktur:

- 3.nf RDBMS / Object strukture

```
public class Address {  
    private String streetname = null;  
    private short streetnumber = null;  
    private int[4] postcode = null;  
    private String city = null;  
    ....  
}
```

```
@Entity  
@Table(name = "ADDRESS")  
public class Address {  
    ...  
    @Embedded  
    public String getStreetName() {  
        return this.streetname;  
    }  
    @Embedded  
    public short getStreetNnumber()  
    {  
        return this.streetnumber;  
    }  
    @Embedded  
    public String getPostCode() {  
        return int[4].postcode;  
    }  
    @Embedded  
    public String getCity() {  
        return this.city;  
    }  
}
```

## Beskrivelse:

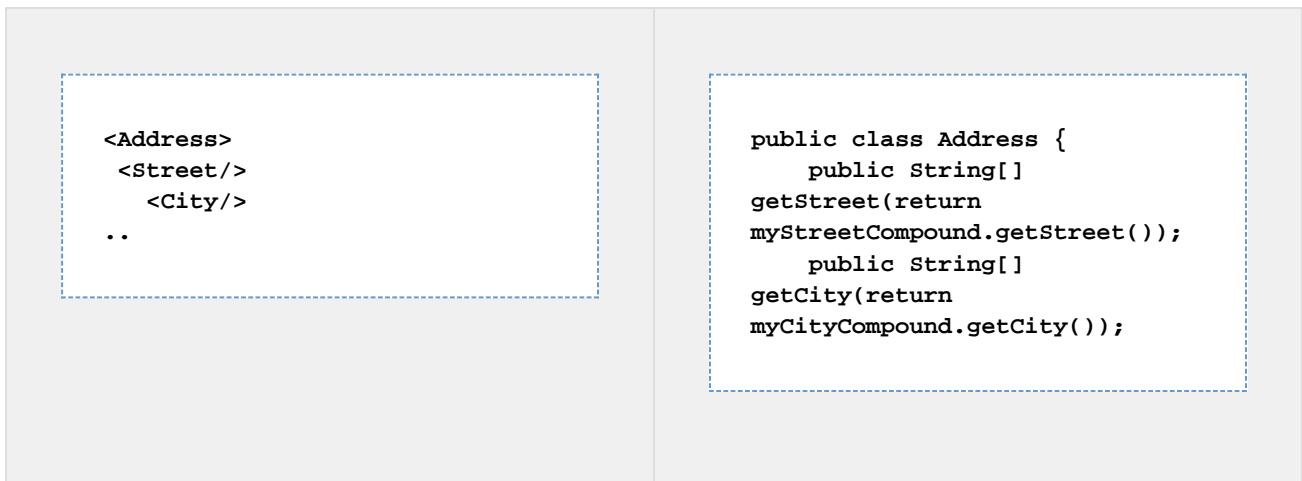
Vi lager adresse som 3 normalform databasestruktur med tabeller for gatenavn, gatenummer, postnummer, poststed og muligens gårds og bruksnummer samt tilhørende oppslag mot "standard kodeverk" som f.eks gyldige poststed.

Egenskaper      Score (0-10)

- Nærhet til det virkelige liv: 8 (Meget god mapping til det opplevde daglige liv, men det er noen underliggende regler/unntak som de fleste mennesker ikke kjenner)
  - Konsistens: 9 (Kan ikke bli 10, siden domenet i seg selv ikke er 100% konsist)
  - Endingsevne: 3 (støtter godt endringer i kodeverk, men knekker helt sammen hvis man begynner med f.eks utenlandske adresser..)
  - Ytelse: 3 (rene oppslag medfører flere joins i database, endringer medfører flere skriveoperasjoner håndert av en overhengende transaksjon som reduserer ytelse og muligheter for paralleisme)
  - Skalering: 2 (sterk kobling mellom delene i strukturen og sterkt kobling til persistens og konsistens reduserer mulighetene for både horisontal og vertikal skalering)
  - Brukseffektivitet: 7 (Normale brukstilfeller mot domeneobjekter basert på denne strukturen er enkelt og naturlig i de fleste programmeringsspråk (OO språk).
- 

### Domenestruktur:

- Semistrukturert aggregat / Compound struktur



### Beskrivelse:

Vi lager adresse et semi-strukturert aggregat med elementer for gate og sted.

### Egenskaper Score (0-10)

- Nærhet til det virkelige liv: 6 (Rimelig god mapping til det opplevde daglige liv, men det er noen underliggende regler/unntak som de fleste mennesker ikke kjenner)
- Konsistens: 4 (Kan inneholde ugyldige kombinasjoner)
- Endingsevne: 8 (Vil enkelt kunne endres og utvides til ikke-trivielle adresser som gyldige unntak i Norge eller utenlandstke adresser)
- Ytelse: 6 (Raske oppslag, endringer vil kunne medføre flere skriveoperasjoner håndert av en overhengende "transaksjon" men uten løsing)
- Skalering: 7 (løs kobling mellom delene i strukturen og løs kobling til persistens og konsistens gir store muligheter for både horisontal og vertikal skalering)
- Brukseffektivitet: 5 (Normale brukstilfeller mot domeneobjekter basert på denne strukturen er greit nok i de

fleste programmeringsspråk (OO og ikke-OO språk).