
IN1010 Våren 2026
Intro til objektorientering
Eyvind W. Axelsen
Bilde fra https://pixnio.com/

```java
public class GodEttermiddag {
  public static void main(String[] args) {
    System.out.println("Hei på deg!");
    System.out.println("Du bør kunne lese dette - sitter du for langt bak?");
  }
}
```

Hvem er jeg?

Eyvind W. Axelsen
(eyvinda@ifi.uio.no)

Førsteamanuensis II ved Ifi, UiO

- Det vil si, jeg er deltidsansatt
- Programmeringsteknologi-gruppa (10. etasje)
- Underviser *IN2031 – Prosjektoppgave i programmering* til høsten (*sannsynligvis)



CTO/IT-direktør ved Fürst Medisinsk Laboratorium

- Norges største medisinske laboratorium, > 500 ansatte
- > 12 000 pasientprøver per dag, > 200 000 analyser per dag
- Lager "ordentlige" programmer daglig i språk som C#, TypeScript og SQL















- Gift med Fride
- To etter hvert ganske store barn + bikkje
- Glad i musikk (og programmering, så klart)

I dag: intro til objektorientering

- Objekter, tegninger og representasjon i Java
- Variabler og datatyper
- Referanser
- Statiske metoder og variabler
- Arrayer
- Metodekall og parametere



Objektorientering handler om å modellere ting og konsepter fra verden som vi er interesserte i

- Hva slags ting da?

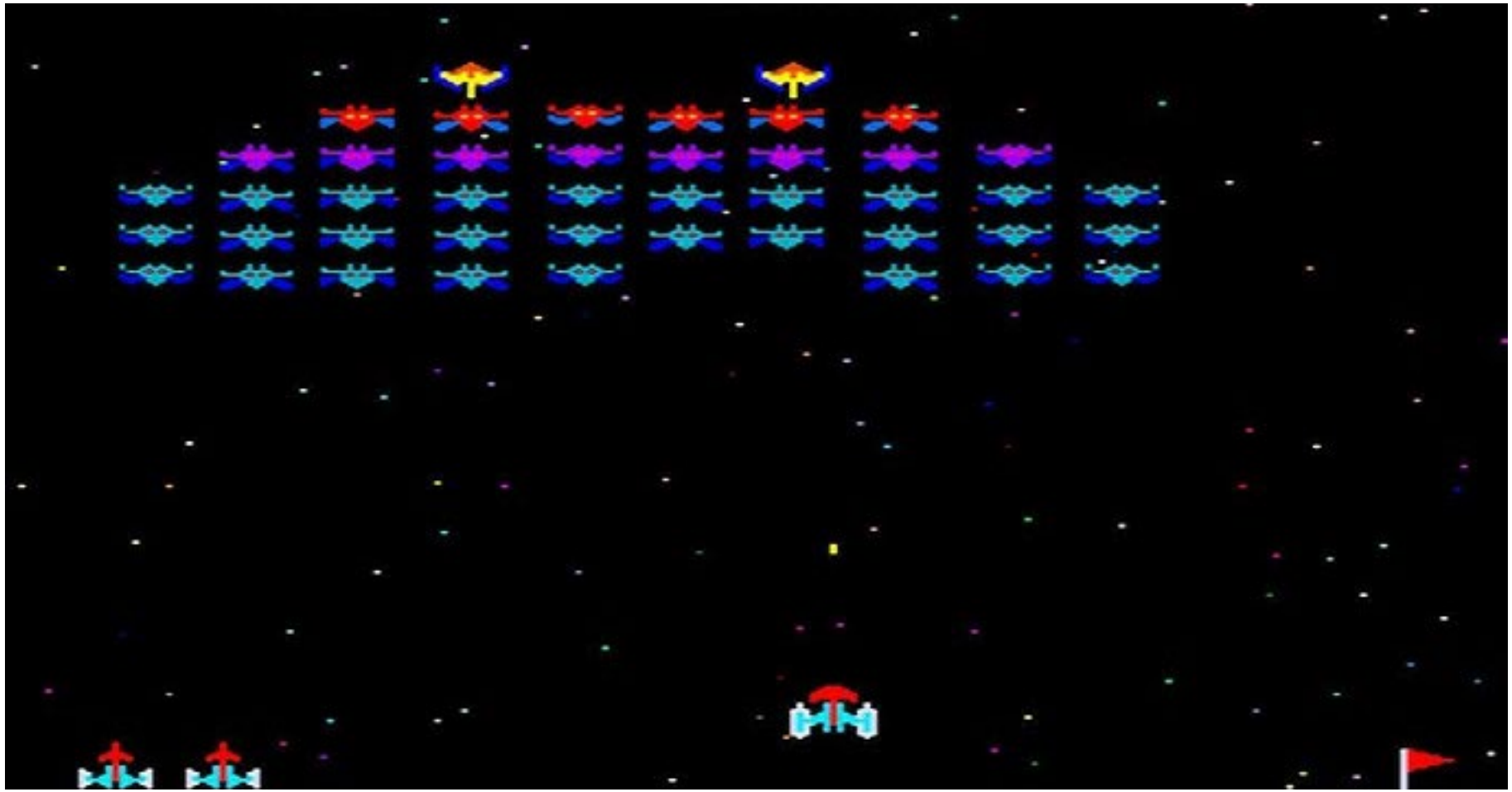
Objektorientering handler om å modellere ting og konsepter fra verden som vi er interesserte i

- Hva slags ting da? Studenter



Objektorientering handler om å modellere ting og konsepter fra verden som vi er interesserte i

- Hva slags ting da? Romskip og fiender i et spill



Objektorientering handler om å modellere ting og konsepter fra verden som vi er interesserte i

- Hva slags ting da? Hunder



Objektorientering handler om å modellere ting og konsepter fra verden som vi er interesserte i

- Hva slags ting da?

- Kort fortalt: alle slags ting som er interessante for vårt problem/ system





I dette bildet er det mange ting

- Oline (min datter)
- 2 hunder, Ego og Luca
- LP-plater
- Et bilde av Bjørn (min sønn)
- Reoler
- En hatt
- En slags beholder av noe slag
- En del rot…



Når vi lager et system, så må vi bestemme oss for hva som er *relevant* å ha med for det *problemet* vi forsøker å løse

Hunder er trivelige dyr

Dersom vi ønsker å modellere en hund i vårt program, hva trenger vi da?

Informasjon/data

- Navn
- Rase
- Vekt
- Alder
- Eier
- …



Hunder er trivelige dyr

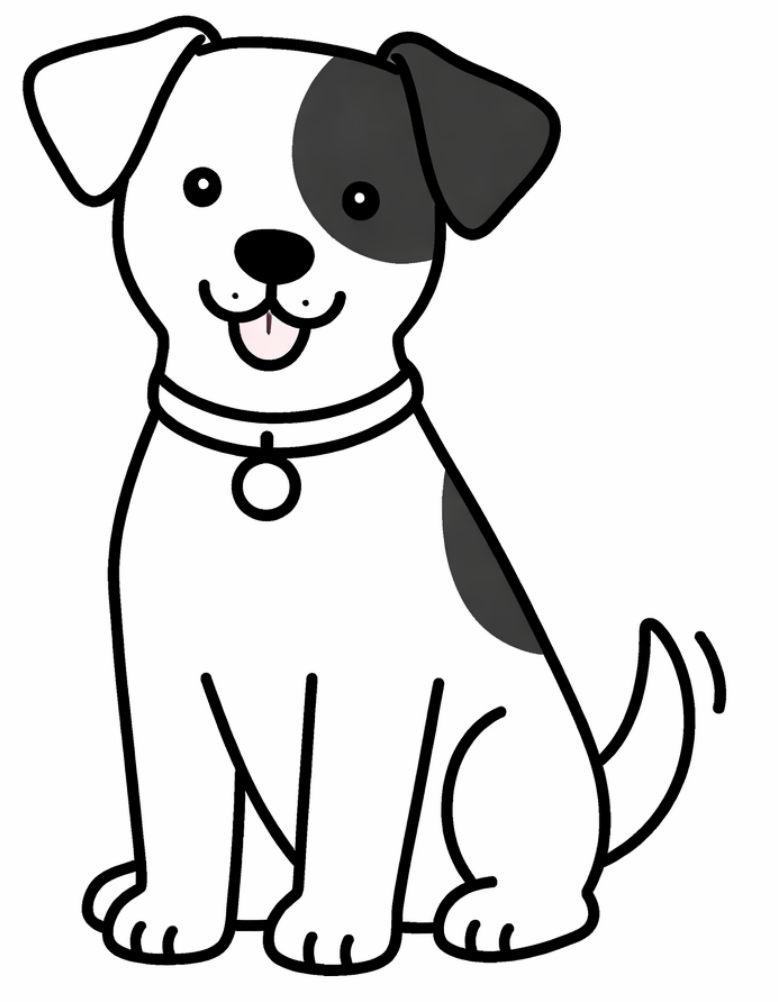
Dersom vi ønsker å modellere en hund i vårt program, hva trenger vi da?

