IN1010 - våren 2025

Torsdag 23. januar

- Del 1: Objektorientering og Simula
- Del 2: Verdier og variabler i Java
- Del 3: Objekter og klasser i Java
- Del 4: static i Java
- Del 5: Metoder og metodeinstanser i Java (hvis tid)

Stein Gjessing

Del 1: Objektorientering og Simula

Hva er en model (av noe)?
Og hva skal modellen brukes til ?

Modeller av biler, traktorer, . . .



Modeller av båter



Modeller av fly



Hva er hensikten med modellen?
Hvor nøyaktig skal vi modellere?


Shell
Kristen Nygaard og
Ole-Johan Dahl
skulle modellere . . .



Hvordan ser en bil ut inne
i datamaskinen ?

Hva kan en bil gjøre
inne i datamaskinen?



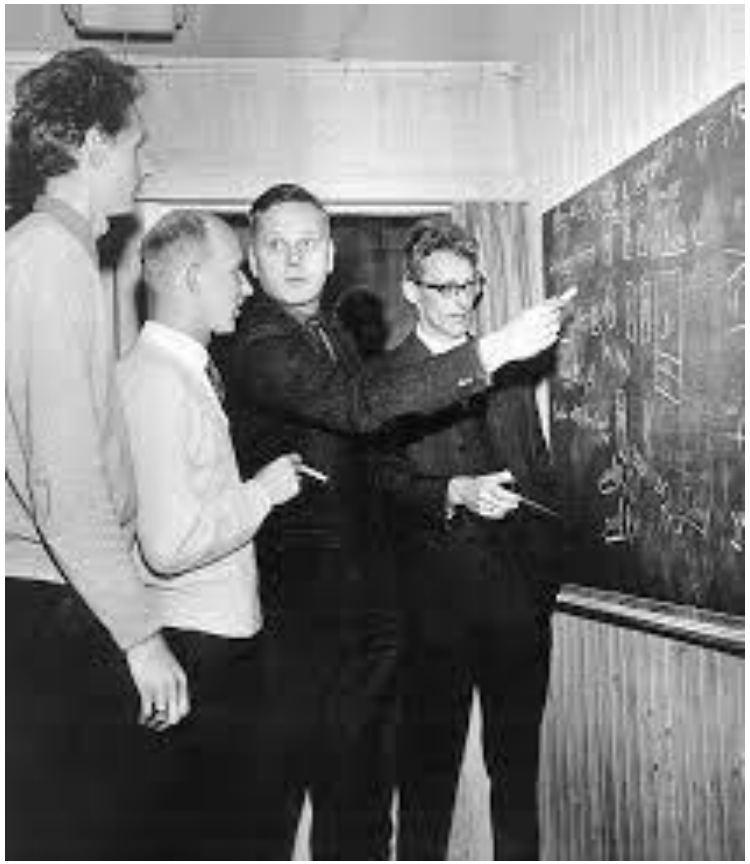
Hvordan ser en tømmerstokk eller
tømmerstabel ut inne
i datamaskinen ?

Hva kan en tømmerstokk gjøre inne
i datamaskinen?

Utvikling av programmeringsspråket Simula

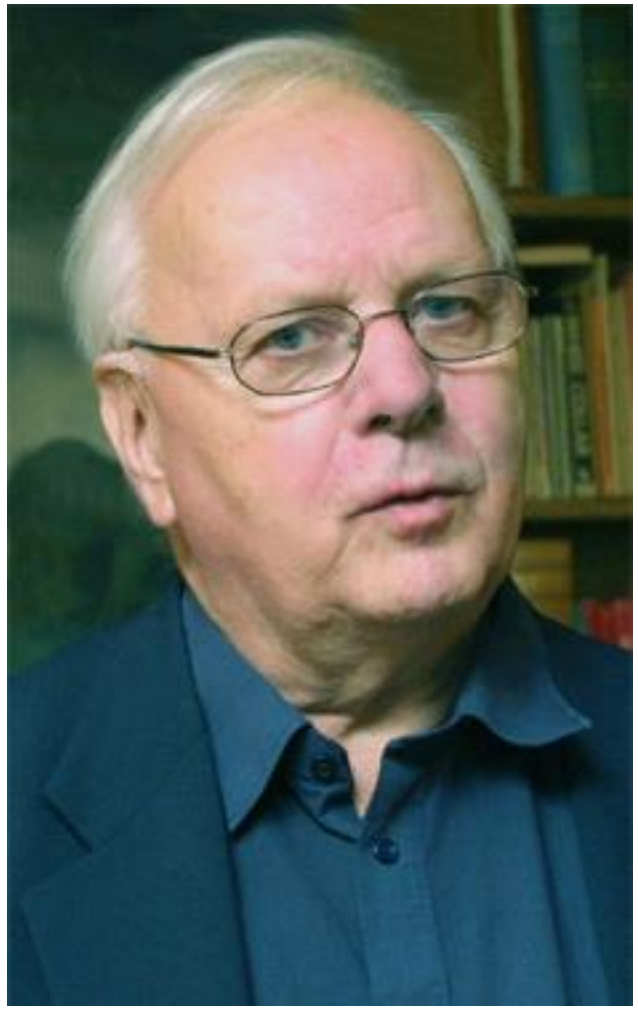
Kristen Nygaard og Ole-Johan Dahl fant på et eget programmeringsspråk til å lage og manipulere **objekter** inne i datamaskinen: **Simula**

Objektene skulle gjerne ha eget liv.



- Kristen (grønn (og rød)): Masse ideer
- Ole-Johan (blå): Men det må virke sammen, være konsistent. Derfor er det bare litt som kan brukes, og det må være på denne måten . . .
- Masse uenighet og konstruktiv "krangel" (ca 1965)

Oppfinnerene av ObjektOrientert programmering



Kristen Nygaard



Ole-Johan Dahl



Ole-Johan og Kristen fikk Turingprisen i 2002
(informatikkens Nobelpris)

