OBJEKTORIENTERING:

KLASSER & OBJEKTER

KLASSER

Man kan lægge instrukser ind i metoder:
- Og man kan lægge metoder ind i klasser!

En klasse er en slags skabelon til et objekt!
- Et blueprint, en opskrift, en grundplan…

Man instantierer et objekt af en klasse:
- Fra Klassen Monster kan man lave et eller flere objekter

```
Monster monster1 = new Monster();
Monster monster2 = new Monster();
```

To selvstændige objekter i programmet
er nu lavet ud fra samme Monster-klasse!

Objektorienteret programmering

- Man opererer med klasser, objekter og metoder
- Data og funktioner samles i én enhed (data- og metodeindkapsling)

Analyse, design og programmering dokumenteres med klasser og objekter

Klasse

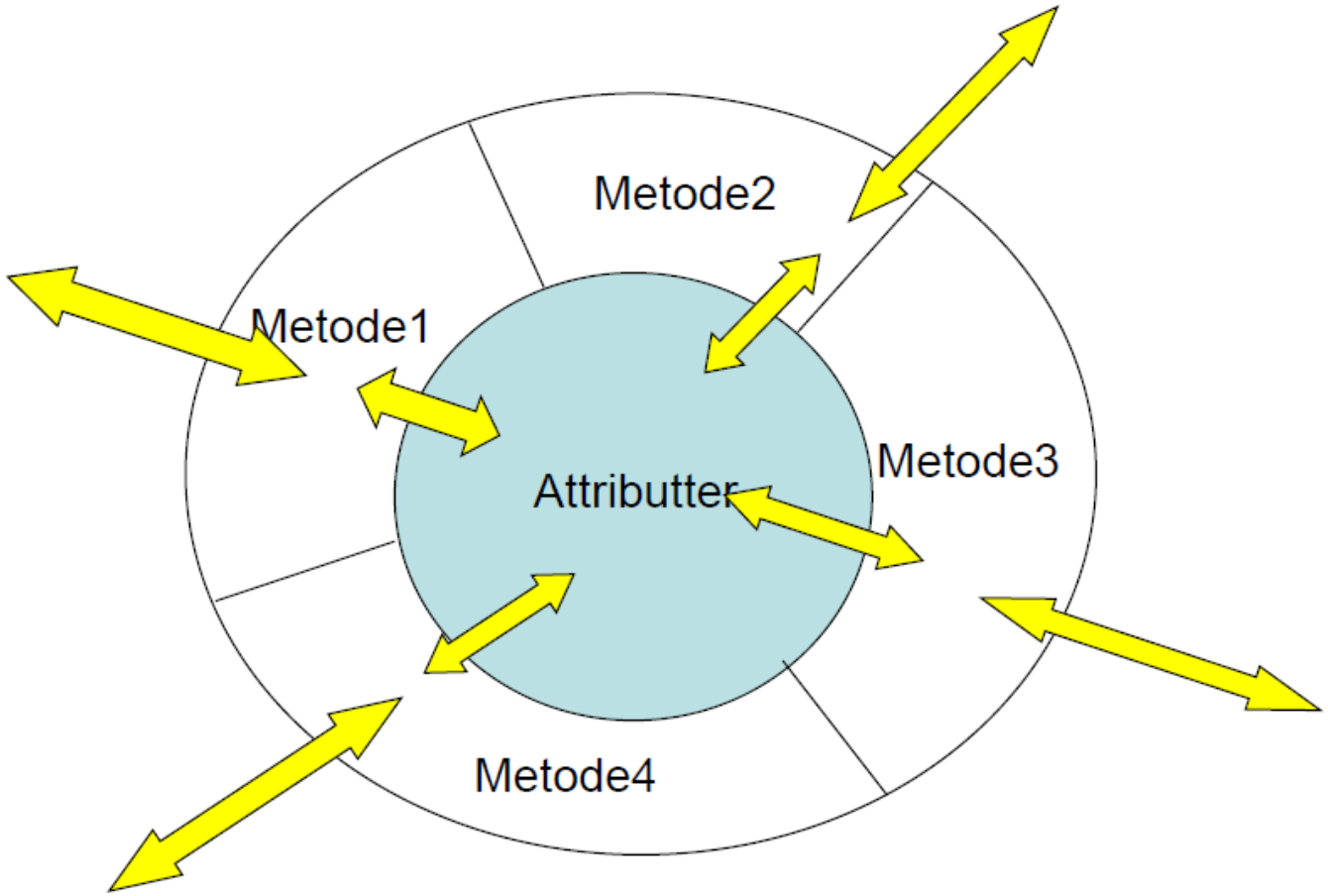
- En type, der beskriver/definerer fælles kendetegn for en samling af flere ensartede objekter (klasse = objekttype)