Oppførsel/handlinger

- Bjeffe
- Spise
- Sove
- Sitte
- Logre
- …



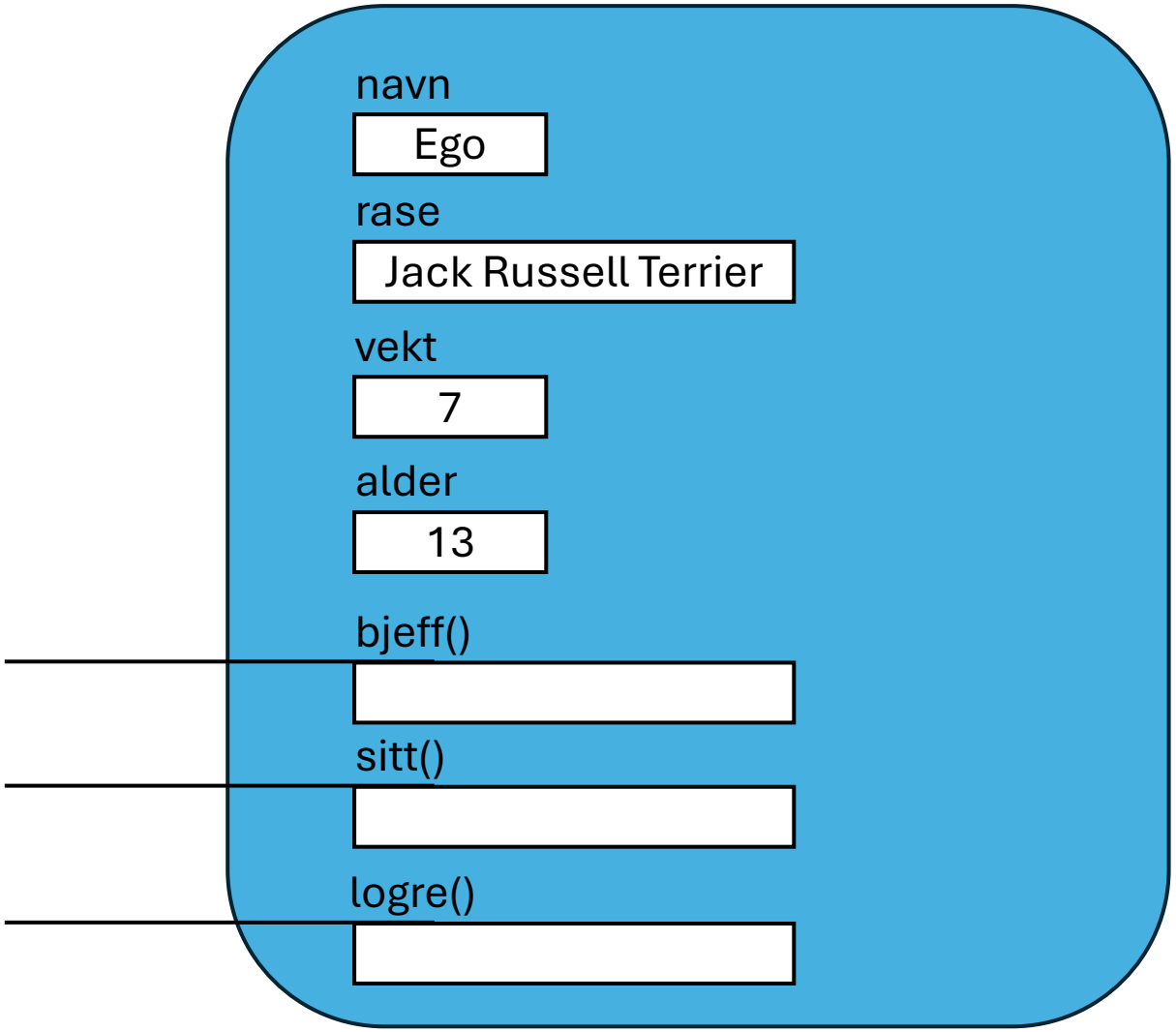
Hvordan tegner vi objekter?



Data
Navn: Ego
Rase: Jack Russell Terrier
Vekt: 7 kg
Alder: 13

Oppførsel
Bjeff()
Sitt()
Logre()
...

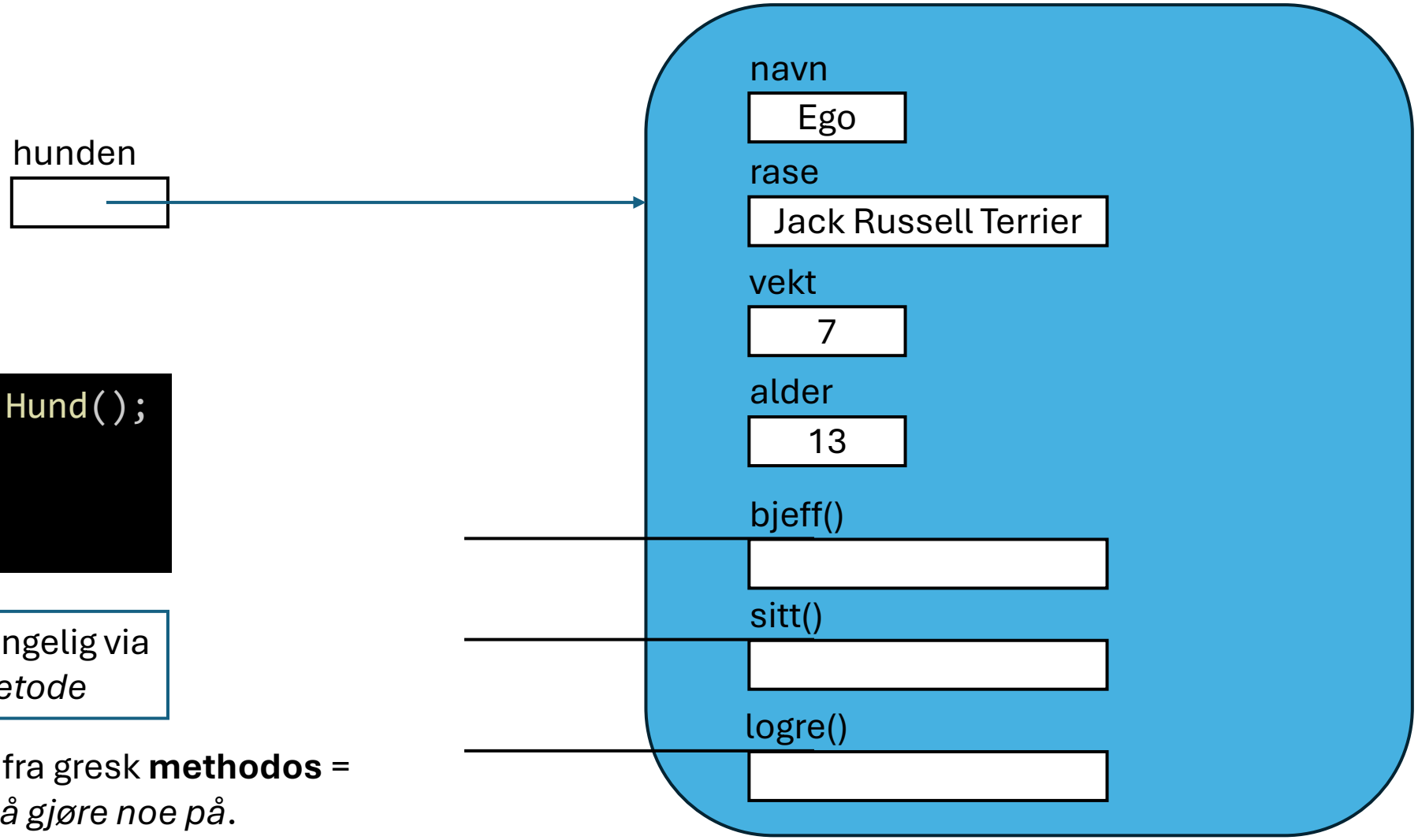
Hvordan tegner vi objekter?



Data
Navn: Ego
Rase: Jack Russell Terrier
Vekt: 7 kg
Alder: 13

Oppførsel
Bjeff()
Sitt()
Logre()
…

Et objekt tilbyr et *grensesnitt* til omverdenen



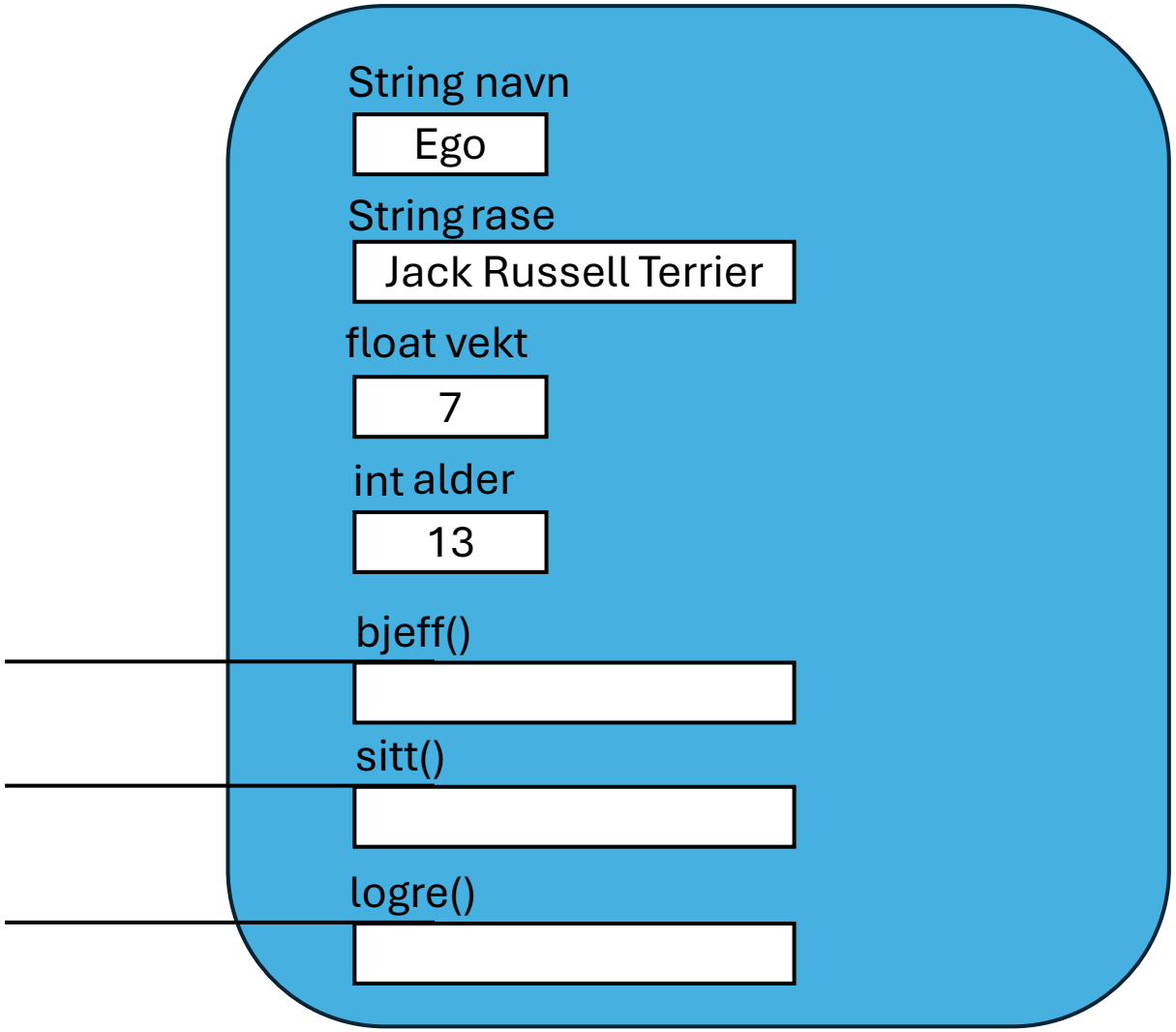

```java
Hund hunden = new Hund();
// ...
hunden.sitt();
```

Oppførsel som er tilgjengelig via et objekt kaller vi en *metode*

Ordet *metode* kommer fra gresk **methodos** = *fremgangsmåte / måte å gjøre noe på*.