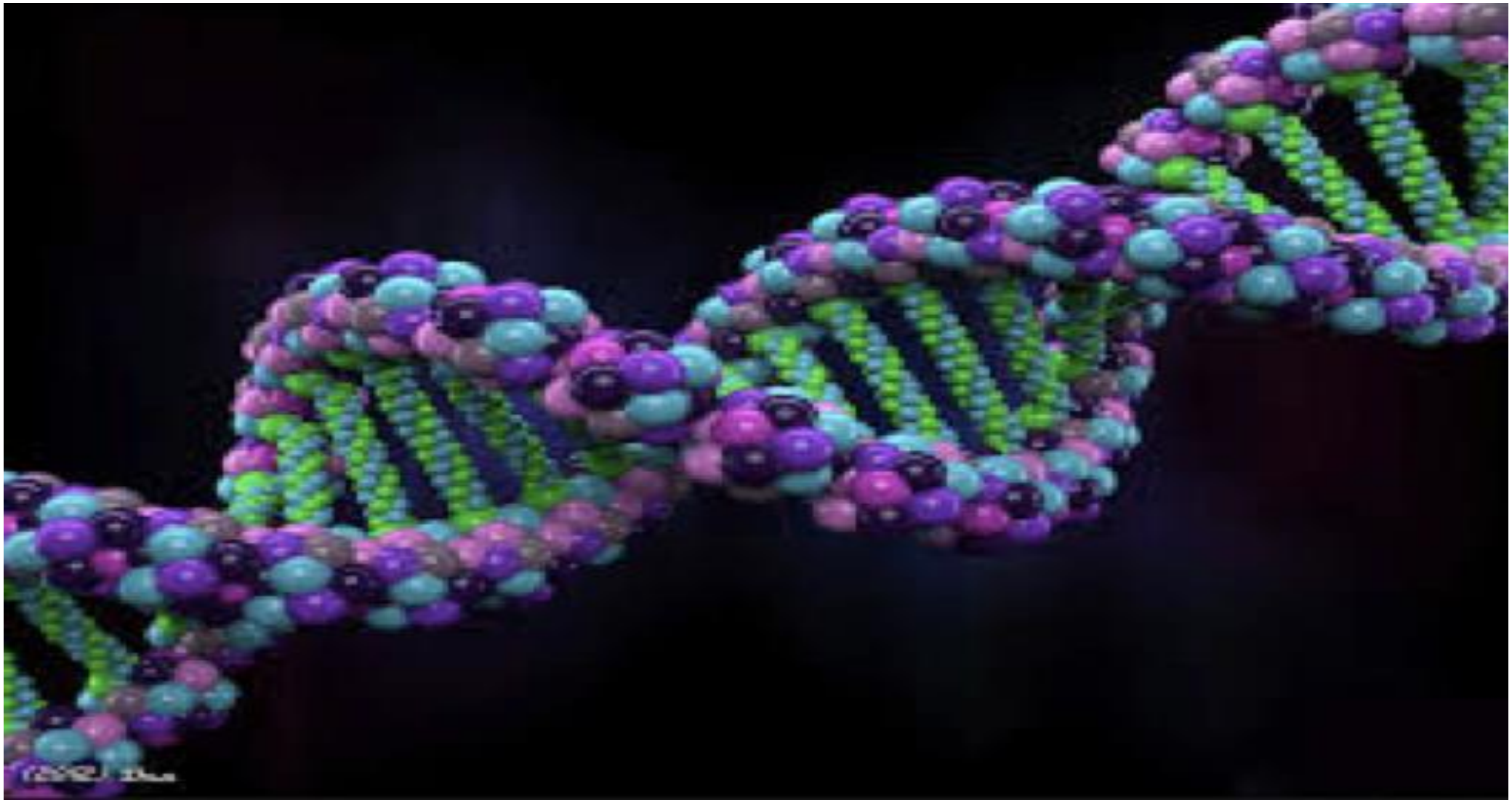


Jeg har ikke hatt et slikt fysisk objekt på mange år



Og slike objekter blir det stadig færre av

DNA kan modelleres inne i datamaskiner



Digital tvilling

- Et objekt i den virkelige verden modelleres og simuleres i datamaskinen

Modell

Simulering

Interaksjon

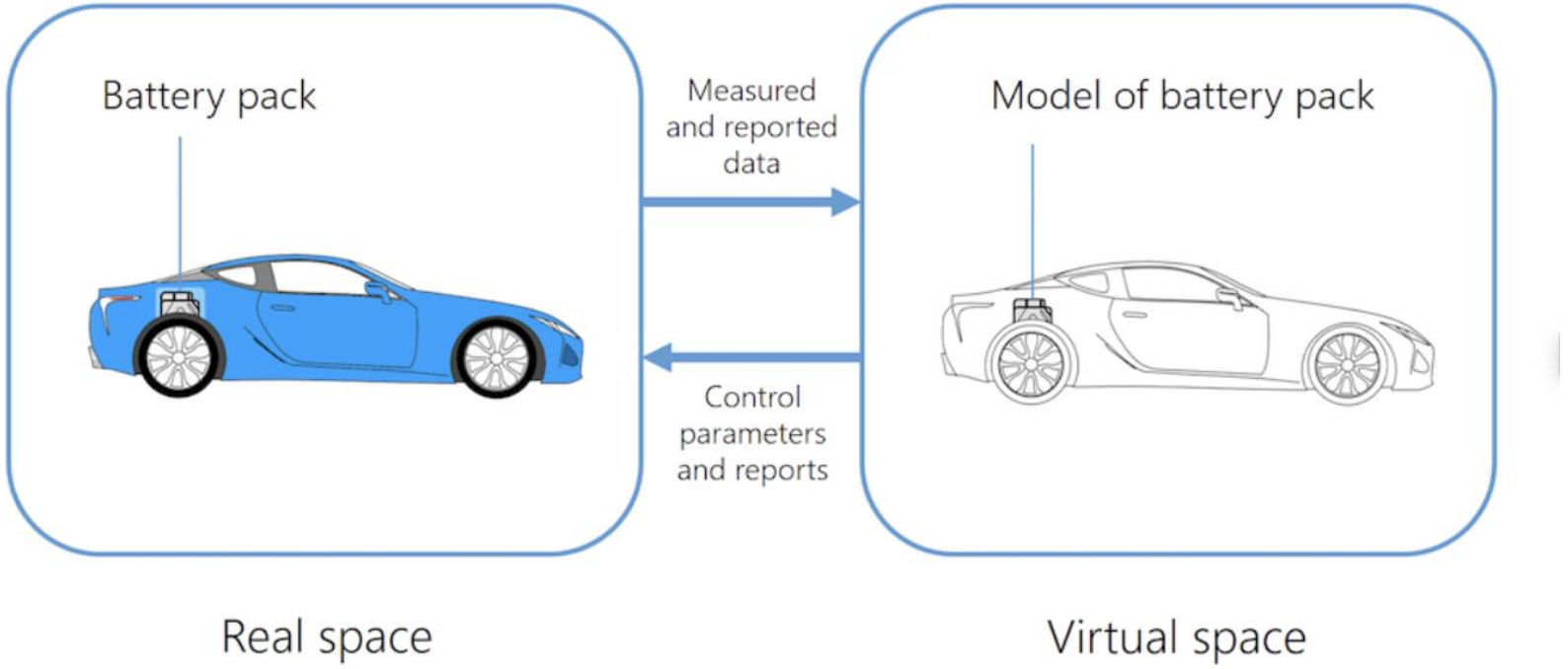


Fig: Comsol

Hvis vi ønsker å modellere en hund inne i datamaskinen.
Hva trenger vi da?

Informasjon/data:

- Vekt
- Alder - Fødselsår
- . . . . .

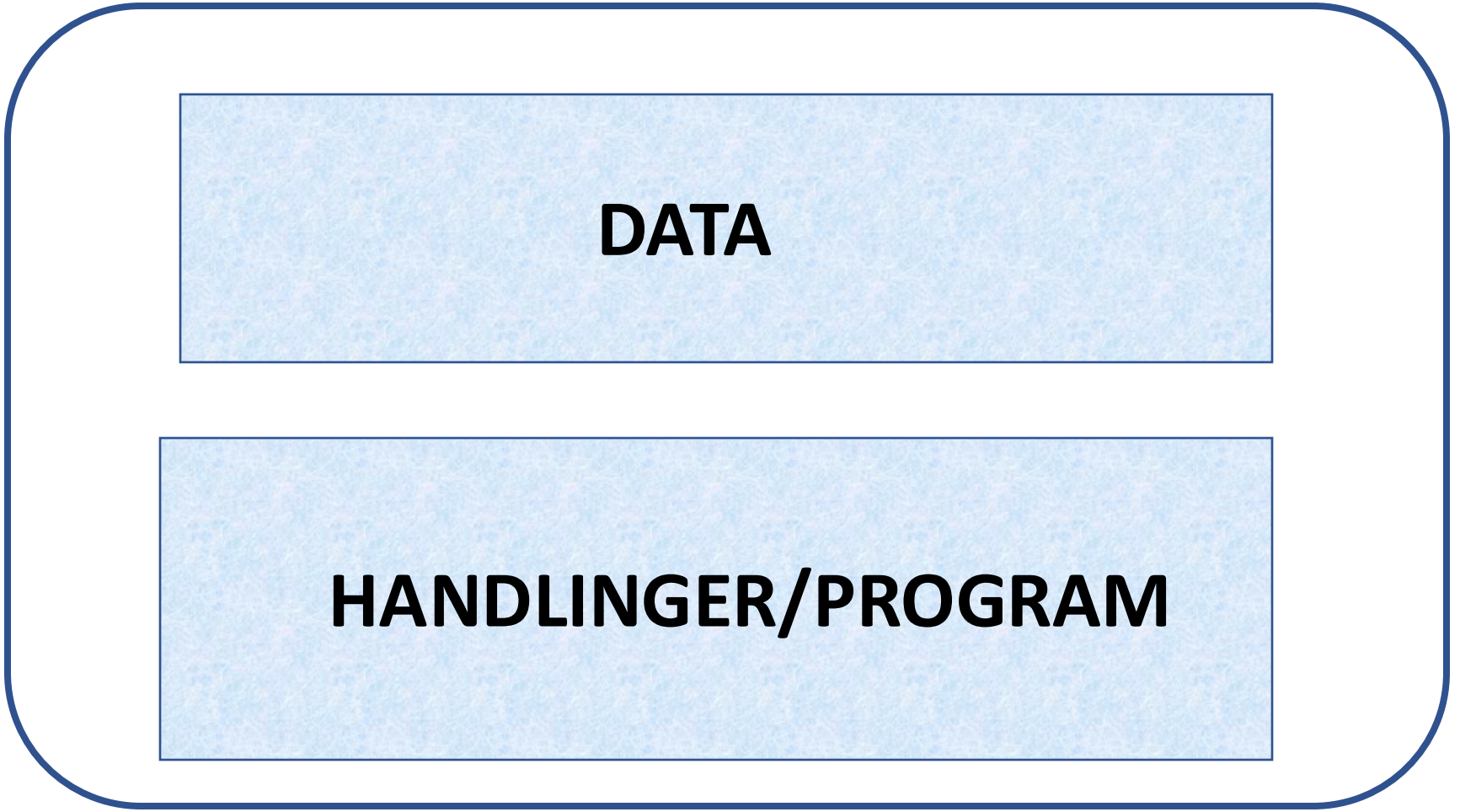
Oppførsel/handlinger:

- Spise
- Sove
- . . . . .