Objekt

- En konkret forekomst (en instans) af en klasse

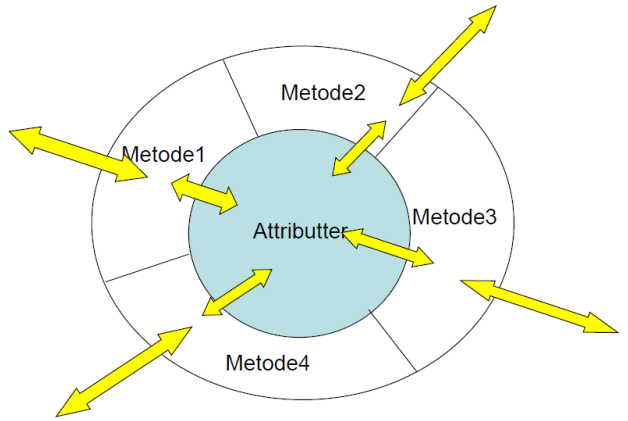
KLASSER: INDKAPSLING, ET PRINCIP

At *indkapsle* data i et objekt – tilgås via metoder

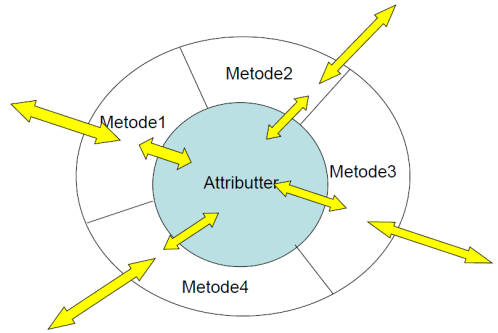


KLASSER: PRINCIPPER

- *Hvad kan man bruge klasser til?*
- *Hvad menes der med indkapsling & nedarving?*
- *Må en klasse have samme navn som en anden klasse?*



KLASSER: PRINCIPPER

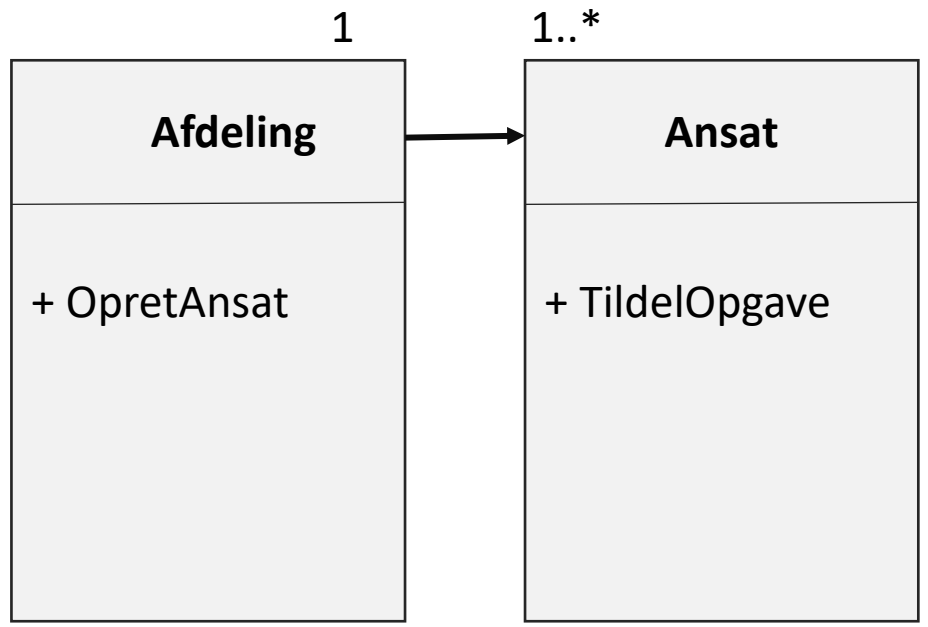


Klasse som skelet eller skabelon for objekter

Modellering af *domænet* i klasser

Ting, fx:
Personer, ansatte, køretøjer, dyr, spilelementer
Funktionalitet, fx:
adgang til datalag, beregningsmetoder indkapslet i objekter, input-objekter med ansvar for
Proces-objekter, fx:
ordre, reservation, booking, møde – data indkapslet i objekter med relevante metoder/adfærd

PROGRAMUDVIKLING & KLASSER



UML-modellering

ANALYSE – SVAR PÅ ”HVAD?”