I Java så har både data og metoder en *type*

- Vi angir typen *eksplisitt*, til forskjell fra f.eks. Python
- For variabler/data angir vi typen data



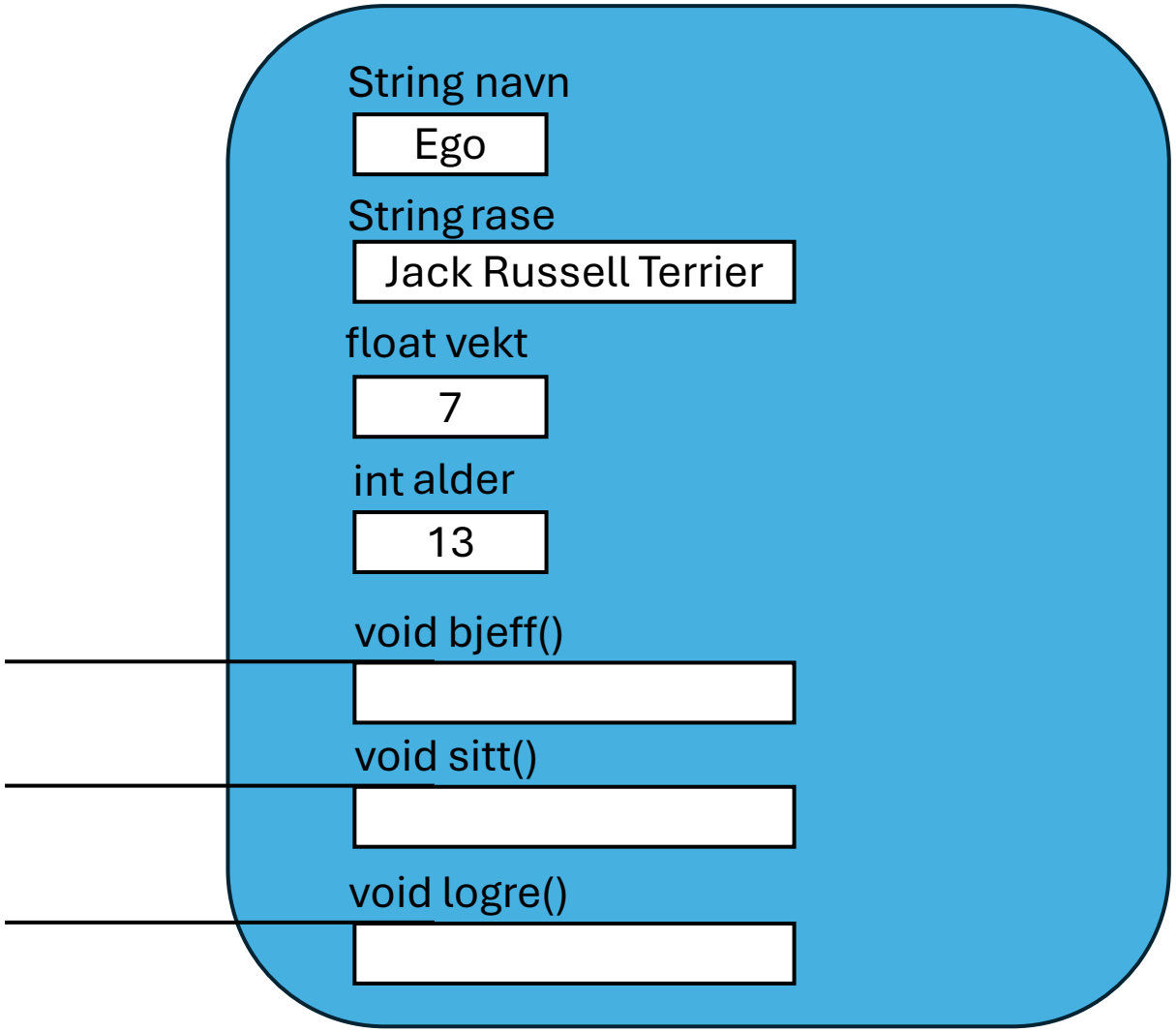
En Hund i Java og Python

```java
class Hund {
    String navn;
    String rase;
    float vekt;
    int alder;
}
```

```python
class Hund:
    def __init__(self):
        self.navn = ""
        self.rase = ""
        self.vekt = 0
        self.alder = 0
```

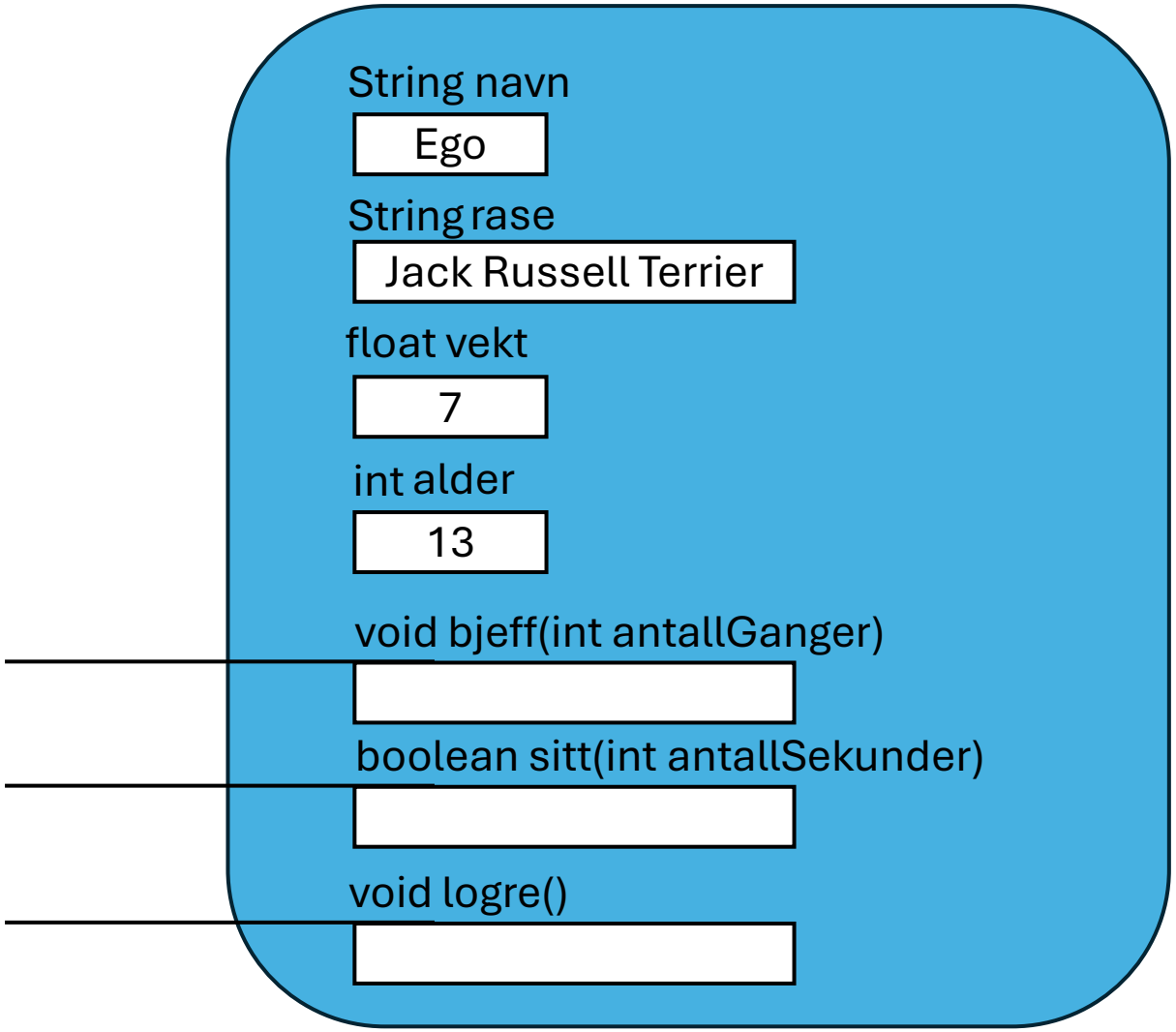
I Java så har både data og metoder en *type*

- Vi angir typen *eksplisitt*, til forskjell fra f.eks. Python
- For variabler/data angir vi typen data
- For metoder angir vi returtype og typen til eventuelle parametere



I Java så har både data og metoder en *type*

- Vi angir typen *eksplisitt*, til forskjell fra f.eks. Python
- For variabler/data angir vi typen data
- For metoder angir vi returtype og typen til eventuelle parametere



```java
class Hund {
    String navn;
    String rase;
    float vekt;
    int alder;


    public void bjeff(int antallGanger) {
        /* ... */
    }
    public boolean sitt(int antallSekunder) {
        /* ... */
    }
    public void logre() {
        /* ... */
    }
}
```

```python
class Hund:
    def __init__(self):
        self.navn = ""
        self.rase = ""
        self.vekt = 0
        self.alder = 0


    def bjeff(antallGanger):
        pass
    def sitt(antallSekunder):
        pass
    def logre():
        pass
```

Variabler

En variabel i Java er et *navngitt* sted i *minnet* som inneholder en *verdi* av en gitt *type*.

Typen til en variabel kan ikke forandres (og navnet kan heller ikke forandres).

Innholdet til en variabel forandres ved tilordning.

```
int antallStudenter = 900;
int etteranmeldte = 42;

antallStudenter = antallStudenter + etteranmeldte;
```

antallStudenter	00000000000000000000001110101110	Type: int
etteranmeldte	00000000000000000000000000101010	Type: int

Datatyper for *verdier* i Java

Datatype	Størrelse	Eksempelverdier
boolean	8 bit (selv om vi strengt tatt bare trenger 1 bit)	true false
byte	8 bit	-128 til 127
char	16 bit	‘A’, ‘0’, ‘€’, alle unicode-tegn
short	16 bit	-32 768 til 32 767
int	32 bit	-2 147 483 648 (-2^{31}) til 2 147 483 647 ($2^{31} - 1$)
long	64 bit	-9 223 372 036 854 775 808 (-2^63) til 9 223 372 036 854 775 807 (2^63 - 1)
float	32 bit	Desimaltall, f.eks. 3.1415927
double	64 bit	Desimaltall, f.eks. 3.141592653589793

Verdier og referanser

Java skiller på referanser og verdier. En *verdi* er av en typene vi så på forrige slide.

