

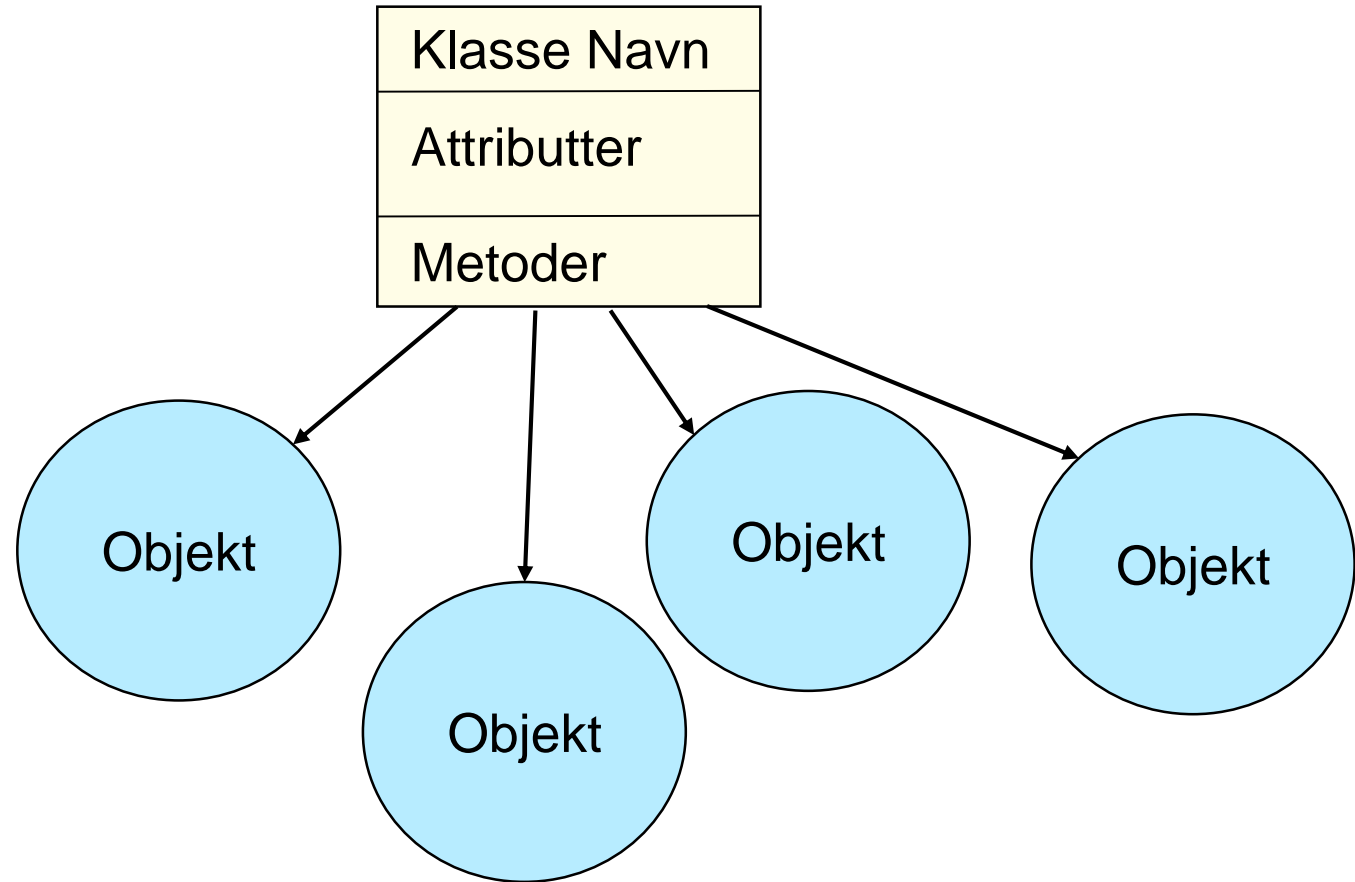
# Interfaces, Abstract Classes Generics

Peter Levinsky, IT Roskilde

13.02.2023

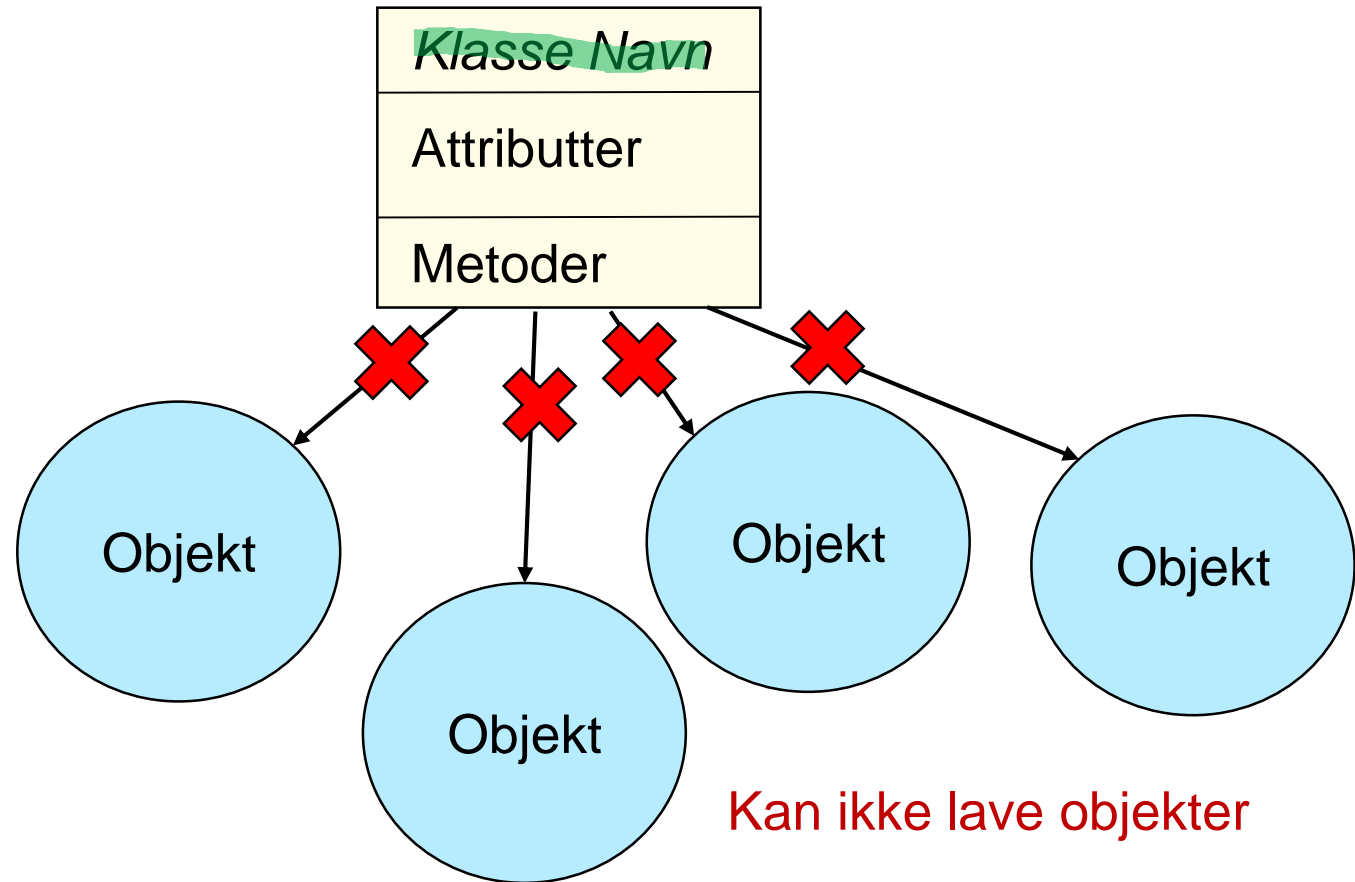
# Normal klasse

- Attributter
- Properties
- Metoder