En hund

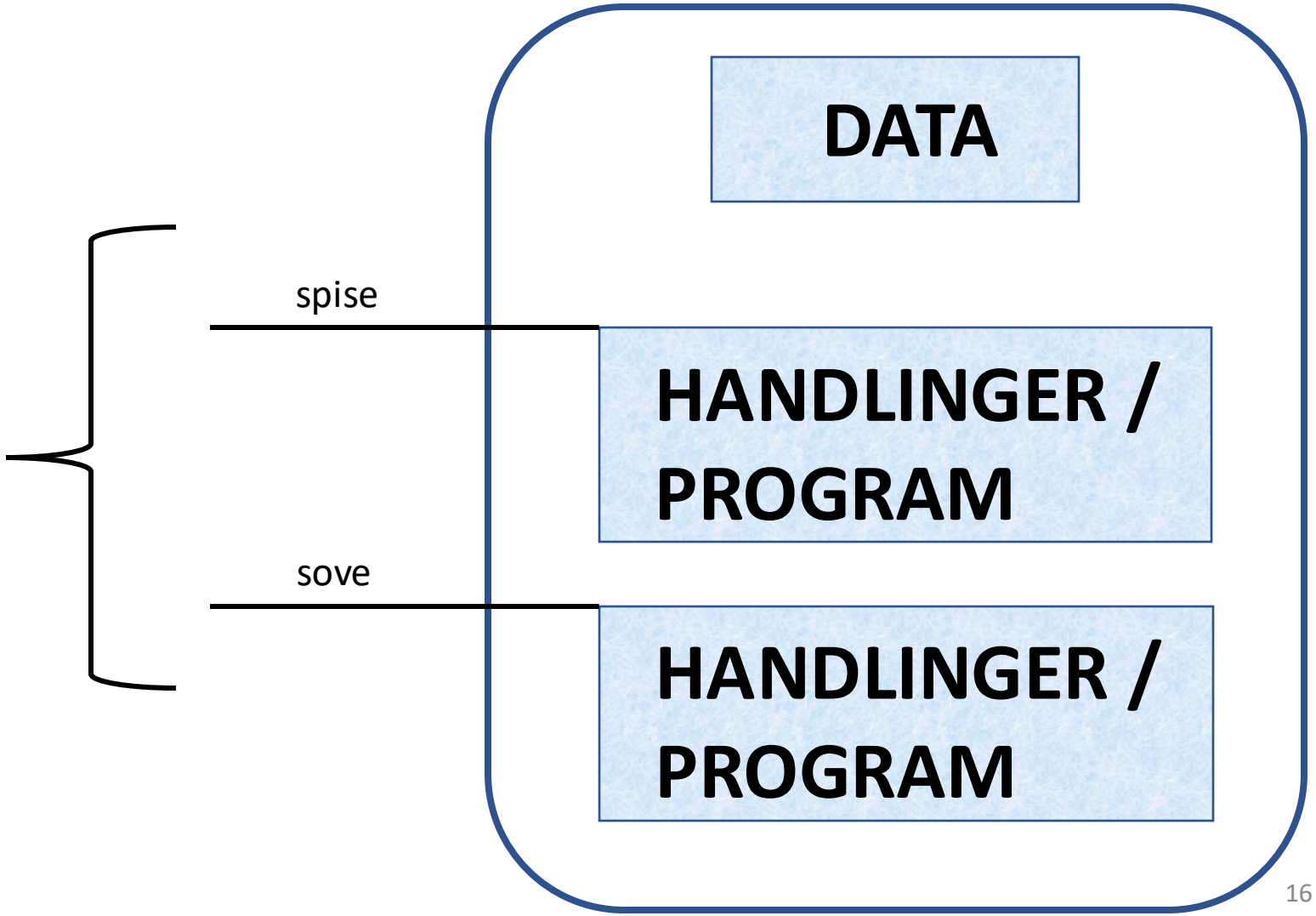


Slik tegner / tenker jeg meg et objekt:



Vi kan be et objekt om å utføre en handling

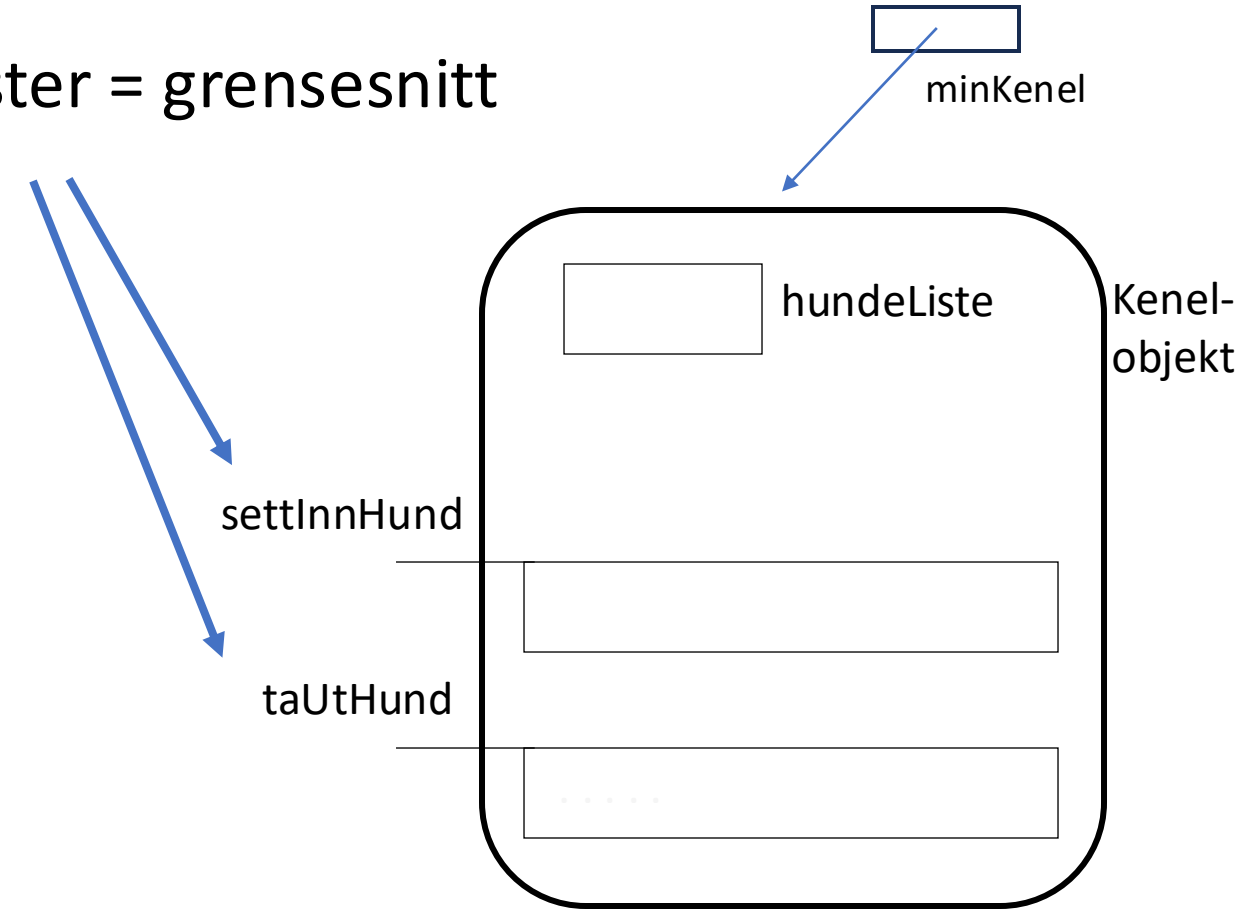
I IN1000 lærte dere om begrepet **objektets grensesnitt**



Et objekt kan tilby **tjenester** til resten av programmet

F. eks.:

Tjenester = grensesnitt



I Python:

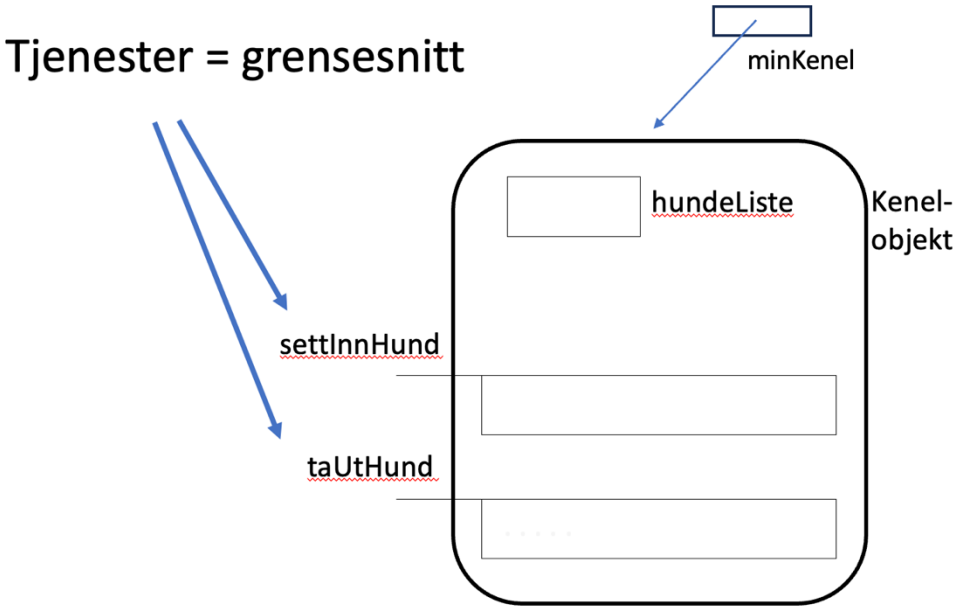
class Kenel . . . .

minKennel = Kenel



Sett dere sammen 2 - 3 – 4 – 5 i 5 minutter:

- Med den erfaringen dere har fra klasser og objekter i Python:
  - Hvordan **tenker dere** at et objekt ser ut?
  - Hvordan ville **dere tegnet** et objekt?
    - Eller er det ikke mulig å tegne et objekt?
  - Har et objekt i Python et navn?
    - (Har et objekt i Java et navn?)

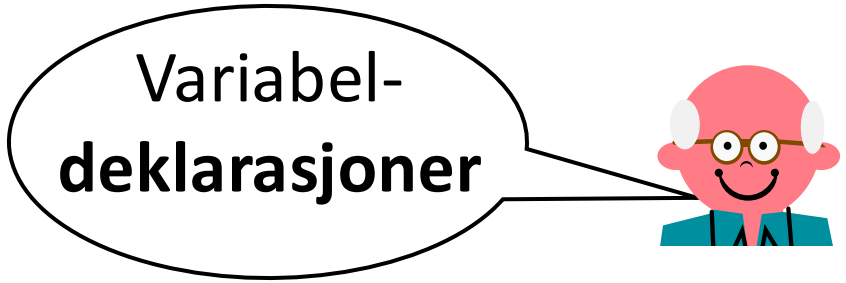


Del 2: Java: verdier, variabler og typer

Vi bygger på mandagens forelesning (20. jan)

Hva skjer med variabler når et Java-program blir utført?