En *referanse* peker på et objekt, eller den er **`null`**.

(Når vi sier at en referanse peker på et objekt, så mener vi egentlig at den inneholder adressen til en plassering i minnet på maskinen, hvor objektet befinner seg.)

Hund hundenTilOline

Hund hundenTilBjørn

Hund hundenTilPostmannPat

null

String navn

Ego

String rase

Jack Russell Terrier

float vekt

7

int alder

13



Hund hundenTilNora

String navn

Luca

String rase

Cobberdog

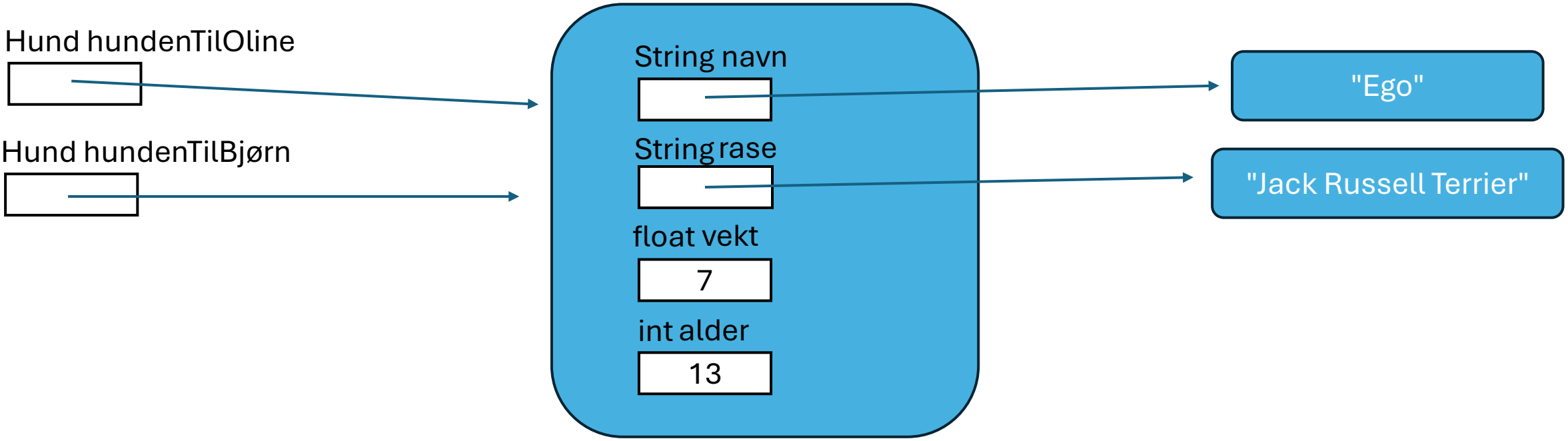
float vekt

8,2

int alder

5

String er også egentlig en referansetype, og strenger er objekter på samme måte som hunder



Men, vi trenger ikke alltid tegne alt; vi holder oss til det vi har bruk for og nytte av

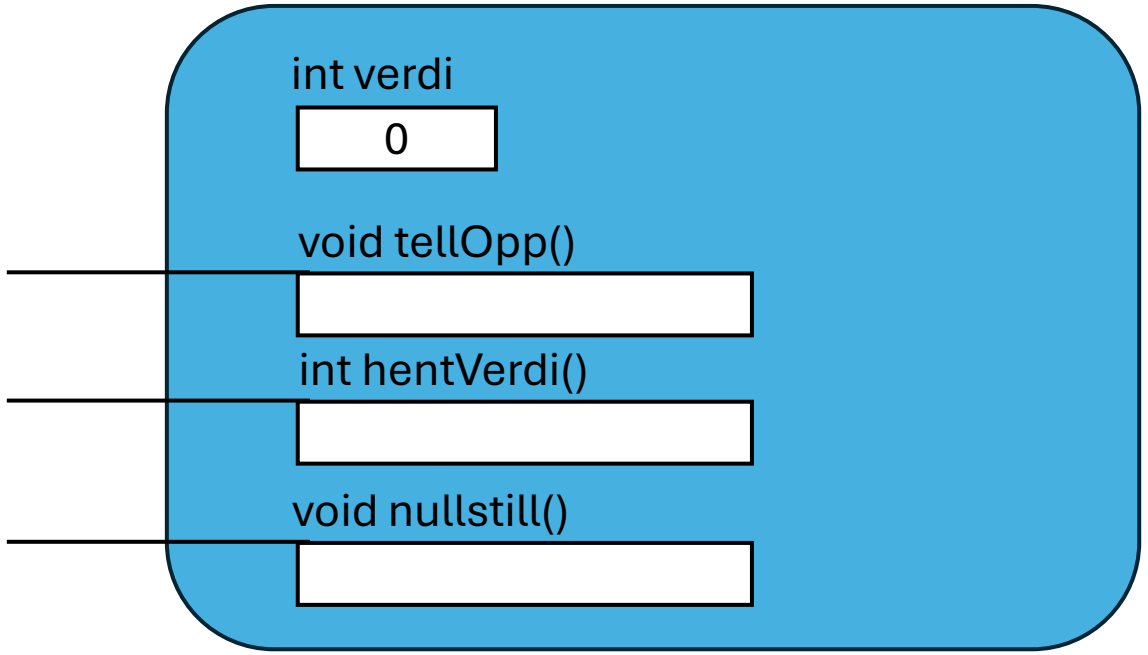
Objekter i Java: en enkel teller

- En teller, som f.eks. betjeningen på et fly kan bruke, må ha funksjonalitet for å
  - Øke telleren med 1
  - Lese av nåværende verdi
  - Resette verdien til 0 igjen
- Kanskje det enkleste fornuftige eksempelet vi kan lage?



Objekter i den virkelige verden modelleres av objekter i programmet (under kjøring)

- Et objekt tilbyr tjenester (oppførsel) til omverdenen ved hjelp av sine metoder
- Resten av implementasjonen er skjult (black box-tenking)
- Når vi programmerer klassen til objektet, så kan vi se alle detaljene





La oss se om vi kan programmere dette sammen

For å teste den nye klassen vår trenger vi en testklasse, med en main-metode

- Hva skal den kunne gjøre?

- Lage et objekt av telleren
- Øke verdien noen ganger
- Lese av verdien, og lagre den i en variabel
- Sjekke at resultatet ble riktig



La oss se om vi kan programmere dette sammen

Hva er alle disse greiene?

```
public static void main(String[] args) {
    // ...
}
```

public: metoden er *offentlig*, altså tilgjengelig for alle

static: metoden tilhører *klassen*, og ikke objekter av klassen

void: metoden returnerer ingenting

main: programmet starter her

String[] args: en array (mer senere) av argumenter/parametre til programmet

Litt mer om static

- En klasse er i utgangspunktet et «mønster» som vi kan lage objekter ut av, ved bruk av new
- I tillegg kan vi deklarere data og metoder som tilhører klassen (og som da vil være felles for alle objekter av klassen)

```
class Teller {
   private static int totalAntall = 0;
   private int verdi;

   public static int hentTotAnt ( ) {
      return totalAntall;
   }

   public void tellOpp( ) {
      verdi++;
      totalAntall++;
   }

   public int hentVerdi( ) {
      return verdi;
   }

   public void nullstill( ) {
      totalAntall -= verdi;
      verdi = 0;
   }
}
```

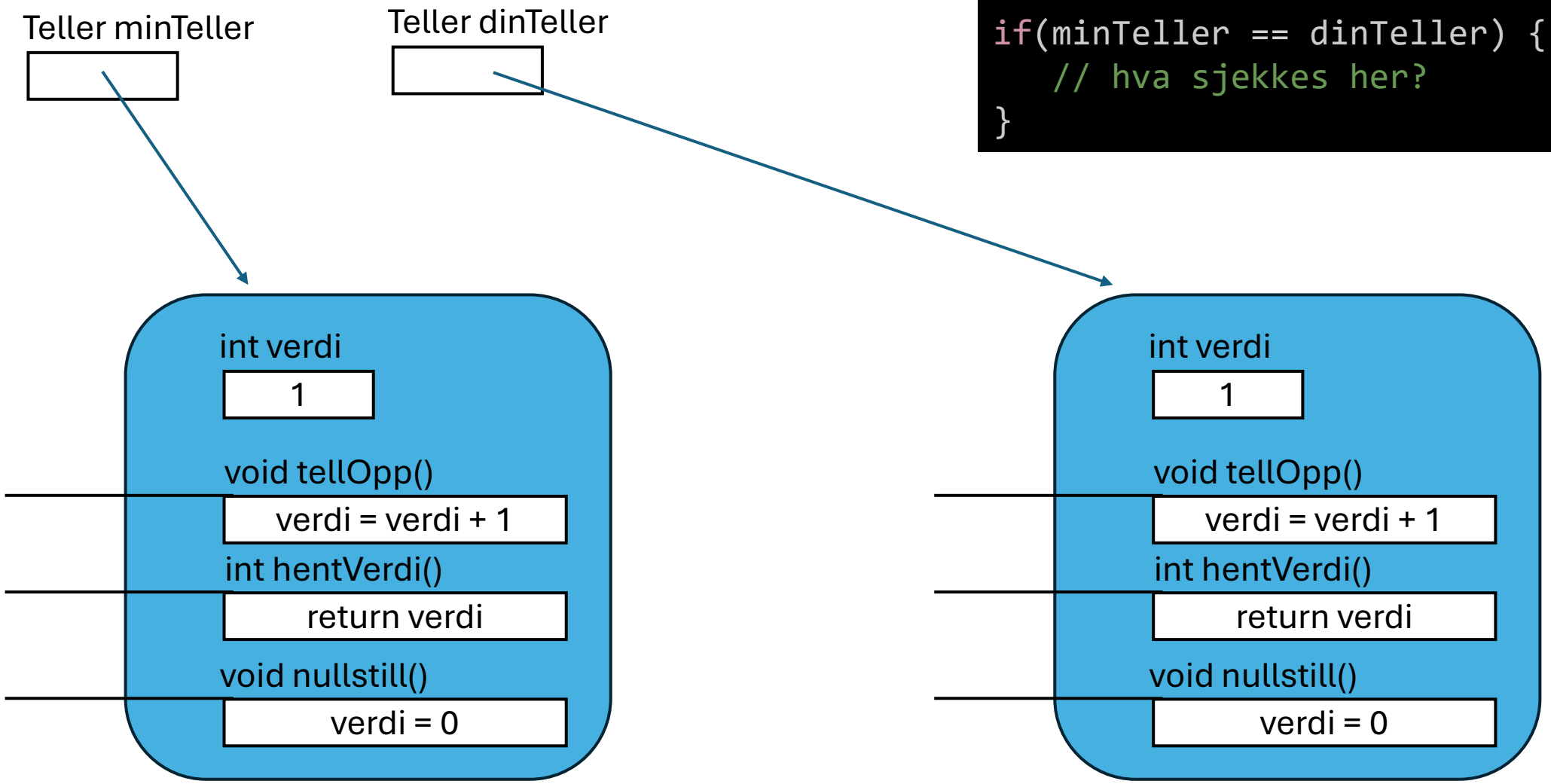
La oss prøve det ut!