- Konstanter
- Konstruktører

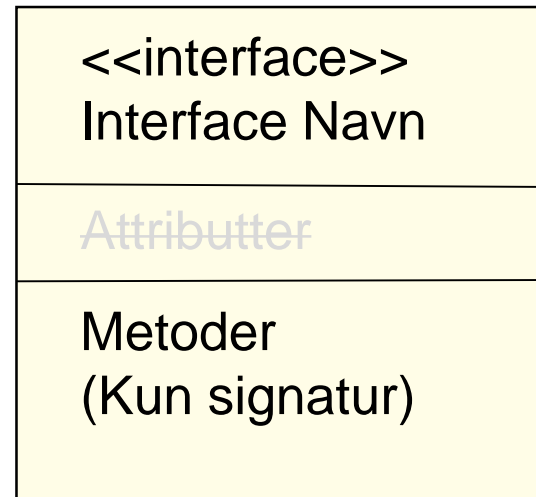


# Abstrakt Klasse

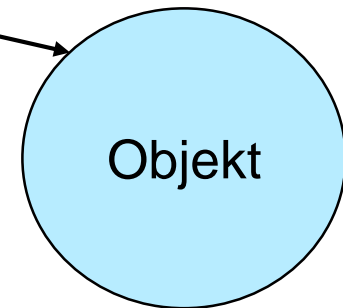
- Attributter
- Properties
- Metoder (evt. abstrakte)
- Konstanter
- Konstruktører