Hva er en variabel (i Java) ?




```
int antall;
int flere;
antall = 6;
flere = 4;
antall = antall + flere;
```

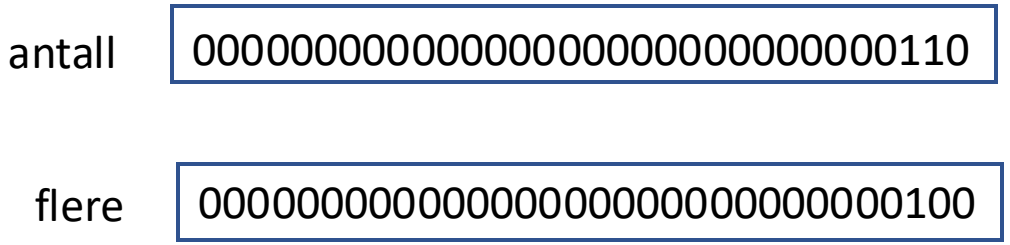
samme som

```
int antall = 6;
int flere = 4;
antall = antall + flere;
```

En variabel i Java er

- Noen bit/byte i minnet som kan innholde en verdi
- Det stedet i minnet der disse bittene ligger har et navn, f.eks. antall eller flere

```
int antall = 6;
int flere = 4;
```



Statisk typing betyr at disse variablene alltid vil inneholde et heltall (er sjekket av kompilatoren)

.

RAM
Primærlager
Minne

Minnet adresseres
i byte
(1 byte = 8 bit)

		0
		46579598
		46579599
		46579600
		46579601
		46579602
		46579603
	11100100	46579604
flere	00000000	46579605
	00000000	46579606
	00000000	46579607
	00000100	46579608
	00111101	46579609
	11100100	46579610
	00100100	46579311
data		
		53485321
	00110100	53485322
	00000100	53485323
	11000100	53485324
antall	00000000	53485325
	00000000	53485326
	00000000	53485327
	00000110	53485328
	10110100	53485329
		53485330
	10101110	93685398
	10101110	
	10101111	
program	00111110	
	10101110	
		f.eks. 16000000000 ca (16 GB)

Minnebuss

Prosessor
f.eks. Intel i7

Registre

Aritmetisk-
Logisk
Enhet
(ALU)

4362738 Instruksjonsregister

93685398 Programteller

Disk/
Sekundærlager/
SSD

I/O buss

Filer
F.eks. 512 GB

UNIVERSITETET I OSLO

Institutt for informatikk

Vennen din spør deg:

Du som studerer informatikk må vel vite om dette er gode spesifikasjoner.
Forklar meg hva dette betyr.



Kort om produktet

Asus Vivobook 15 X1504 er designet for å imøtekomme alle dine produktivitets- og underholdningsbehov med kraftige komponenter og en fantastisk skjerm. Hev dataopplevelsen din, enten du er profesjonell på farten, student eller hverdagsbruker.

Les mer om produktet

Spesifikasjoner

Intel® Core™ i7-1255U-prosessor

15,6" FullHD IPS-skjerm

16GB DDR4 RAM, 512GM M.2 PCIe SSD

Se alle spesifikasjoner

Spesifikasjoner

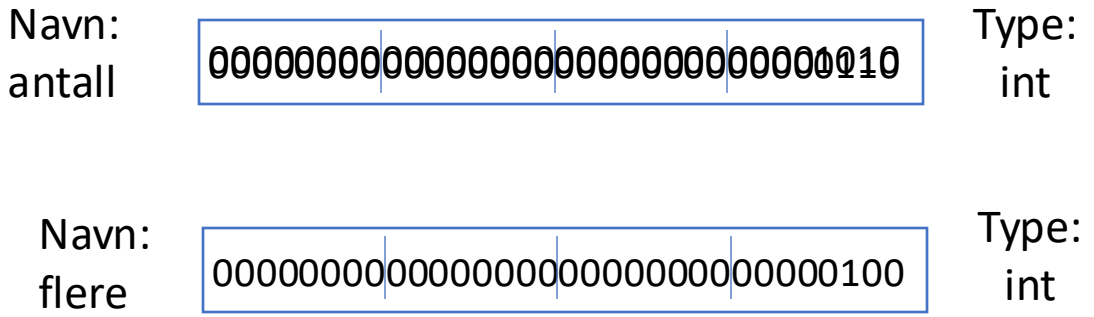
Intel® Core™ i7-1255U-prosessor

15,6" FullHD IPS-skjerm

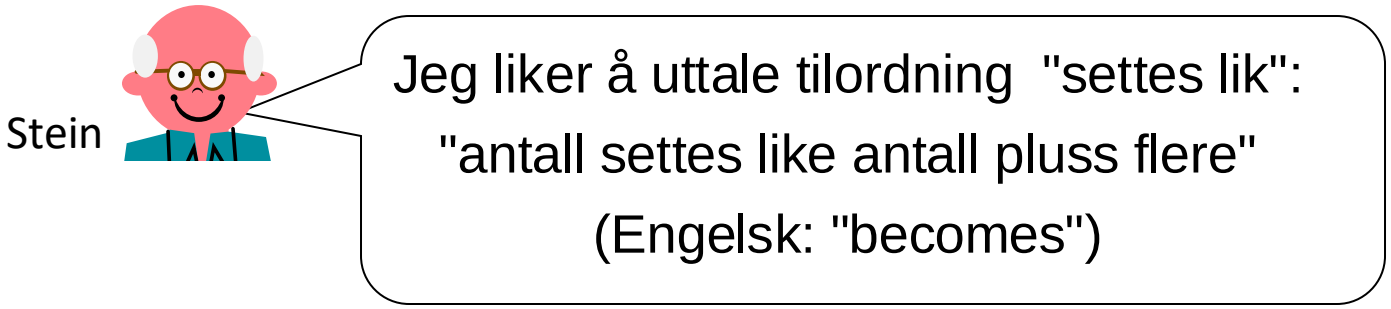
16GB DDR4 RAM, 512GM M.2 PCIe SSD

Variabler i Java


```
int antall = 6;
int flere = 4;
antall = antall + flere;
```



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 heller ikke navnet kan forandres).
Innholdet til en variabel forandres ved **tilordning.**



Javas verdier

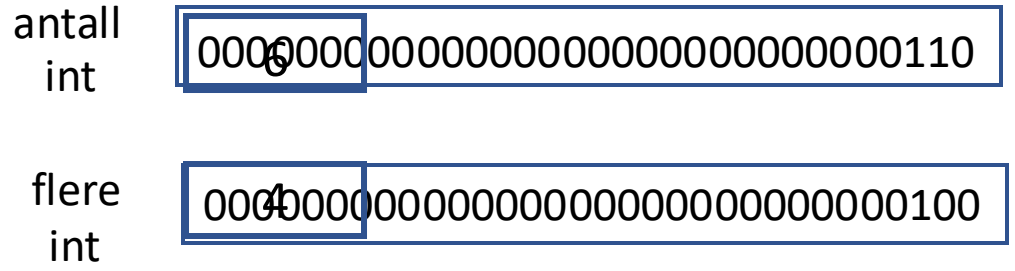
- 8 **primitive** typer: boolean, byte, char, short, int, long, float, double.

boolean	~~1~~ 00000001	8 bit	(Det er for upraktisk å lagre (sammen med andre data), pakke ut og inn og teste ett bit)
byte	01001110	8 bit	
char	0100111010110110	16 bit	
short	0100100101101110	16 bit	
int	01010010110110101000100101101110	32 bit	
long	0101001011010011010110100010101001010110100110101000100101101110	64 bit	
float	01010010110110101000100101101110	32 bit	IEEE standard 754
double	0101001011010011010110100010101001010110100110101000100101101110	64 bit	

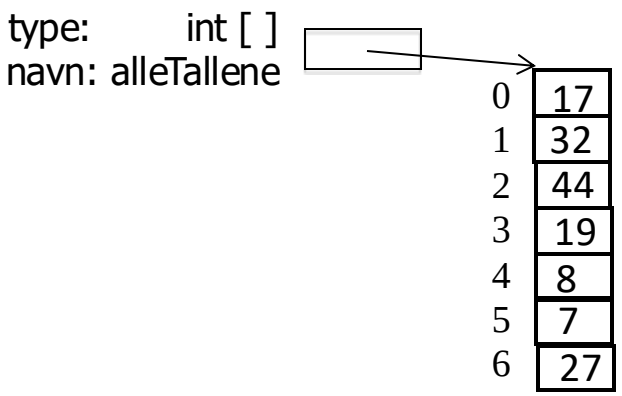
Det er disse verdiene som kan ligge i en primitiv variabel

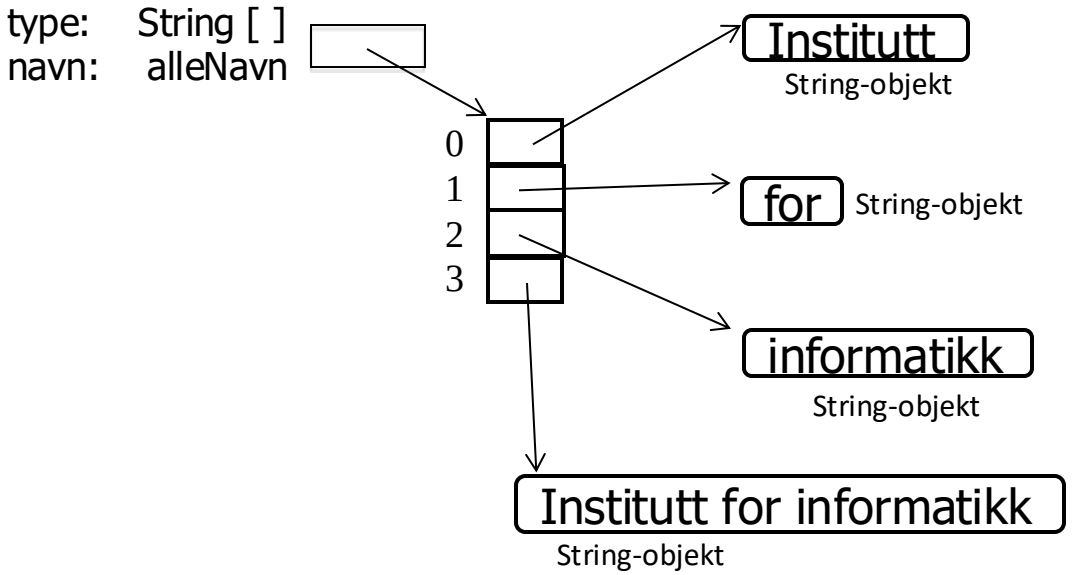
Slik tenker vi på og tegner en variabel i Java

```
int antall = 6;
int flere = 4;
```