Litt mer om referanser

```
Teller minTeller = new Teller();
Teller dinTeller = new Teller();
minTeller.tellOpp();
dinTeller.tellOpp();

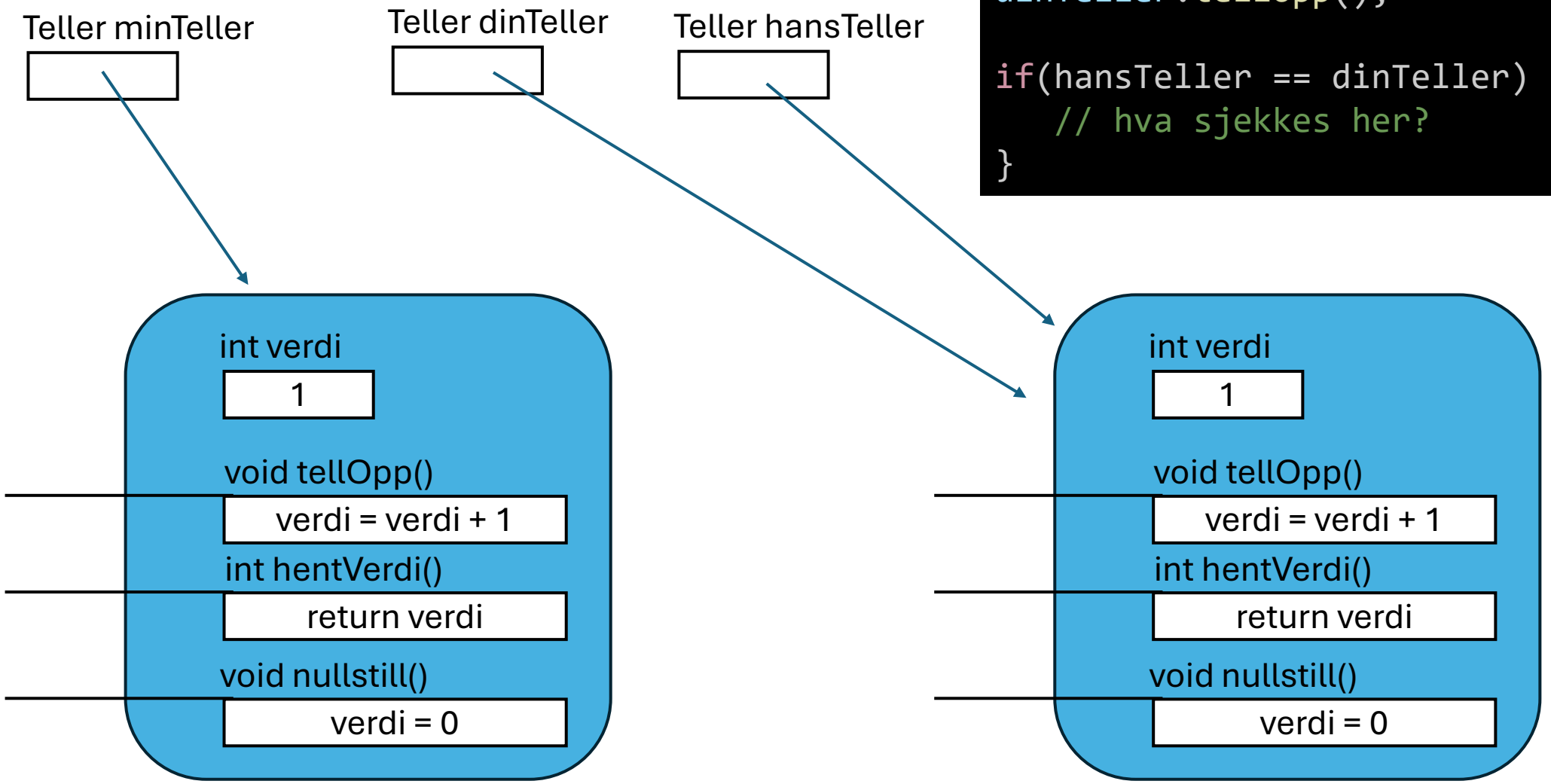
if(minTeller == dinTeller) {
   // hva sjekkes her?
}
```



Litt mer om referanser

```
Teller minTeller = new Teller();
Teller dinTeller = new Teller();
Teller hansTeller = dinTeller;
minTeller.tellOpp();
dinTeller.tellOpp();

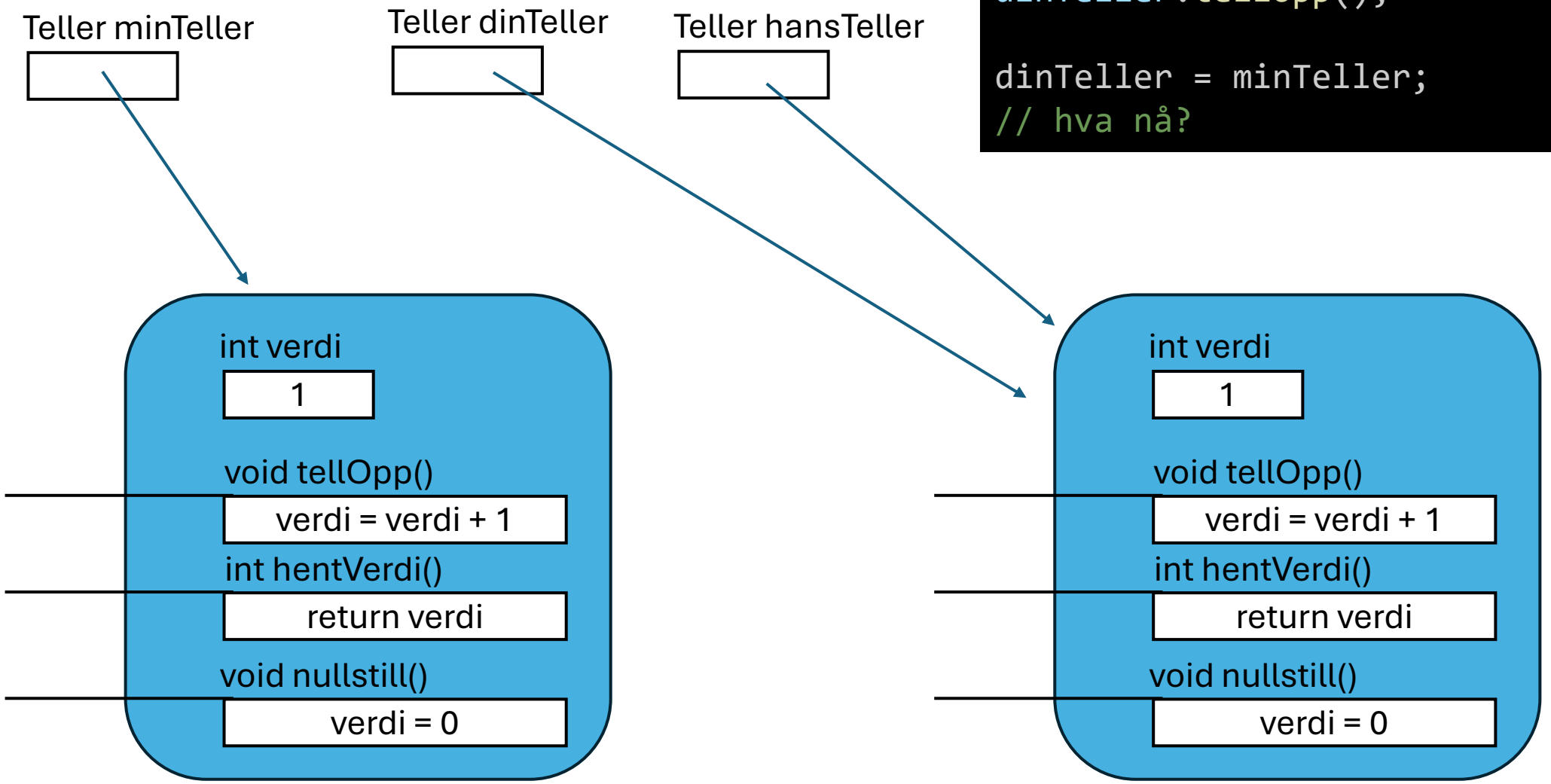
if(hansTeller == dinTeller) {
   // hva sjekkes her?
}
```



