Gruppetime IN1000

Uke 7 - Objektorientert programmering

Program

- Hva er objektorientering?
- Klasser
- Konstruktører
- Innkapsling

Hva er objektorientert programmering?

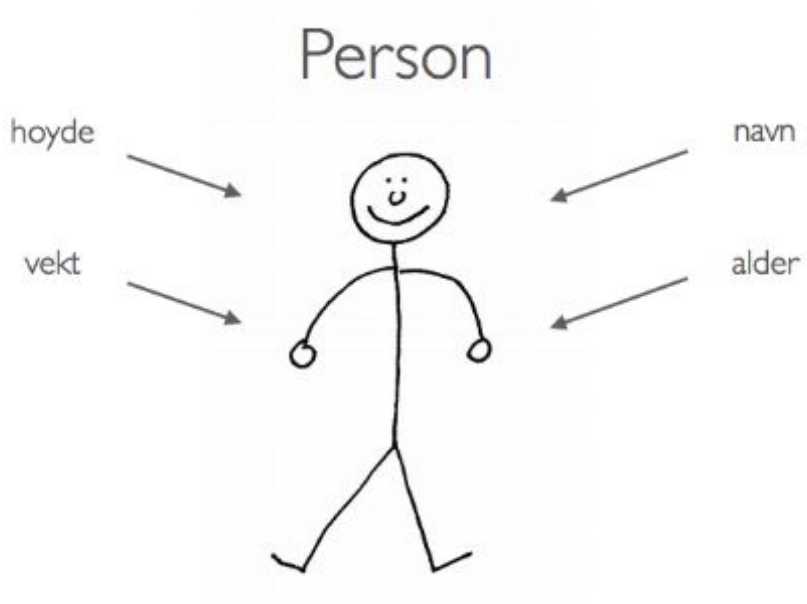
Hva er objektorientert programmering?

- Grunntanke: Lage en “modell” av objekter i den virkelige verden
  - Ikke presis modell, men den informasjonen vi trenger
- Objekter defineres av klasser
  - En slags “blueprint”
- Et objekt har
  - Instansvariabler
  - Metoder
  - Et “grensesnitt” mot omverden

Eksempel person

Her har vi en modell av en person, det vil si de egenskapene hos en person vi er interessert i ved dette programmet.

En person har flere egenskaper i tillegg, men vi programmerer kun de vi er interessert i/har bruk for å lagre.



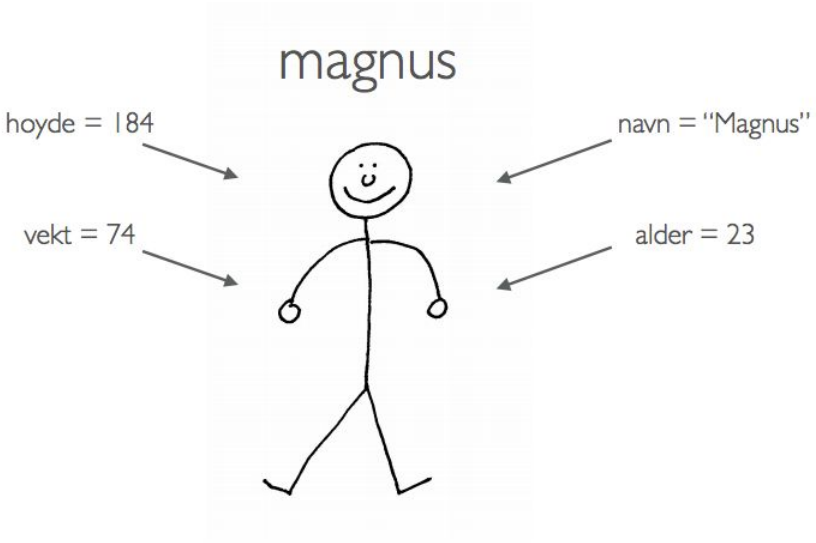
Alle egenskapene vi har lagt til i person er noe som er felles for alle personer!