Sammenhengende variabler er en **array**

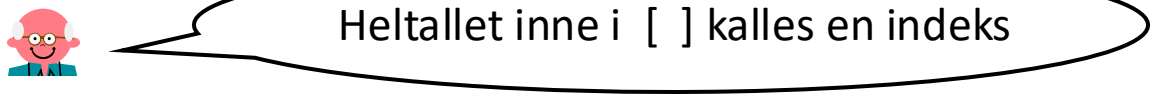




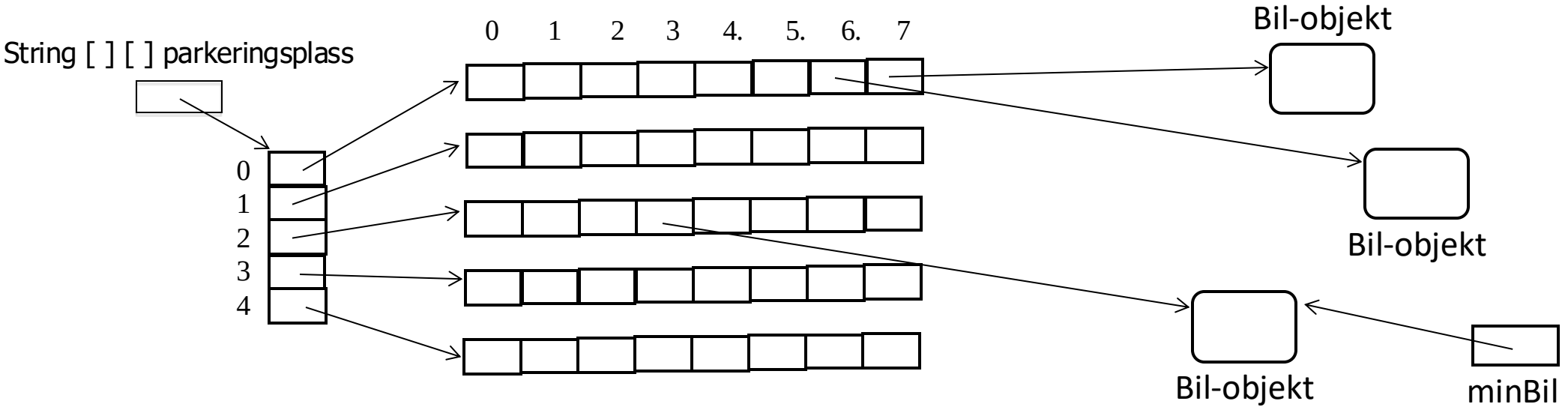

```
int [ ] alleTallene;
alleTallene = new int[7];
// Men innholdet nå er bare 0-er
alleTallene[0] = 17;
alleTallene[1] = 32;
// . . . .
```

```
String [ ] alleNavn;
alleNavn = new String[4];
// Men innholdet nå er bare "null"-er
```

```
alleNavn[0] = "Institutt";
alleNavn[1] = "for";
alleNavn[2] = "informatikk";
alleNavn[3] = alleNavn[0] + " " + alleNavn[1] +
              " " + alleNavn[2];
```



To-dimensjonale **array-er**



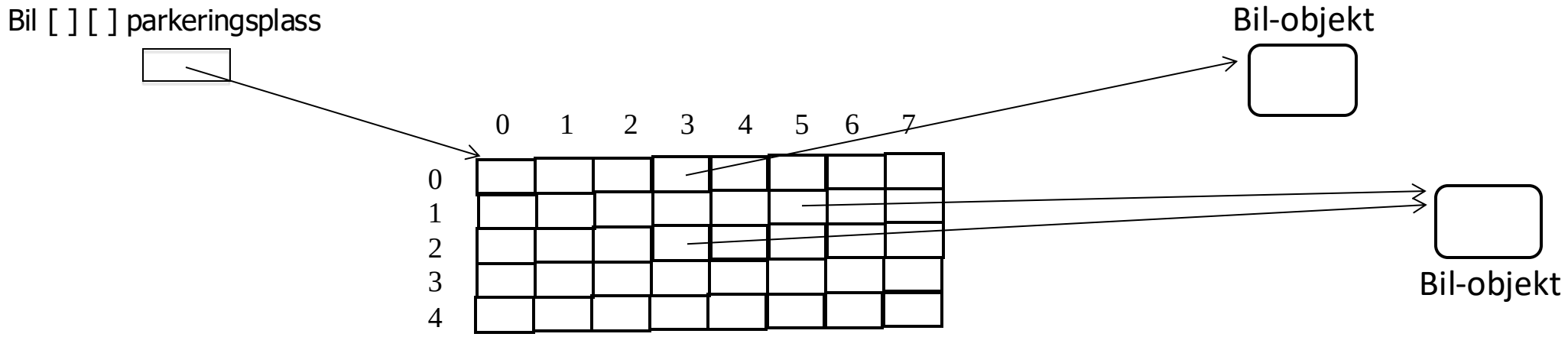

```
Bil [ ] [ ] parkeringsplass;
parkeringsplass = new Bil[5] [8];
// Men innholdet nå er bare "null"-er
```

```
parkeringsplass [0] [6] = new Bil();
parkeringsplass [0] [7] = new Bil();
Bil minBil = new Bil();
int radNr = 1;
int kolNr = 4;
parkeringsplass [radNr+1] [kolNr-1] = minBil;
```

radNr 1

kolNr 4

Alternativ (og enklere) sannhet ☺
(når alle radene er like lange)

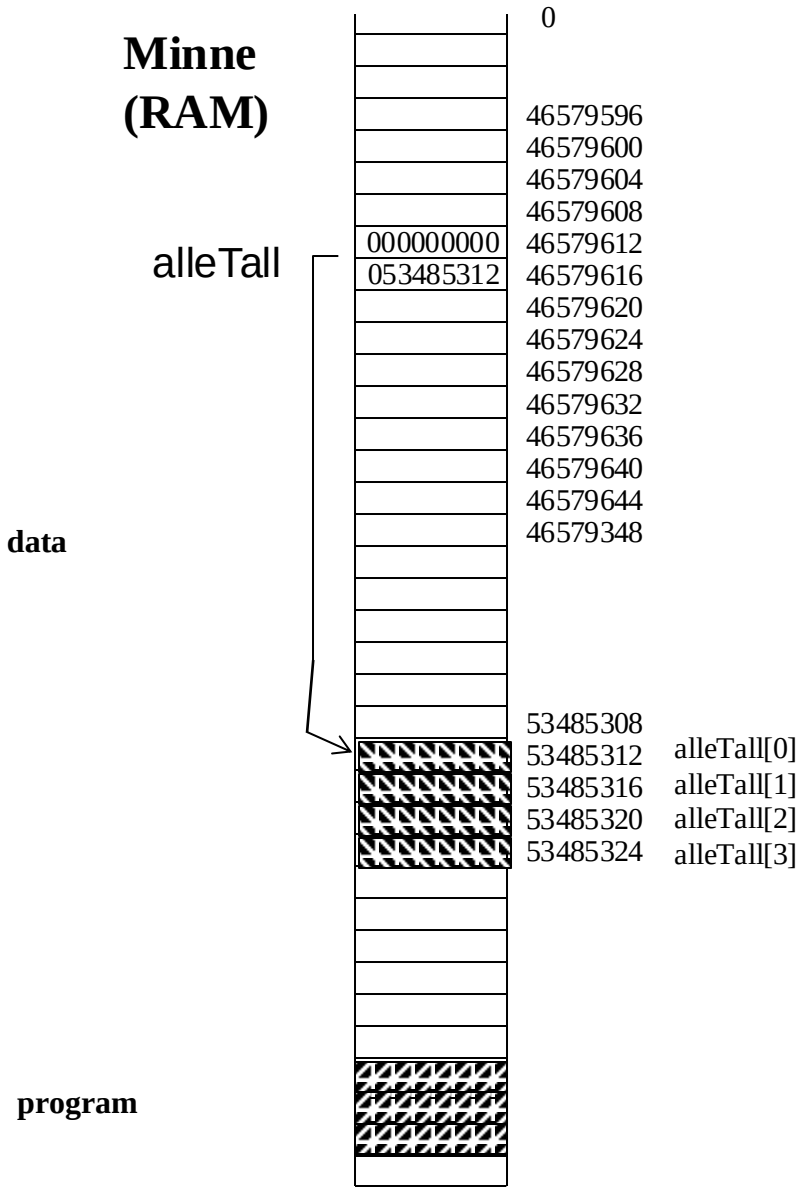


Indekser er alltid et uttrykk

```
Bil [ ] [ ] parkeringsplass;
int antRader = 5
int antKolonner = 8
parkeringsplass = new Bil[antRader] [antKolonner];
```

```
// Flere (tilfeldige) eksempler for å vise at indekser er uttrykk
int radNr = 6;
int kolNr = 2;
parkeringsplass [radNr/2-3] [kolNr+1] = new Bil();
parkeringsplass [1] [1+kolNr*2] = new Bil();
radNr = 1;
parkeringsplass [3-radNr] [kolNr+1] = parkeringsplass [radNr] [3+kolNr];
```