Litt mer om referanser

```
Teller minTeller = new Teller();
Teller dinTeller = new Teller();
Teller hansTeller = dinTeller;
minTeller.tellOpp();
dinTeller.tellOpp();

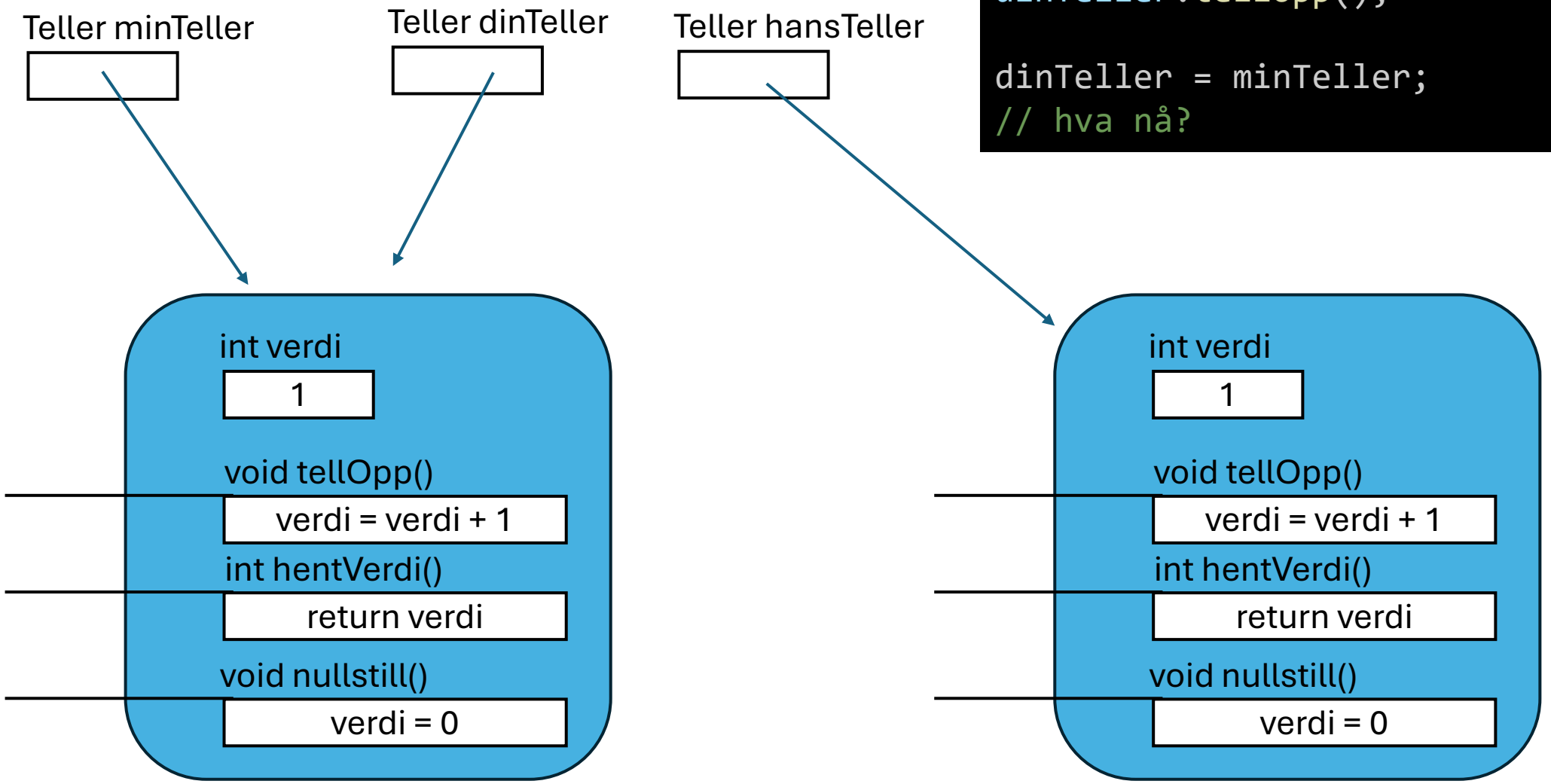
dinTeller = minTeller;
// hva nå?
```



Litt mer om referanser

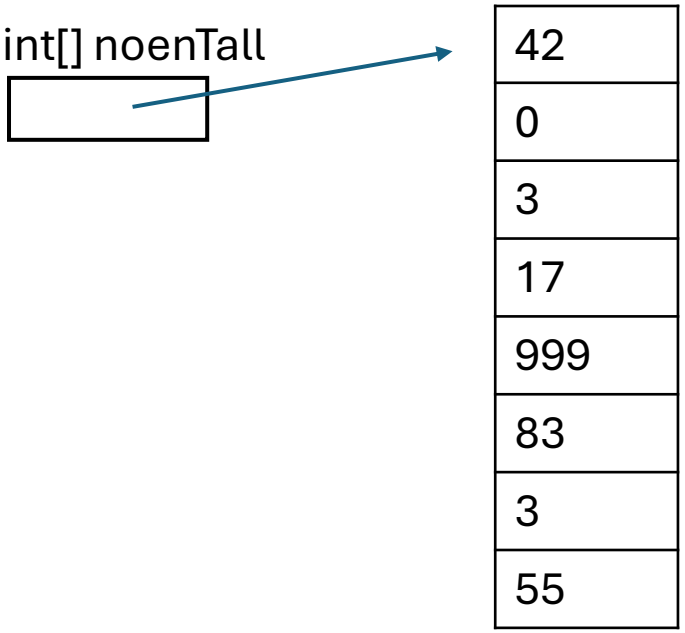
```
Teller minTeller = new Teller();
Teller dinTeller = new Teller();
Teller hansTeller = dinTeller;
minTeller.tellOpp();
dinTeller.tellOpp();

dinTeller = minTeller;
// hva nå?
```



Arrayer

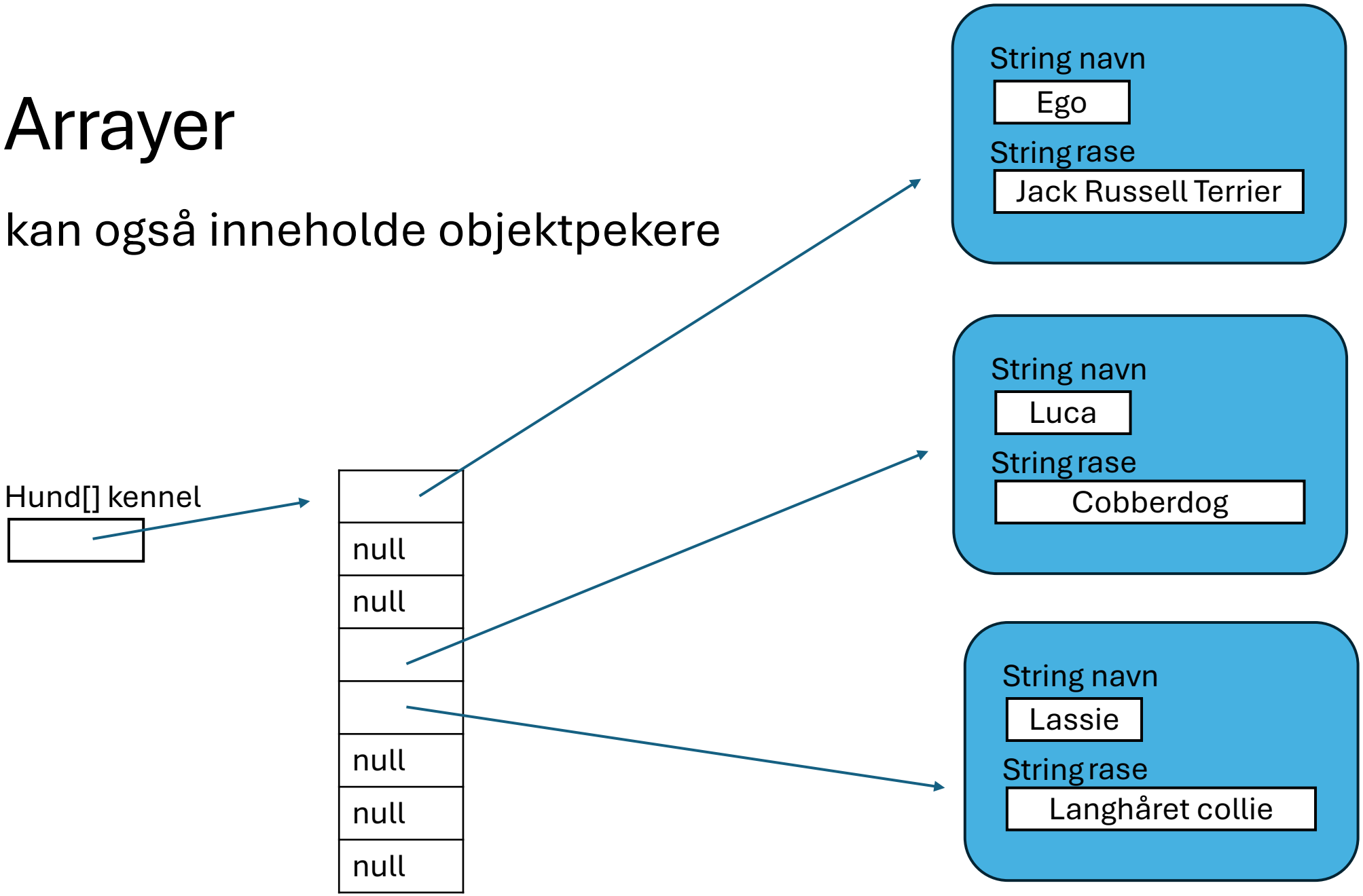
- En array er en sammenhengende rekke (i minnet) av variabler av samme type, som kan indekseres med et heltall
- Arrayer er også objekter




```
int[] noenTall = new int[8];
noenTall[0] = 42;
noenTall[1] = 0;
noenTall[2] = 3
noenTall[3] = 17;
...
```

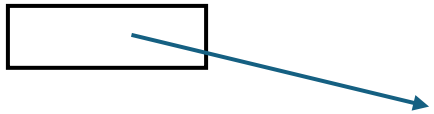
Arrayer

kan også inneholde objektpekere



Vi kan lage arrayer med 2 dimensjoner for å modellere f.eks. et rutenett

int[][] rutenett



0	1	2	3
4	5	6	7
8	9	10	11

```
int[][] rutenett = new int[3][4];
rutenett[0][0] = 0;
rutenett[0][1] = 1;
rutenett[0][2] = 2;
rutenett[0][3] = 3;
rutenett[1][0] = 4;
rutenett[1][1] = 5;
rutenett[1][2] = 6;
rutenett[1][3] = 7;
rutenett[2][0] = 8;
rutenett[2][1] = 9;
rutenett[2][2] = 10;
rutenett[2][3] = 11;
```

Størrelsen på rutenettet er ikke en del av typen, så det vil være lov å si f.eks.

```
rutenett = new int[5][2];
```