# Interface

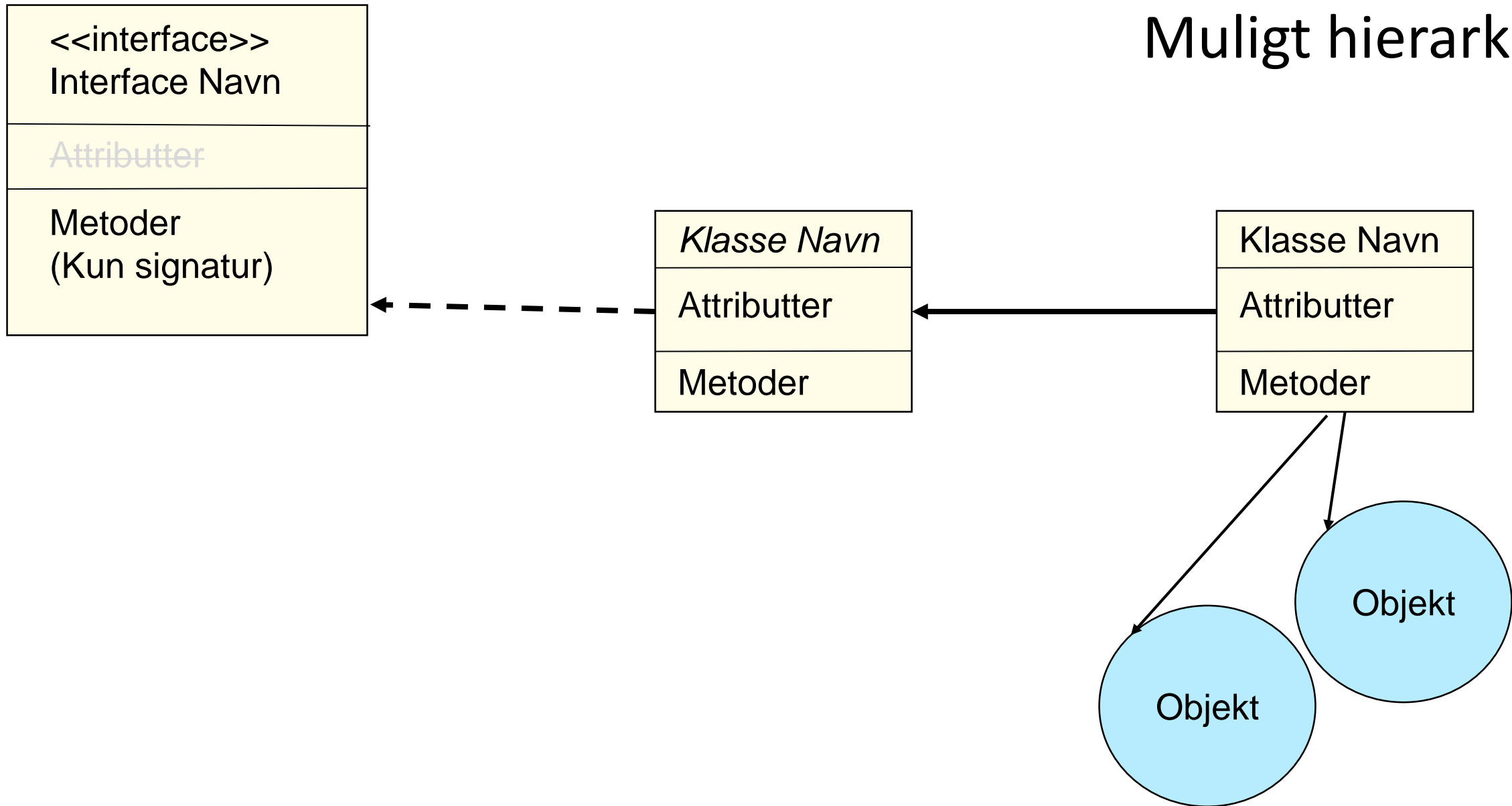


- ~~Attributter~~
- Properties (minus kode/krop)
- Metoder (minus kode/krop)
- Konstanter
- ~~Konstruktører~~



Er på mange måder,  
det I definerer i et Design Class Diagram  
Nemlig: Metodenavn,returværdi og parameter til metoden

# Muligt hierarki



# OPGAVE

# Generics

- DRY – Don't Repeat Yourself
- Flere klasser til gem og hent til json fil
- Standard bibliotek Lister  
Det kan være lister af hvad som helst – men ønsker fx liste af Biler hhv. cykler

```
List<Bil> biler = new List<Bil>(); // bliver det en liste af Biler
```

```
List<Cykel> biler = new List<Cykel>(); // bliver det en liste af Cykler
```

# Generics - hvordan

- Lave en klasse, der er generisk (generic)
- Benytte et metategn for en type – som oftest T
- T kan så erstattes med en specifik type
- Eksempel:

```
class Generic<T>{  
    private T element; -- et element af typen T (ikke bestemt)  
  
    public void XX(T elem) { -- parameter er af typen T  
        ...  
    }  
}
```



# Generics – god praksis

- Lav dine klasser som du ville almindelig
- Hvis det viser sig du skal lave en klasse mangan til så skal du refaktorere klassen til at blive generisk

# OPGAVE