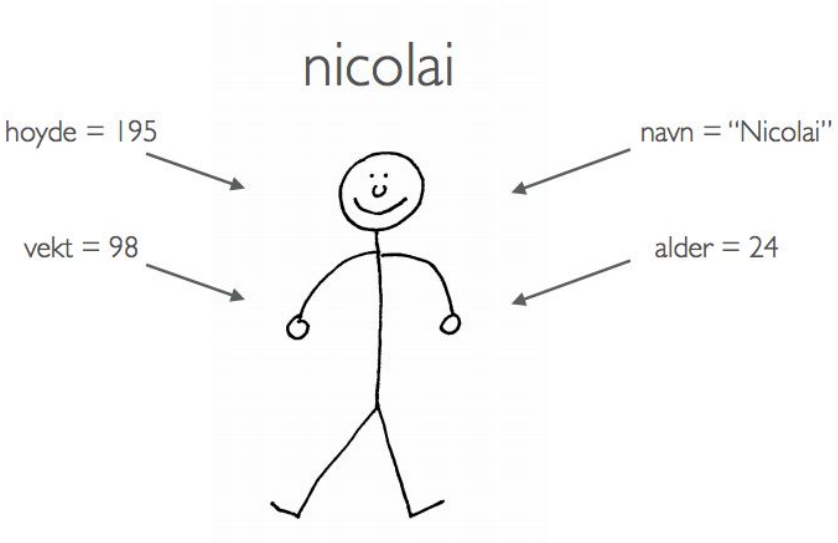
Inne i klassen kaller vi disse egenskapene for *instansvariabler*.

Foreløpig er dette kun en ***mal*** og ikke en person i seg selv.

Eksempel person

Når vi lager en person, lager vi en ***instans*** av klassen Person.





Både *magnus* og *nicolai* er **ulike objekte**r/ulike instanser av et objekt av **samme klasse**. Men vi er ikke bare opptatt av egenskaper, også hvilke **handlinger** objektene kan gjøre.

Syntaks

For å definere en klasse bruker vi nøkkelordet **class:**

```
class Person :
```

I IN1000 har vi én fil per klasse.

Konstruktør
Konstruktøren bestemmer hva som må være med ved opprettelsen av et objekt.
I eksempelet vårt over er dette navn, alder, høyde og vekt.

```
class Klasse :

      def __init__(self, parameter1, parameter2, ….):        #få med to understreker _ _!
            self._verdi = parameter1
           self._verdi2 = parameter2
```

Syntaks - eksempel

```
class Person :

      def __init__(self, navn, alder):
            self._navn = navn
            self._alder = alder

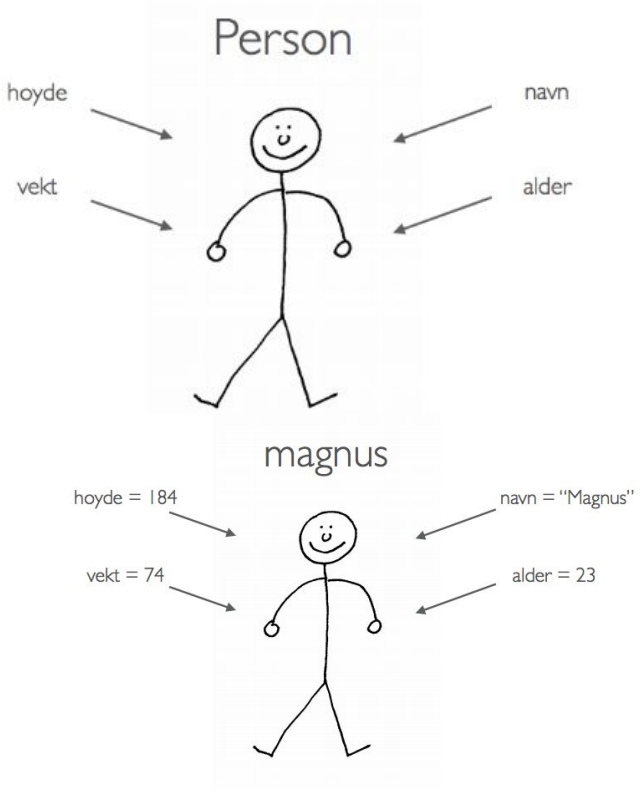

# vi oppretter et person-objekt

person1 = Person("Emilie", “23”)
```

Oppgave 1 - Enkel klasse

Vi skal nå implementere vår første klasse - Person!

