OO-DESIGN: KLASSER, OBJEKTER OG REFERANSER

REPETISJONSKURS IN1000 H18
CHRISTINE MOEN WINGE (CHRISMWI)

HVA ER OBJEKTORIENTERING?

- Programmering som for det meste består av bruk av objekter og referanser
- Virkelighetsnær programmering
- Objekter er representasjoner av ting fra virkeligheten.
- Typisk substantiver
- Programmet får objekter til å interagere med hverandre
- Simulerer virkeligheten

KLASSER

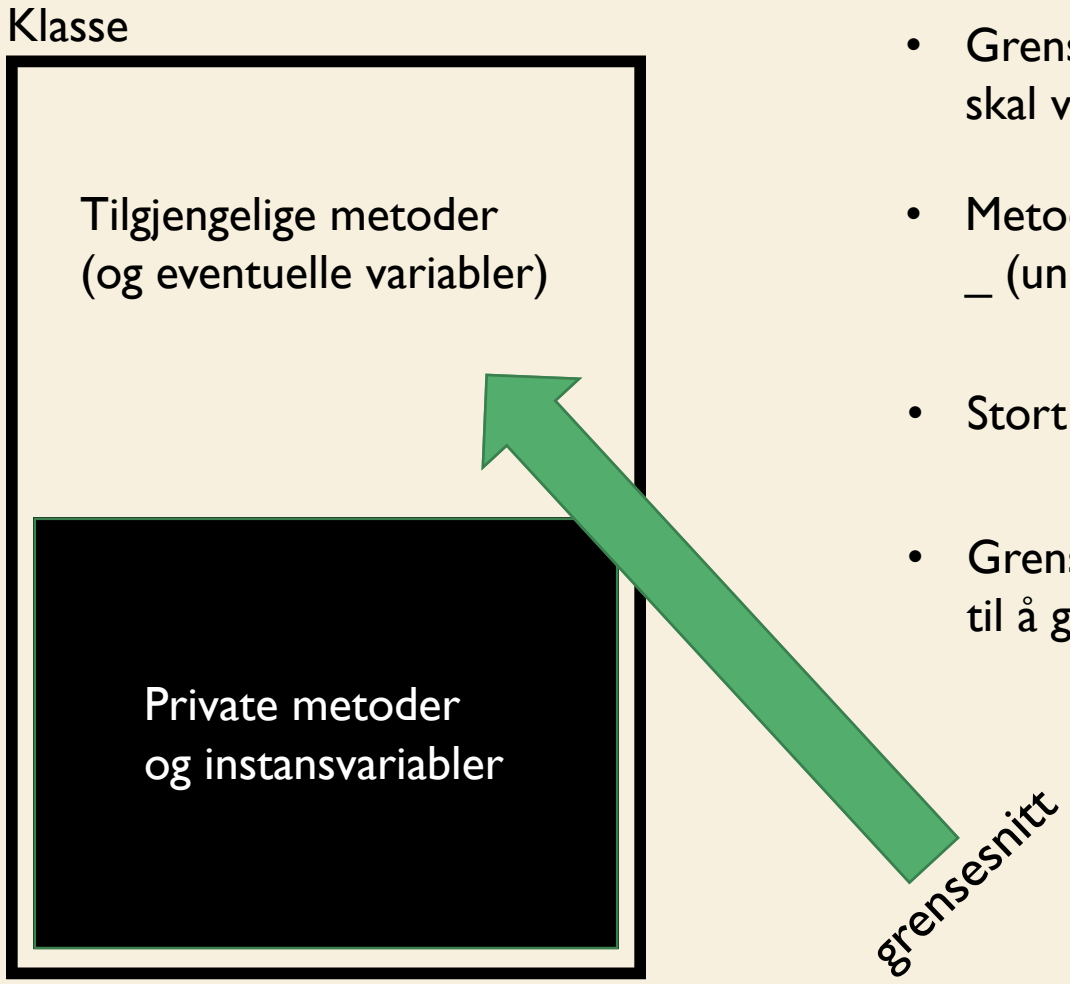
```
class klassenavn:
    #opprette konstruktør
    def __init__(self, eventuelleParametere):
        #opprette instansvariabler

    #opprette metoder
```

- Konstruktør:
  Metode som automatisk blir kalt på under opprettelsen av et objekt

- Instansvariabler:
  Variabler som skal kunne aksesseres av hele klassen

- Metoder:
  Funksjoner som finnes inne i en klasse

Grensesnitt



- Grensesnittet består av det innholdet i klassen som skal være tilgjengelig utenifra
- Metoder og variabler som har navn som starter med _ (understrek) skal holdes private
- Stort sett er instansvariabler private
- Grensesnittet beskriver hva vi har lov til å gjøre med et objekt