Metodekall og parameteroverføring

Vi starter med å modifisere Teller-klassen litt

```
class Teller {
   private int verdi;

   public void tellOpp( ) {
      verdi = verdi + 1;
   }
   public int hentVerdi( ) {
      return verdi;
   }
   public void nullstill( ) {
      verdi = 0;
   }
}
```



```
class Teller {
   private int verdi;

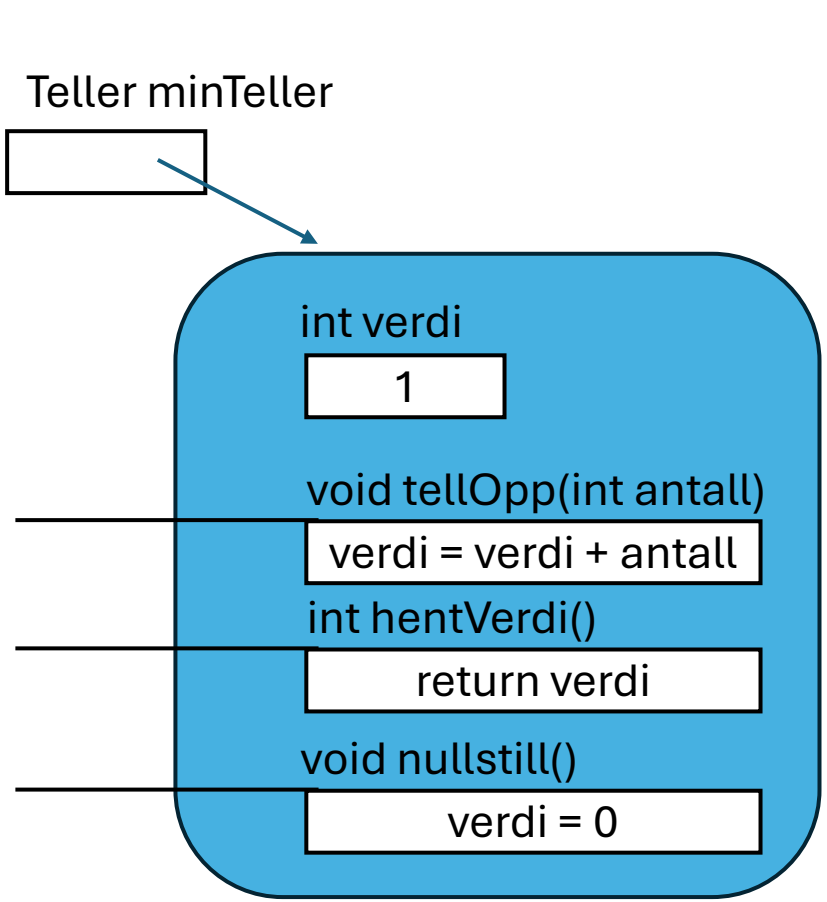
   public void tellOpp(int antall) {
      verdi = verdi + antal;
   }
   public int hentVerdi( ) {
      return verdi;
   }
   public void nullstill( ) {
      verdi = 0;
   }
}
```

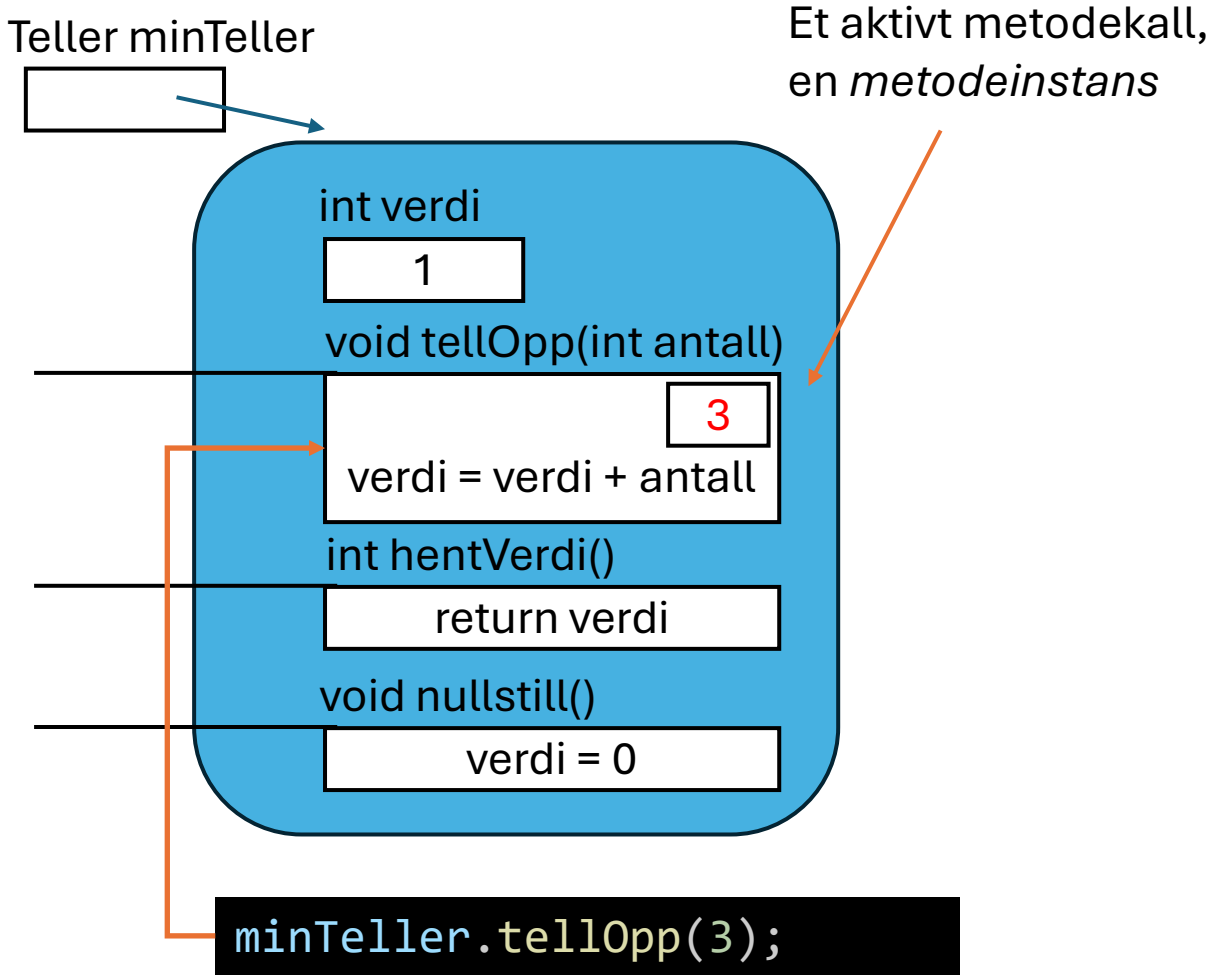
Formell parameter i deklarasjonen

```
Teller minTeller = new Teller();
minTeller.tellOpp(3);
```

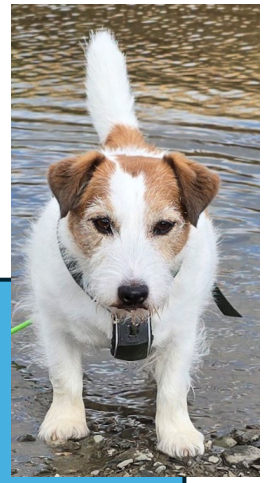
Aktuell parameter der metoden kalles

Når vi kaller en metode med en aktuell parameter, kopieres *verdien* inn i metodekallet





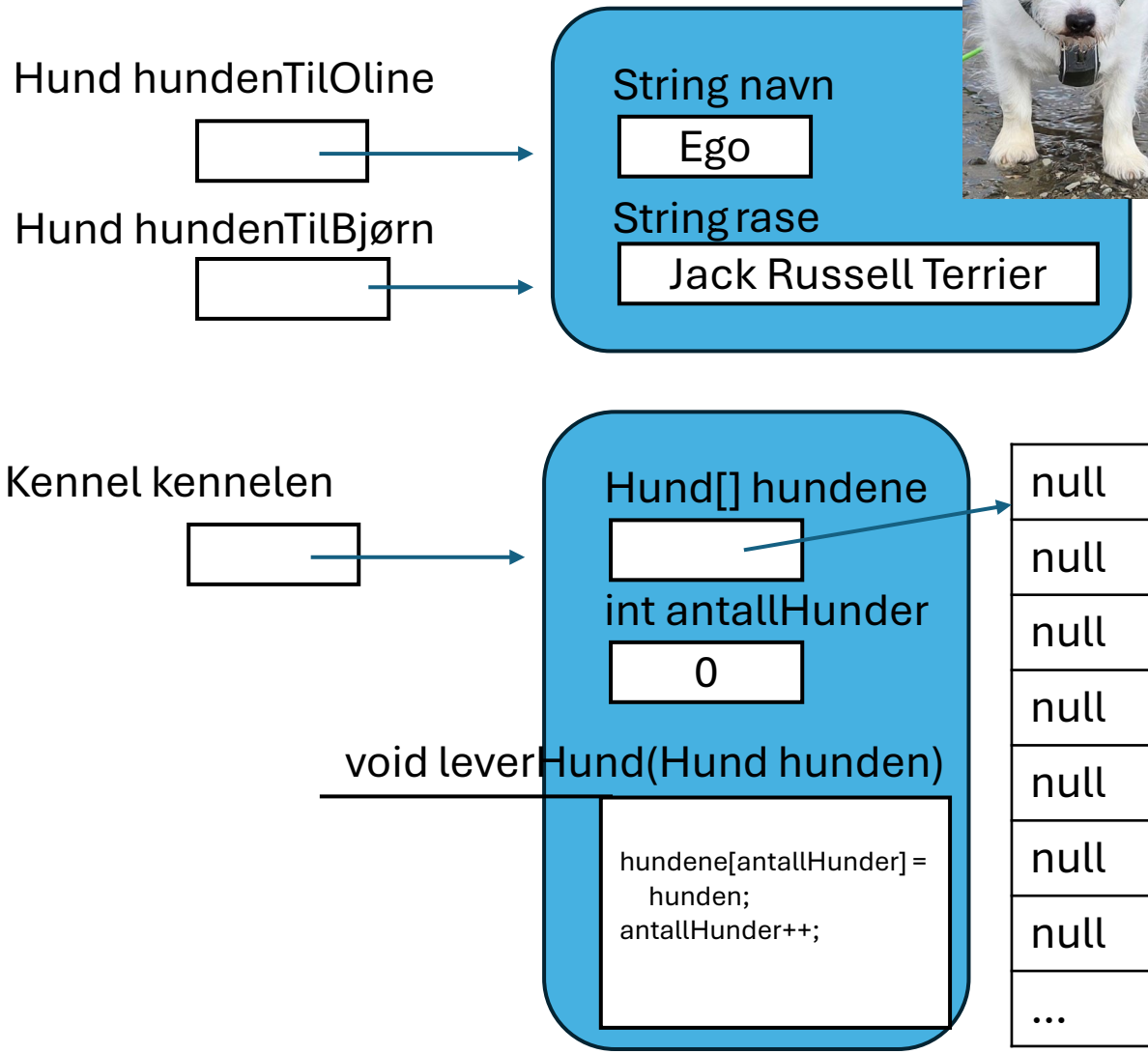
Parameteroverføring med objekter: verdien kopieres fortsatt