Med utgangspunkt i tegningene laget av personene, lag en klasse Person, med de gitte instansvariablene: navn, alder, høyde og vekt.



Oppgave 1 - Løsningsforslag

```
class Person:

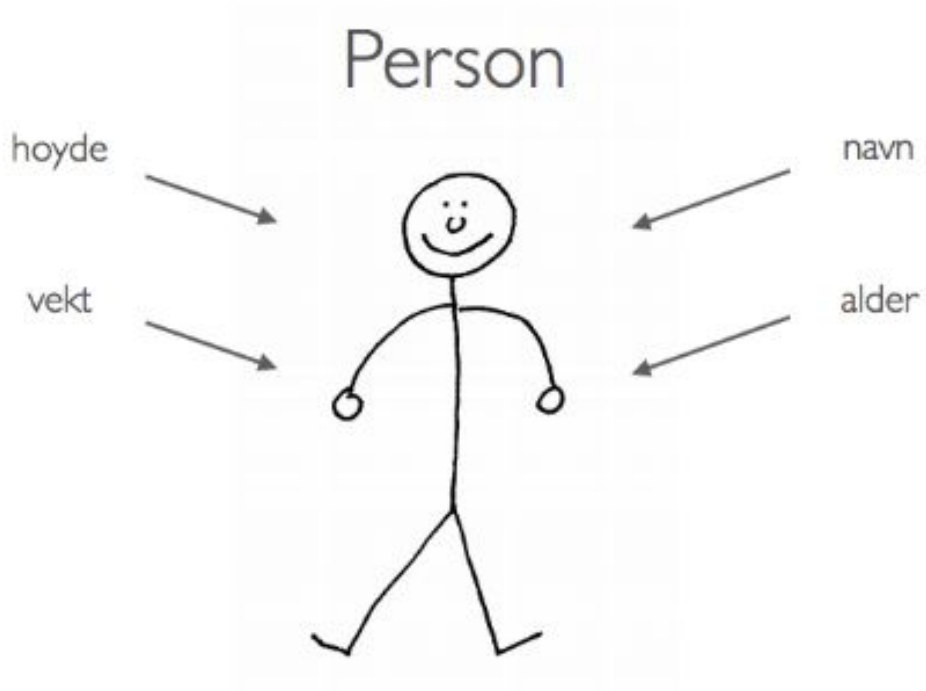
    def __init__(self, navn, alder, høyde, vekt):

        self._navn = navn

        self._alder = alder

        self._høyde = høyde

        self._vekt = vekt
```



Metoder

- I tillegg til at klassen vår skal ha variabler ønsker vi at den skal tilby tjenester
- Dette gjøres via metoder
- Metoder ligner funksjoner, men:
  - Er definert internt i en klasse
  - Må alltid inneholde “self” som første argument
- Metoder er “grensesnittet” til resten av programmet - altså hvordan et objekt kommuniserer med resten av programmet

Grensesnitt

- Eksempel: Radio
  - En radio har knapper og knotter som vi kan trykke på
  - Utgjør “grensesnittet”
  - Vi trenger ikke å vite hvordan disse er implementert - bare hva de gjør!
- Gjør program mye mer oversiktlig
- Å skjule implementasjon bak metoder kalles *innkapsling*
- Eksempel: Vi vet at len(liste) gir oss lengden på en liste, men ikke hvordan dette beregnes.