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

    def hentNavn(self):
        return self._navn

    def blirEldre(self):
        self._alder += 1
```

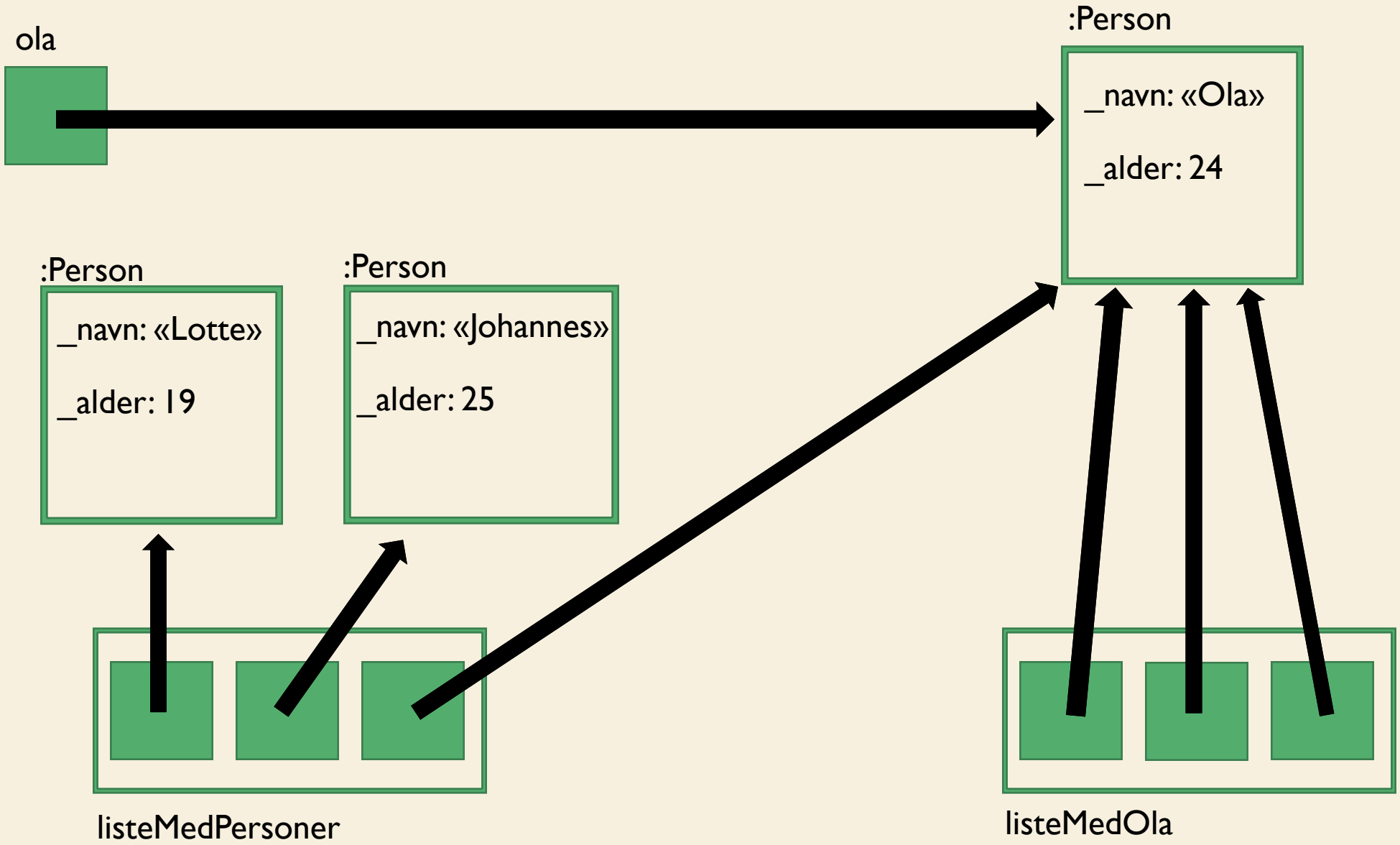
Eksempel på grensesnitt

REFERANSER OG OBJEKTER

- En klasse er en oppskrift/modell, beskriver hvordan et objekt skal se ut og hva slags egenskaper det skal ha
- Et objekt er en instans av en klasse, et modellert eksemplar av klassen
- En referanse er en ting som peker til et objekt (inneholder minneadressen til et spesifikt objekt)
- Når vi manipulerer objekter, gjør vi det via referansene
- Objekter kan inneholde referanser til andre objekter!

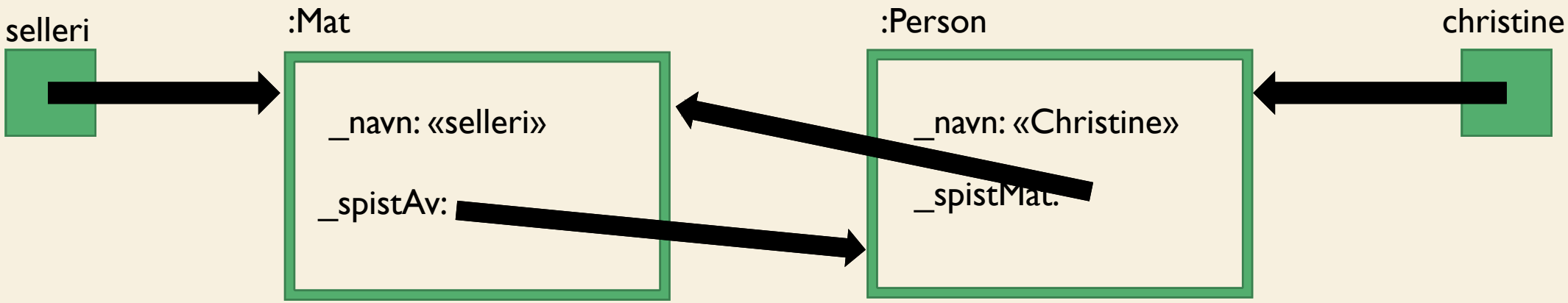
```
ola = Person("Ola", 24)
print(ola.hentNavn())