Detaljer i RAM (men ikke veldig viktig i IN1010)




```
Int[] alleTall = new int[4];
```

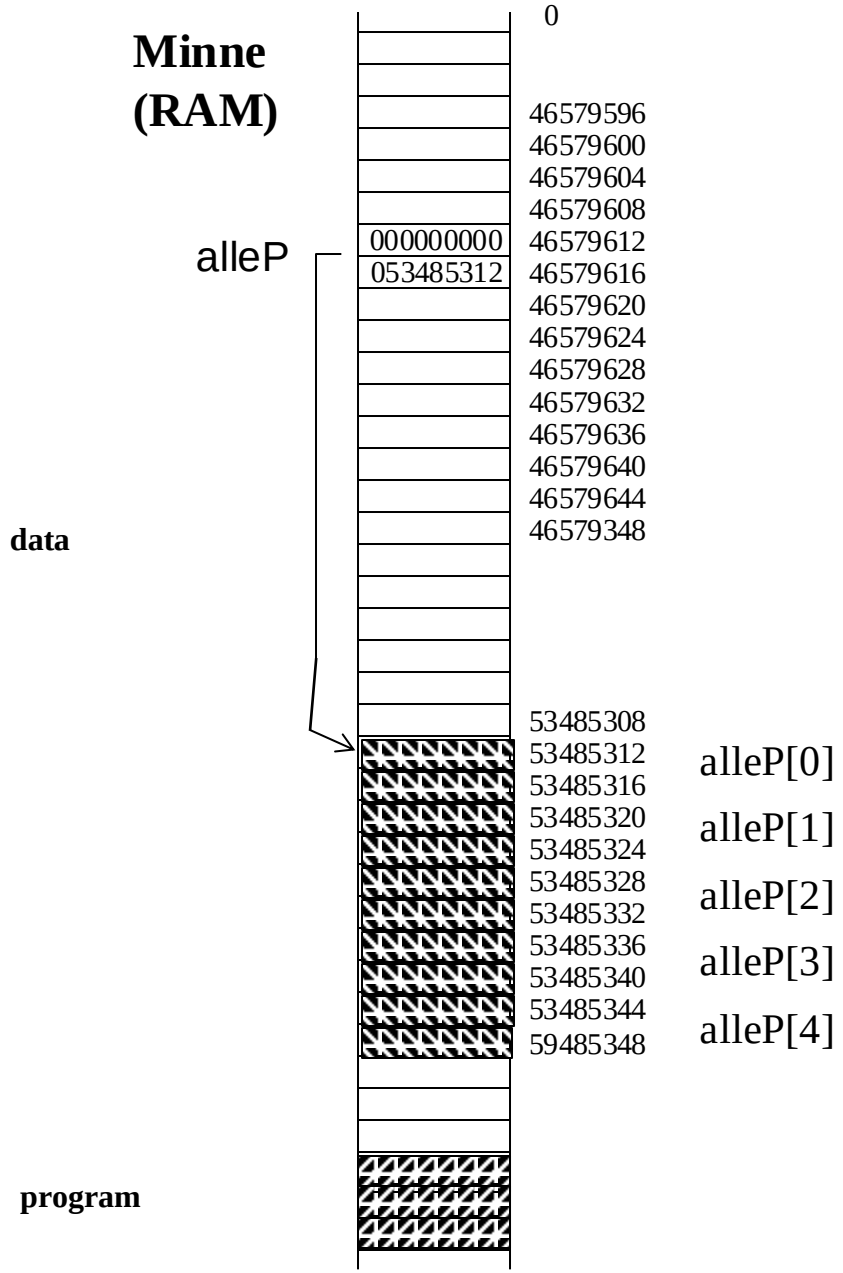
Array er iboende i datamaskinen og i Java

Oppslag (indeksering) i en array er veldig fort gjort av datamaskinen

Husk: Minnet er byte-adresserbart

Her har jeg tegnet 4 byte = 32 bit på hver linje og alleTall er en array av heltall, int (á 32 bit)

Detaljer i RAM (men ikke veldig viktig i IN1010)



String [] alleP = new String[5];

Array er iboende i datamaskinen og i Java

Oppslag (indeksering) i en array er veldig fort gjort av datamaskinen

Husk: Minnet er byte-adresserbart

Her har jeg tegnet 4 byte = 32 bit på hver linje og alleP er en array av referanser (á 64 bit) til String-objekter

UNIVERSITETET I OSLO

Todimensjonal array

Eksempel

```java
class ToDimArray {
  public static void main(String[] args) {
    int antRader = 4;    int antKol = 4;
    int[][] minToTab = new int[antRader][antKol];
    int tall = 1;
    for (int i = 0; i < antRader; i++) {
      for (int j = 0; j < antKol; j++) {
        minToTab[i][j] = tall;
        tall++;
      }
    }
    System.out.println("Arrayen ser slik ut: ");
    for (int i = 0; i < antRader; i++) {
      for (int j = 0; j < antKol; j++) {
        System.out.print(minToTab[i][j] + " ");
      }
      System.out.println();
    }
  }
}
```

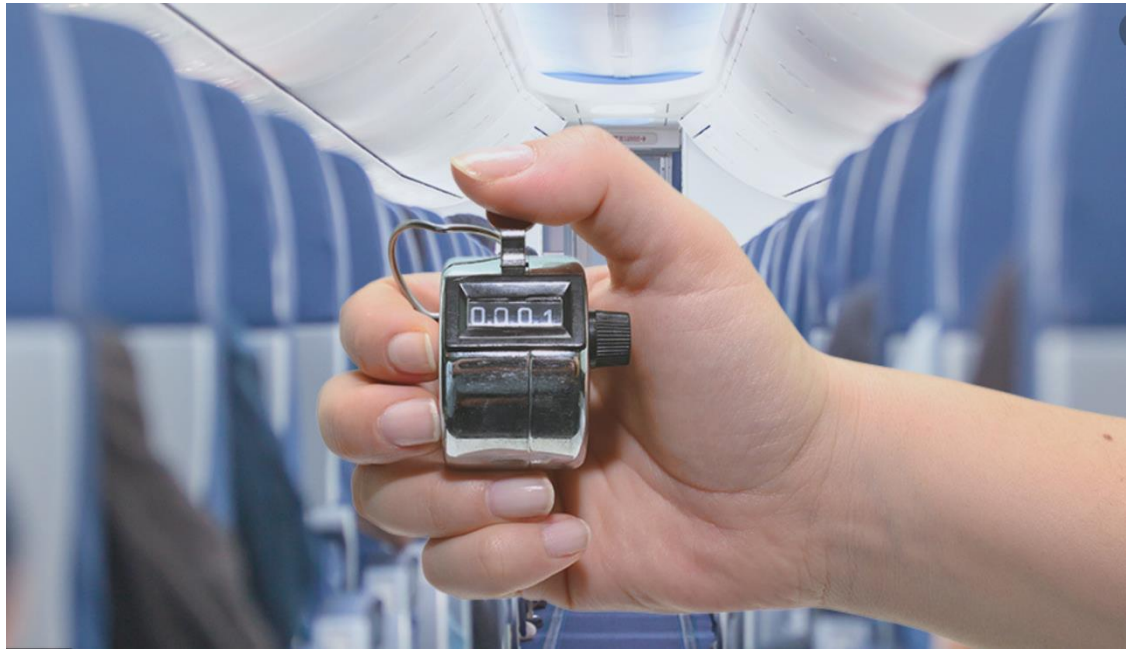
Del 3:

Objektorientert programmering i Java

- Hva er et objekt ?
- Hva er en klasse ?

Objekter i Java: En teller (engelsk: counter)

- Aller enkleste eksempel (Horstmann kap 8.2):
  - En teller (som f.eks. betjeningen på et fly bruker)
    - Tell én opp
    - Les av telleren
  - Starter på null
- Vi skal programmere en slik



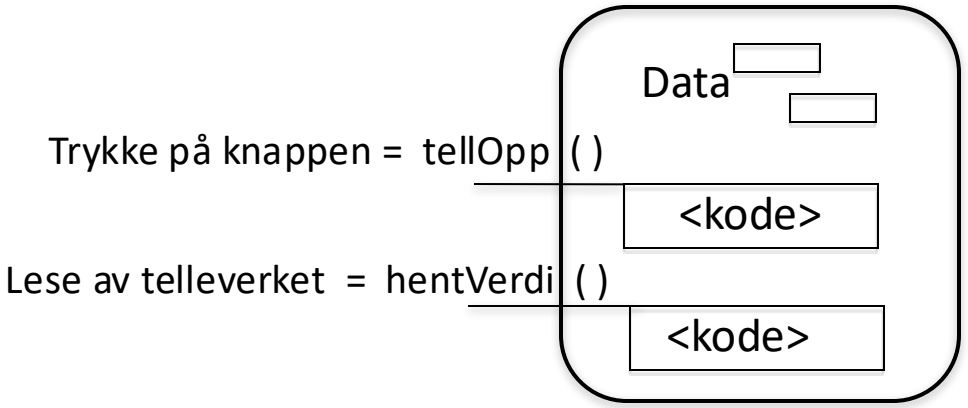
Objekter i den virkelige verden modelleres av objekter inne i datamaskinen

Et objekt tilbyr tjenester til omverdenen

For omverdenen er et objekt er en “sort boks” + grensesnittet mot omverdenen

Når vi programmerer objektet (klassen) er vi inne i objektet og ser alle detaljene