Hvilke klasser hører til domænet?
Hvilket medlemmer & ansvar har klasserne?
Hvilke relationer har klasserne?

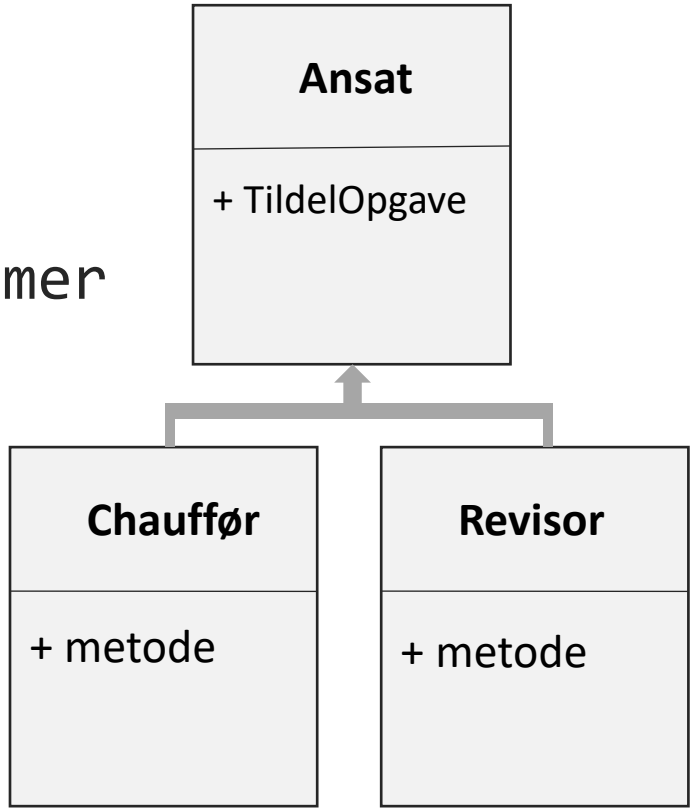
Fx: Applikation til organisation
Ansat
(medlemmer: stilling, navn, alder, opgaver)
Afdeling
(har objekter af ansatte i en-til-mange-relation)
Løn

KLASSER: PRINCIPPER

DESIGN EFTER ANALYSE AF DOMÆNET: HVORDAN
Tilføjelse af fx controller/manager-klasser
som et bud på design af domænet via klasser…

Indkapsling
Private/public – grænseflade til objektets medlemmer
Get & set-metoder af felterne

Nedarving
Baseklasse & subklasse (parent & child)



UML-modellering

KLASSER: PRINCIPPER

I processen frem til at have lavet et program, skal vi igennem en række arbejdsfaser:

Analyse af hvad programmet konkret skal kunne.
Design, der viser hvordan programmet skal laves.
Programmering (implementering) af designet
Test af om programmet svarer til beskrivelsen fra analyse og design.

Klasser & objekter!

Opgave 1: Lav en klasse, fx Person!

- En konstruktør-metode
- Datafelter!
- Metoder!

Lav ny tab ved siden af den primære tab i editoren og skriv fx

```
class Person {
// datafelter – objektets forskellige tilstand
// lav konstruktør-metode som kaldes ved oprettelse af
objekt
// metoder - adfærd lagt ud i metoder, man kan kalde
}
```

Klasser & objekter!

Opgave 1: Lav en klasse, fx Person!

```
class Person {
// datafelter – objektets forskellige tilstande
int hoejde;
String navn;

}
```

Klasser & objekter!

Opgave 1: Lav en klasse, fx Person!

```
class Person {
// lav konstruktør-metode som kaldes ved oprettelse af objekt
// samme navn som Klassen!

Person(int hoejde, String navn){

this.hoejde = hoejde; // this peger på denne klasse
this.navn = navn;
      }
}
```

Klasser & objekter!

Opgave 1: Lav en klasse, fx Person!

```
class Person {
// metoder -  adfærd lagt ud i metoder, man kan kalde

void flytPerson(){
      position.X++;
      }
}

// metoden defineres inde i klassen, men kaldes ude fra klassen
```

Klasser & objekter!

Opgave 1: Lav en klasse, fx Person!

Opret objekt af klassen – klassen er jo en skabelon til nye objekter

```
Person minPerson = new Person(180, ”Brunhilde”);

minPerson.sigNoget(”I am here!”);
// metoden tegner fx taleboble rundt om Brunhilde og skriver i midten ”I am here!”
```

Man kalder metoden via punktum-notationen.