listeMedPersoner = [Person("Lotte", 19), Person("Johannes", 25), ola]

listeMedOla = [ola, ola, ola]
```

- Variabelen «ola» er en referanse til et spesifikt Person-objekt.
- Vi bruker referansen til å hente ut navnet
- I listen «listeMedPersoner» ligger det tre referanser til tre forskjellige Person-objekter
- I listen «listeMedOla» ligger det tre referanser til det samme objektet.


ola
:Person
_navn: «Ola»
_alder: 24
:Person
_navn: «Lotte»
_alder: 19
:Person
_navn: «Johannes»
_alder: 25
listeMedPersoner
listeMedOla

KRYSSREFERANSER

```
selleri = Mat("selleri")
christine = Person("Christine")

christine.spisMat(selleri)
selleri.bliSpistAv(christine)
```



DATASTRUKTUR

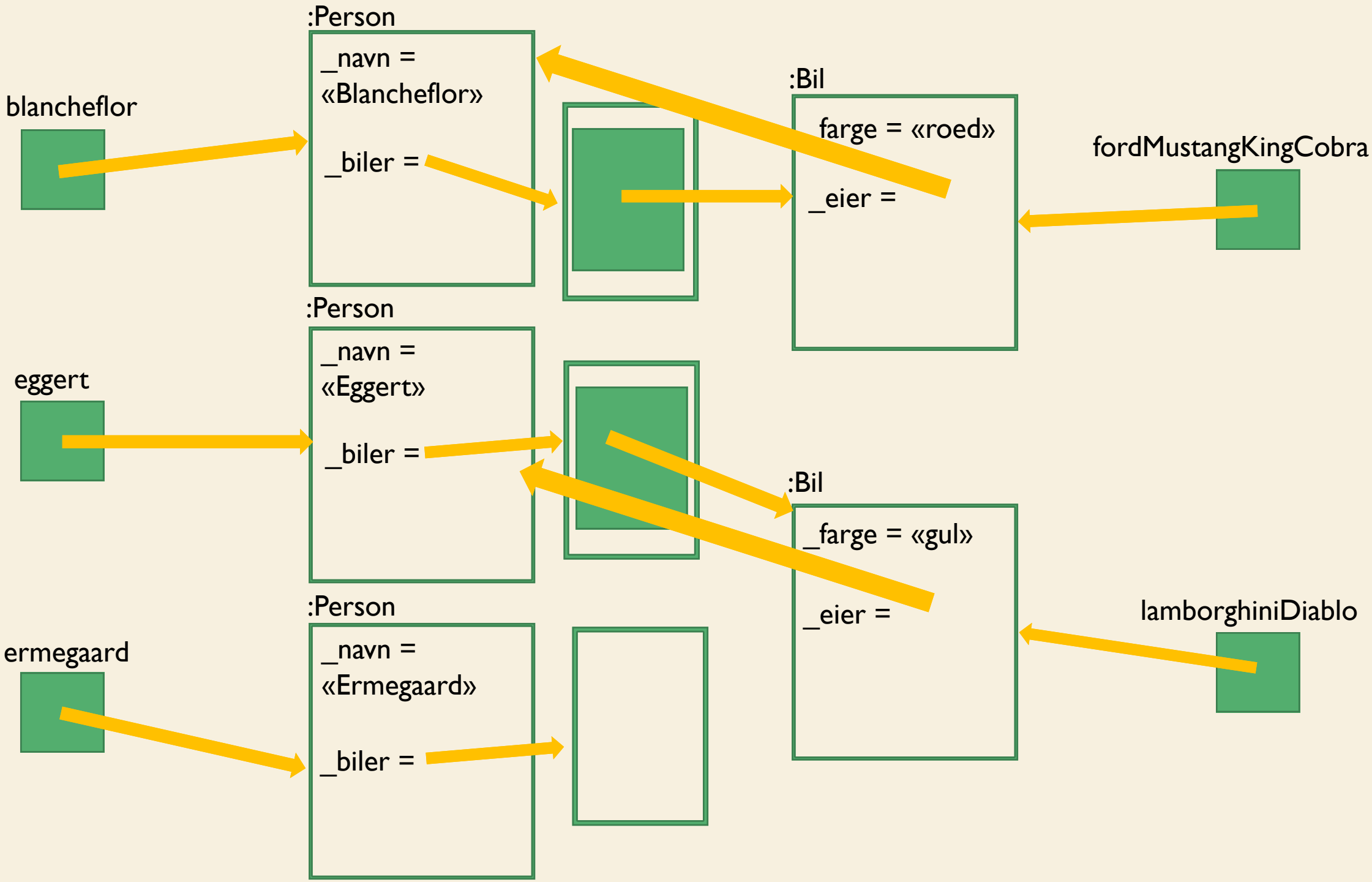
- Viser en tilstand!
  Hvordan dataene/objektene er organisert på et gitt tidspunkt