Eksempel på metode

```
class Klasse :

      def metode_navn(self, param1, ...) :
       #gjør noe

# Vi kaller metoden på et objekt av klassen, for eksempel slik:

person1 = Person("Emilie", ...)
person1.metode_navn(...)
```

Oppgave 2 - utvider med metoder

Vi skal nå utvide Person-klassen vår med følgende metoder:

1. `endre_navn(self, nytt_navn)` → endre navnet til nytt navn.

2. `skriv_ut_hilsen(self):` → skriv ut en hilsen fra personen, på formen “Hei, jeg heter <navn> og jeg er <alder> år gammel”.

3. `feire_bursdag(self)` →
   a. Øk alderen til personen med 1.
   b. Skriv ut en bursdagshilsen med den nye alderen.
   c. Hvis personen fremdeles er under 18 år, skal også høyden økes med 2.

Oppgave 2 - løsningsforslag

```
class Person:
    def __init__(self, navn, alder,
                 høyde, vekt):
        self._navn = navn
        self._alder= alder
        self._høyde = høyde
        self._vekt = vekt

    def endre_navn(self, nytt_navn):
        self._navn = nytt_navn

    def skriv_ut_hilsen(self):
        print(“Hei, jeg heter”,
              self._navn, “og jeg
              er”, self._alder, “år
              gammel”)

    def feire_bursdag(self):
        self._alder += 1

        print(“Hipp hurra!”,
              self._navn, “blir”,
              self._alder, “år!”)

        if self._alder < 18:
             self._høyde += 2
```

Laste inn en klasse

Når vi skal bruke en klasse i en annen fil, må vi laste inn klassen for å bruke den.
Dette gjør vi slik, øverst i filen:

```
from <filnavn/modul> import <Klasse>
```

Har vi klassen Person som er definert i filen *person.py*, kan vi skrive dette i hovedprogrammet vårt:

```
from person import Person
```

Note: filene du importerer fra og til må ligge i samme mappe!

Oppgave 3

Gitt klassen Person som vi har laget, i en egen fil:

1. Opprett tre instanser (objekter) av klassen med navn/alder/vekt/høyde som du velger selv.
2. Skriv deretter ut en hilsen fra hver person ved hjelp av skriv_ut_hilsen-metoden.
3. Feire bursdagen til én av personene, og skriv ut en ny hilsen fra denne personen.
4. Endre navnet til én av personene og skriv ut en ny hilsen fra denne personen.

Oppgave 3 - Løsningsforslag

```
from person import Person


silje = Person(“Silje”, 12, 46, 151)
sofie = Person(“Sofie”, 68, 84, 189)
sara = Person(“Sara”, 25, 76, 191)


silje.skriv_ut_hilsen()
sofie.skriv_ut_hilsen()
sara.skriv_ut_hilsen()
```

```
silje.feire_bursdag()
silje.skriv_ut_hilsen()


sara.endre_navn(“Saga”)
sara.skriv_ut_hilsen()
```

Oppgave 4 - Ekstraoppgave for øving

Skriv en klasse **Dyr**. Et dyr har følgende egenskaper:

- Art
- Kjønn
- Vekt

Lag en konstruktør (init-metode).

Lag en metode som skriver ut informasjonen om dyret.

Opprett tre objekter av klassen Dyr i en annen fil, for å representere tre forskjellige dyr. Skriv ut informasjon om alle dyrene.

Oppgave 4 - Løsningsforslag

I dyr.py:

```
class Dyr:

        def __init__(self, art, kjønn, vekt):

                self._art = art

                self._kjønn = kjønn

                self._vekt = vekt



        def skriv_info(self):

                print("Art: ", self._art, ", kjønn: ",
self._kjønn, ", vekt: ", self._vekt)
```

I hovedfilen:

```
hund = Dyr("hund", "hann", 10.2)

hund.skriv_info()

katt = Dyr("katt", "hunn", 5.7)


katt.skriv_info()

sau = Dyr("sau", "hann", 30.6)

hund.skriv_info()
```