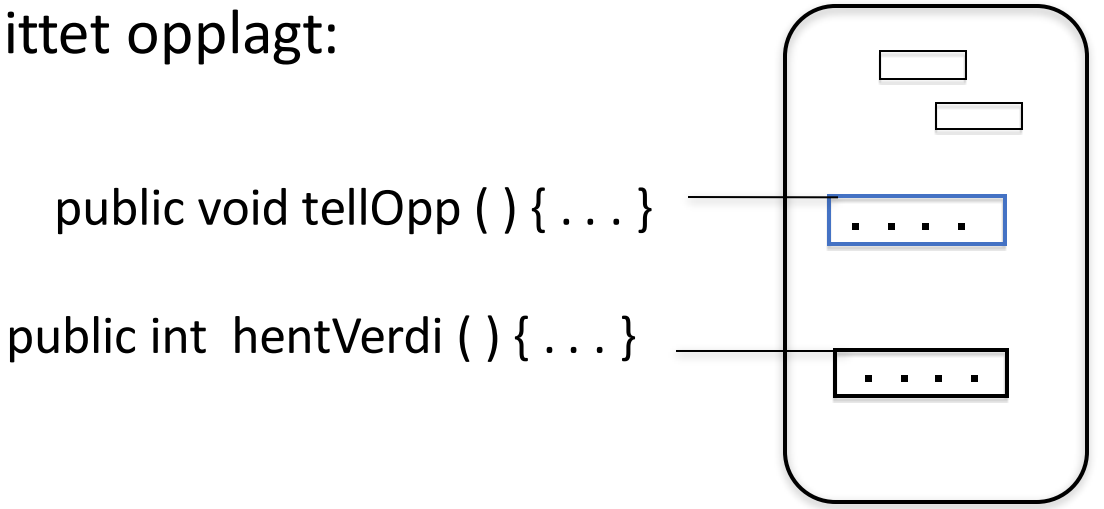
Et objekt er en samling av data og handlinger (kode)



Data er variabler
Koden er i metoder

En teller

- Hvordan skal tellOpp() og hentVerdi() virke
  - Senere i IN1010 blir det fort mer kompliserte grensesnitt
    - Og vi skal da snakke mer om virkemåten (semantikken) til et grensesnitt.
  - Men her er (skrivemåten til) grensesnittet opplagt:

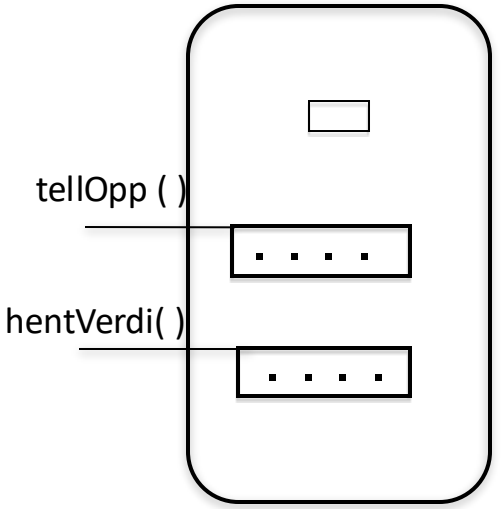


- Hvordan skal **metodene** tellOpp() og hentVerdi() programmeres OG
- Hvilke private **data** må være inne i et slikt teller-objekt?

Da må vi se for oss et teller-objekt

- Er det nok med èn variabel inne i objektet?

- Er det nok at
- tellOpp() teller opp denne verdien med èn og at
- hentVerdi() leser av denne verdien ?

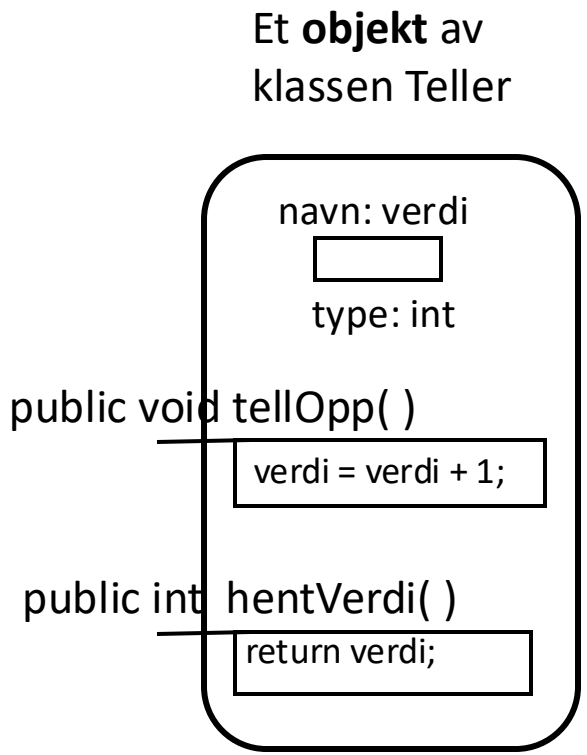




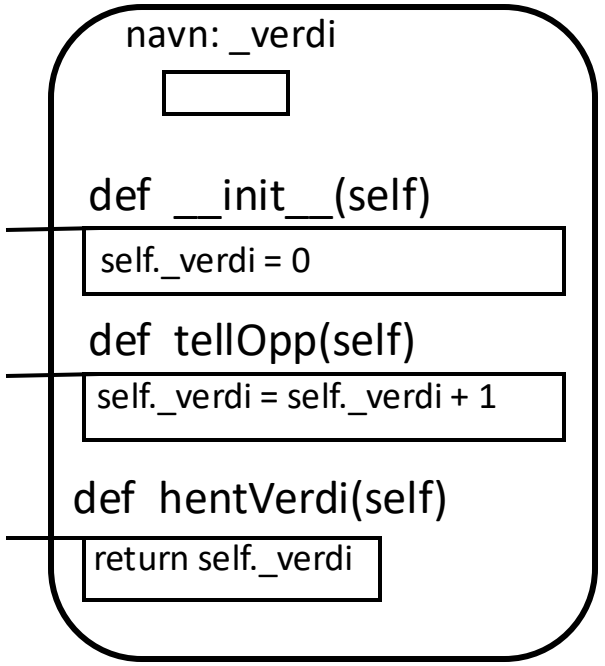
Det er dette som er å programmere !!
Men vanligvis er problemet vanskeligere !!

Et forslag til et teller-objekt

- Generelt innholder objekter
  - **Metoder – tjenester – handlinger**
    - public (som regel - grensesnittet)
    - men også private metoder
      - til bruk inne i objektet
  - **Variabler og konstanter - "DATA"**
    - av de primitive typene eller referanser
    - som regel skjult for omverdenen – private (innkapsling)



Python



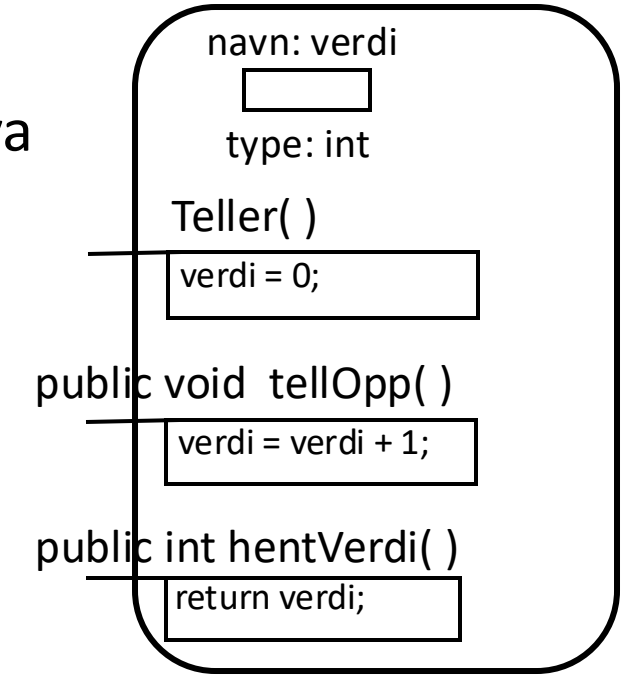

```
class Teller:

   def __init__(self):
      self. _verdi = 0

   def tellOpp(self):
       self._verdi = self. _verdi + 1

   def hentVerdi(self):
       return self._verdi
```

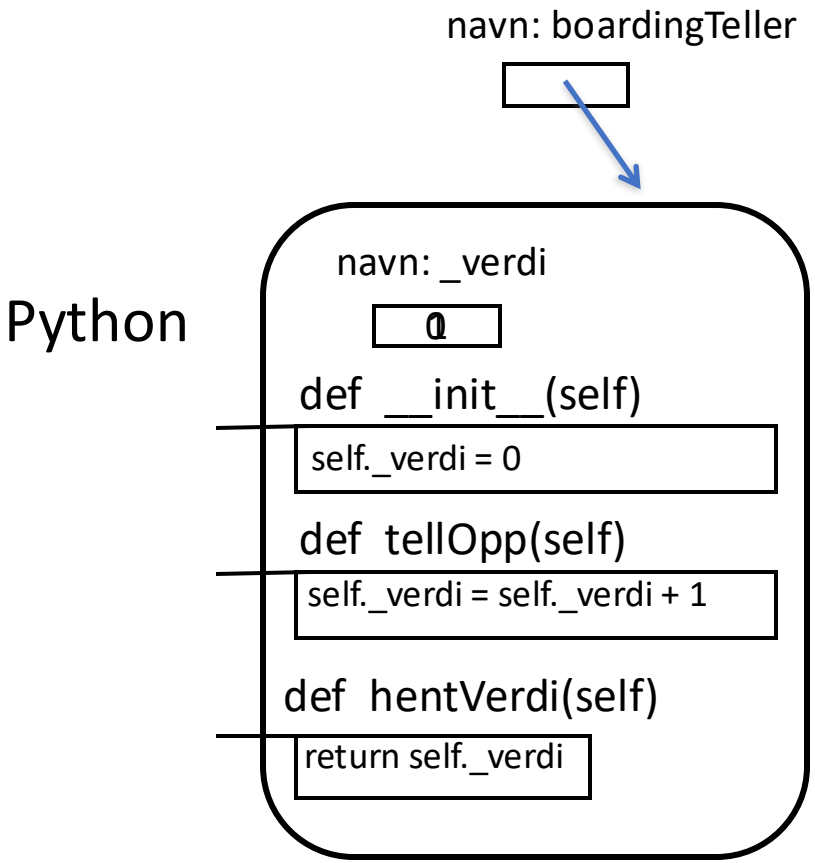
Java




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

Neste skritt er et hovedprogram som bruker Teller-klassen (kan utvides til å bli et aller første test-program)