```python
class Person:
    def __init__(self, navn):
        self._navn = navn
        self._biler = []

    def kjopBil(self, bil):
        self._biler.append(bil)

class Bil:
    def __init__(self,farge):
        self._farge = farge
        self._eier = None

    def settEier(self, eier):
        self._eier = eier
```

```python
blancheflor = Person("Blancheflor")
eggert = Person("Eggert")
ermegaard = Person("Ermegaard")

fordMustangKingCobra = Bil("roed")
lamborghiniDiablo = Bil("gul")

blancheflor.kjopBil(fordMustangKingCobra)
fordMustangKingCobra.settEier(blancheflor)

eggert.kjopBil(lamborghiniDiablo)
lamborghiniDiablo.settEier(eggert)
```

Hvordan vil datastrukturen se ut når dette har kjørt?

Bruk noen minutter på å diskutere


:Person
_navn =
«Blancheflor»
_biler =
blancheflor
:Bil
farge = «roed»
_eier =
fordMustangKingCobra
:Person
_navn =
«Eggert»
_biler =
eggert
:Bil
_farge = «gul»
_eier =
lamborghiniDiablo
:Person
_navn =
«Ermegaard»
_biler =
ermegaard

SPØRSMÅL?

```
pellePolitibil = Bil("pelle politibil")
brannmannSam = Person("Brannmann Sam")

pellePolitibil._eier = brannmannSam
```

```
pellePolitibil = Bil("pelle politibil")
brannmannSam = Person("Brannmann Sam")

pellePolitibil.settEier(brannmannSam)
```

Diskuter forskjeller og ulikheter ved disse kodesnuttene. Hvilken synes dere er best?

```python
listeNoder = []

for i in range(10):
    node = Node(1024, 2)
    listeNoder.append(node)
```

```python
listeNoder = []

node = Node(1024, 2)
for i in range(10):
    listeNoder.append(node)
```

Diskuter forskjeller og ulikheter ved disse kodesnuttene. Hva skjer i hver av dem?

Har jeg gjort noe galt her? Hvorfor/hvorfor ikke? Hva kan konsekvensene være?

```
class Stasjon:
    def __init__(self, navn):
        self._navn = navn
        self._neste = None

    def settNeste(self, neste):
        self._neste = neste

    def hentNeste(self):
        return self._neste

blindern = Stasjon("Blindern")

blindern._navn = "Forskningsparken"
```

Jeg vil at stasjonene skal ligge strukturert i denne rekkefølgen:

blindern → forskningsparken → ullevaalStadion

De må derfor ha riktige referanser i sine _neste-variabler

Hvordan gjør jeg dette?
Diskuter!

```
class Stasjon:
    def __init__(self, navn):
        self._navn = navn
        self._neste = None

    def settNeste(self, neste):
        self._neste = neste

    def hentNeste(self):
        return self._neste

blindern = Stasjon("Blindern")

forskningsparken = Stasjon("Forskningsparken")

ullevaalStadion = Stasjon("Ullevål Stadion")
```

Hva printes ut?
Diskuter

```python
class Hus:
    def __init__(self, farge):
        self._farge = farge

    def settFarge(self, nyFarge):
        self._farge = nyFarge

    def hentFarge(self):
        return self._farge


hus1 = Hus("blå")

hus2 = hus1

hus3 = Hus("grønn")

hus1 = hus3

hus2.settFarge("gul")

print(hus1.hentFarge())
```

Kahoot!