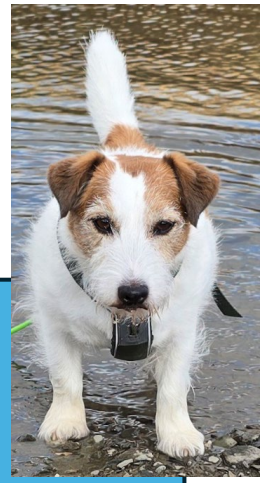
```
class Kennel {
    private Hund[] hundene = new Hund[10];
    private int antallHunder = 0;


    void leverHund(Hund hunden) {
        hundene[antallHunder] = hunden;
        antallHunder++;
    }
}
```

```
Kennel kennelen = new Kennel();
```



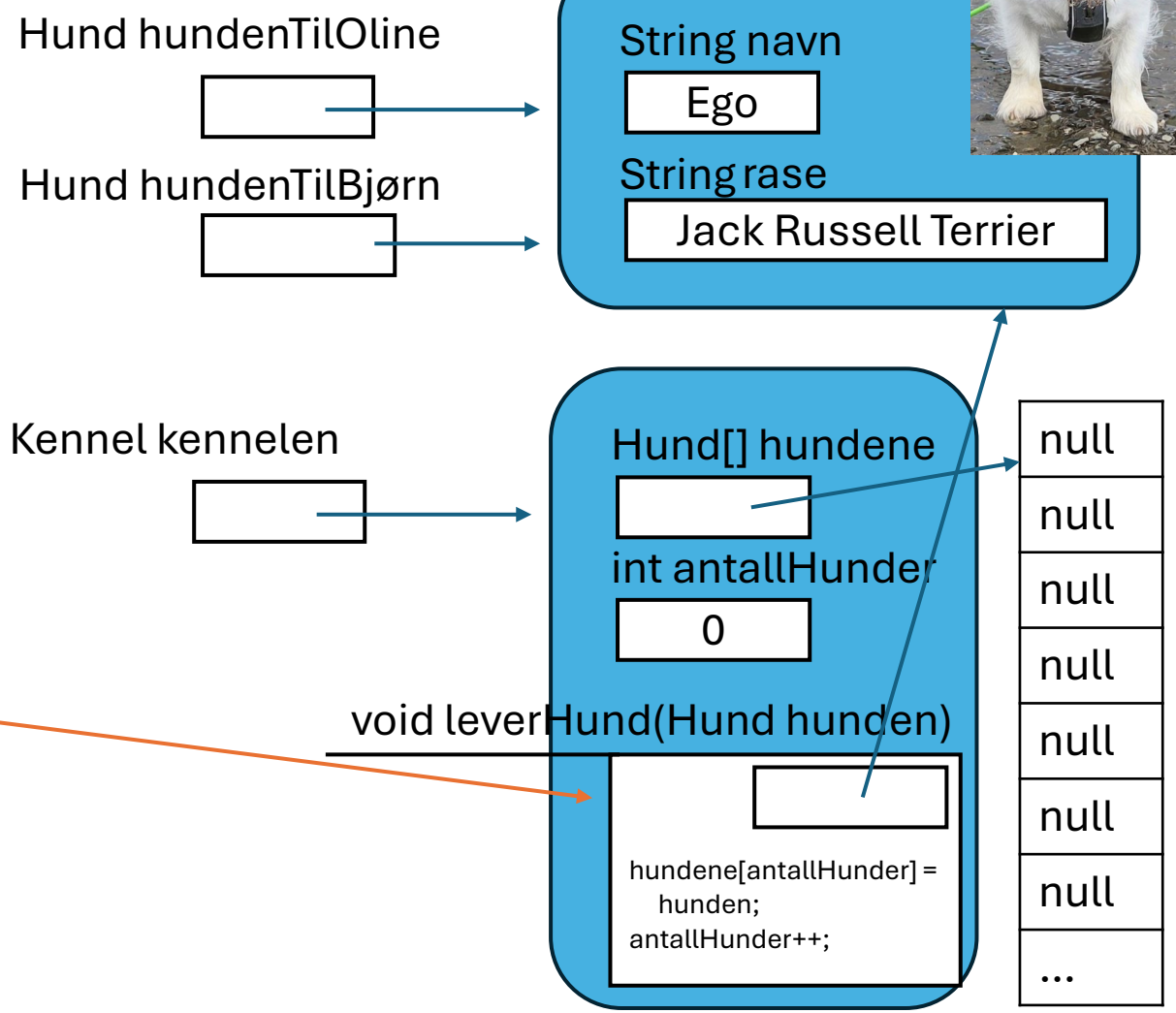
Parameteroverføring med objekter: verdien kopieres fortsatt



```
class Kennel {
    private Hund[] hundene = new Hund[10];
    private int antallHunder = 0;


    void leverHund(Hund hunden) {
        hundene[antallHunder] = hunden;
        antallHunder++;
    }
}
```

```
Kennel kennelen = new Kennel();
kennelen.leverHund(hundenTilOline);
```



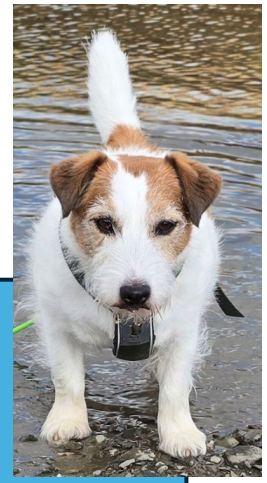
Parameteroverføring med objekter: verdien kopieres fortsatt

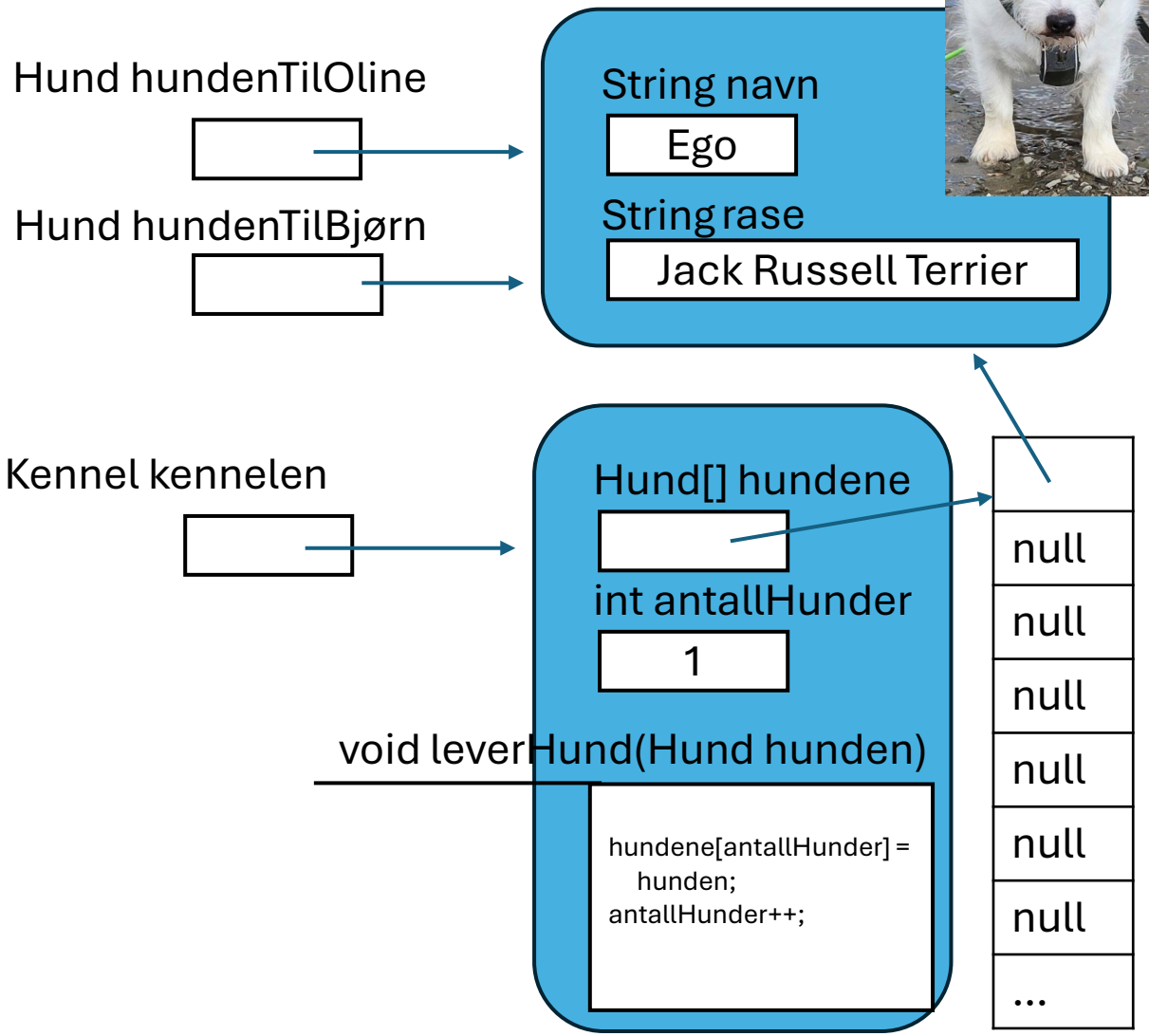
```
class Kennel {
    private Hund[] hundene = new Hund[10];
    private int antallHunder = 0;


    void leverHund(Hund hunden) {
        hundene[antallHunder] = hunden;
        antallHunder++;
    }
}
```

```
Kennel kennelen = new Kennel();
kennelen.leverHund(hundenTilOline);

// nå er vi her
```





I dag har vi lært ganske mye!

- Objekter inneholder data (variabler) og metoder (funksjoner/oppførsel)
- Det kan være nyttig å tegne objekter og pekere
- I Java har variabler og metoder typer som sjekkes av kompilatoren
- Det er forskjell på verdier og referanser – referanser peker på et objekt
- Når vi kaller en metode så kopieres den aktuelle parameteren



Neste gang: mer Java med Siri 🥰

Bilde fra https://pix4free.org



Husk: programmering er et *håndverk*.
Du blir ikke god til det uten å programmere mye selv.

Bilde fra https://unity.com