1. Lage et program som deklarerer en Teller-klasse
2. La programmet opprette et Teller-objekt
3. Øk telleren i Teller-objektet med én
4. Les av telleren i Teller-objektet og lagre resultatet i en variabel i programmet



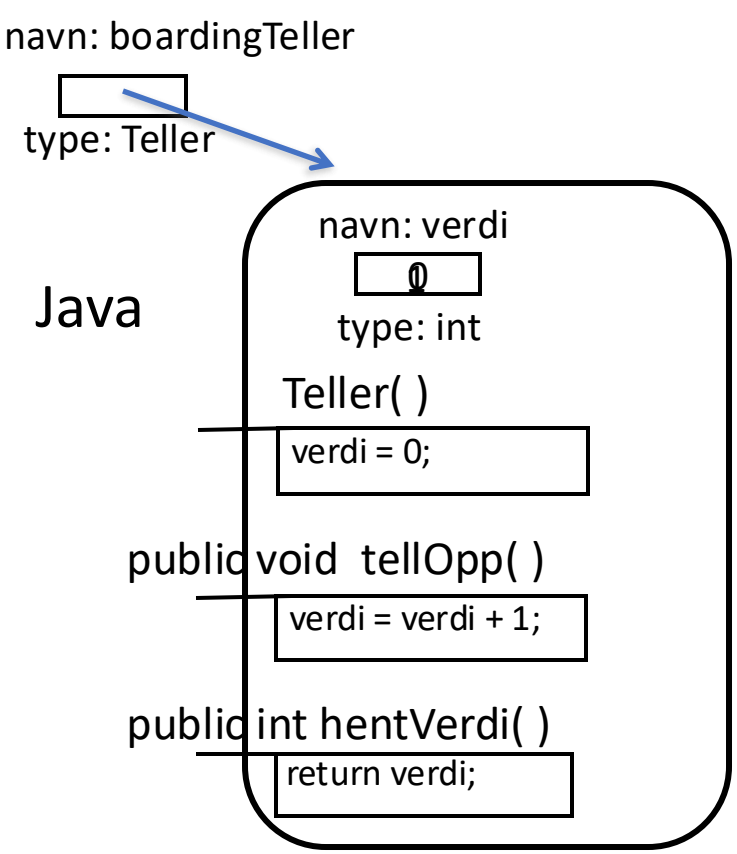

```
boardingTeller = Teller ( );

boardingTeller.tellOpp( );

tall = boardingTeller.hentVerdi( );
```

navn: tall

1




```
Teller boardingTeller = new Teller ( );

boardingTeller.tellOpp( );

int tall = boardingTeller.hentVerdi( );
```

navn: tall

1

type: int

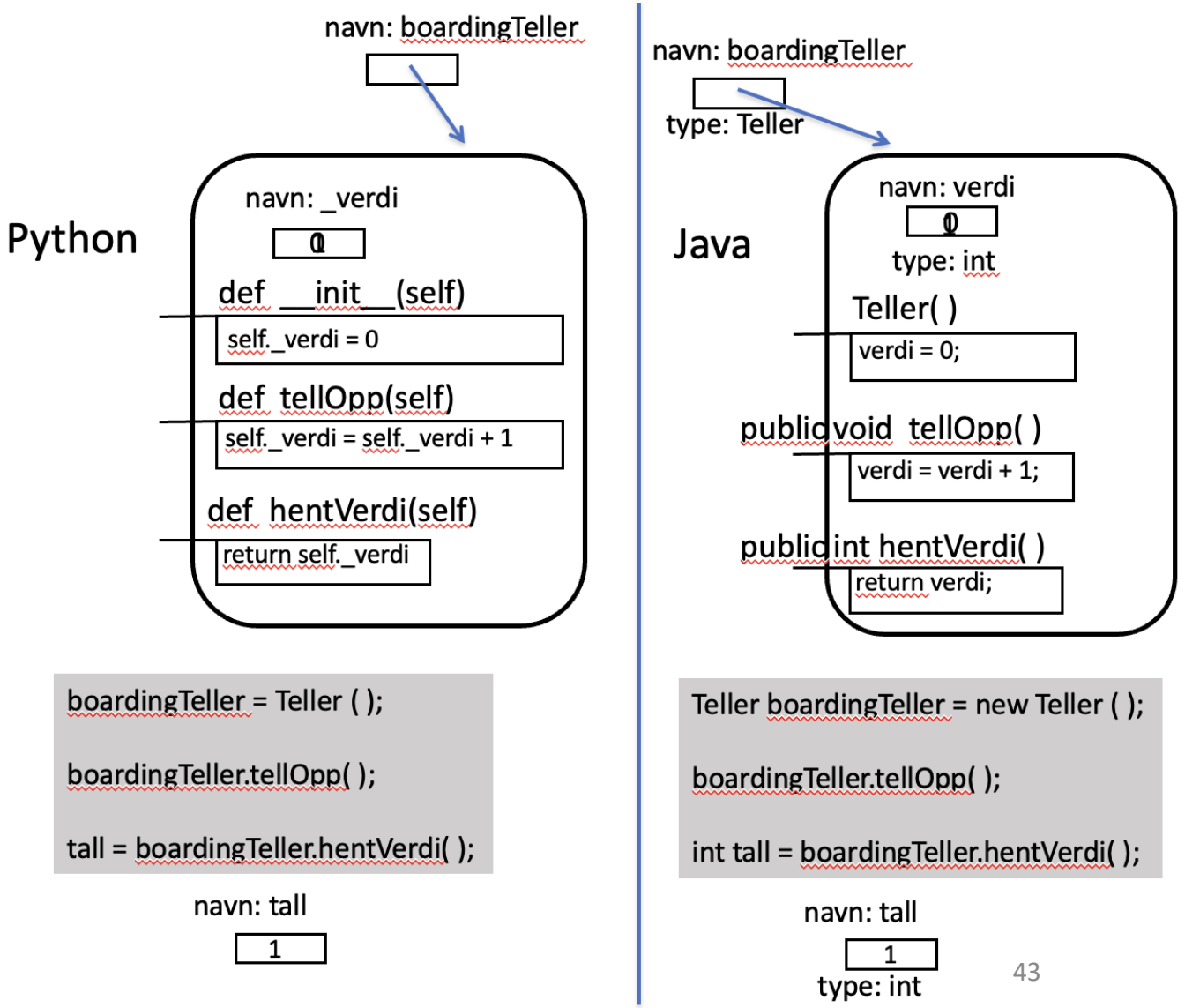
Et første fullstendig testprogram


```java
class Teller {
      private int verdi = 0;
      public void tell() {
           verdi = verdi + 1;
      }
      public int hentVerdi( ) {
           return verdi;
      }
}

class Test1Teller {
     public static void main (String [ ] arg) {
          Teller tell1 = new Teller( );
          tell1.tell();
          tell1.tell();
          int svar = tell1.hentVerdi();
          if (svar == 2) System.out.println("OK første test");
          else  System.out.println("Feiler første test");
      }
}
```

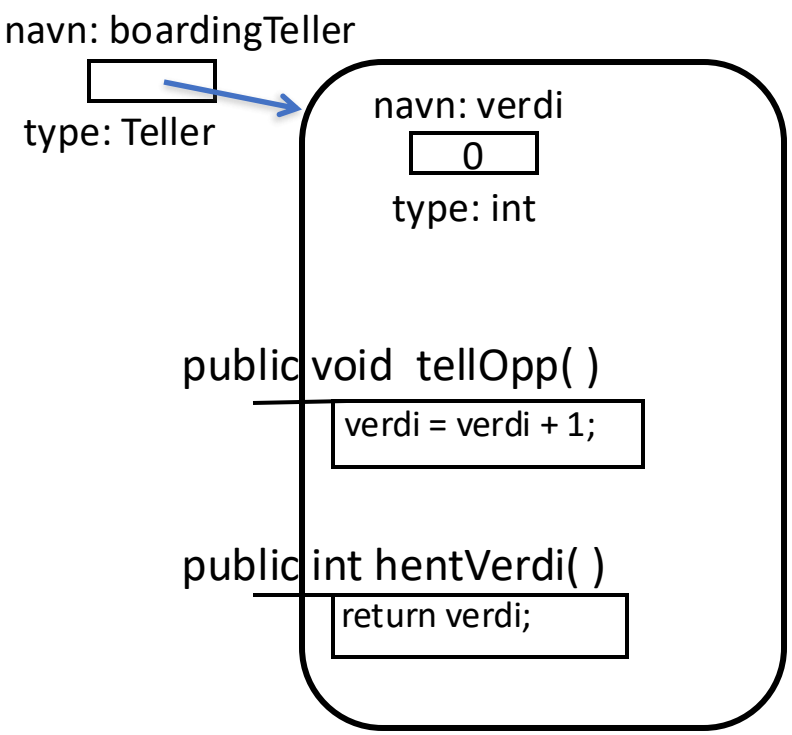
Sett dere sammen 2 – 3 – 4 – 5 i 5 minutter:

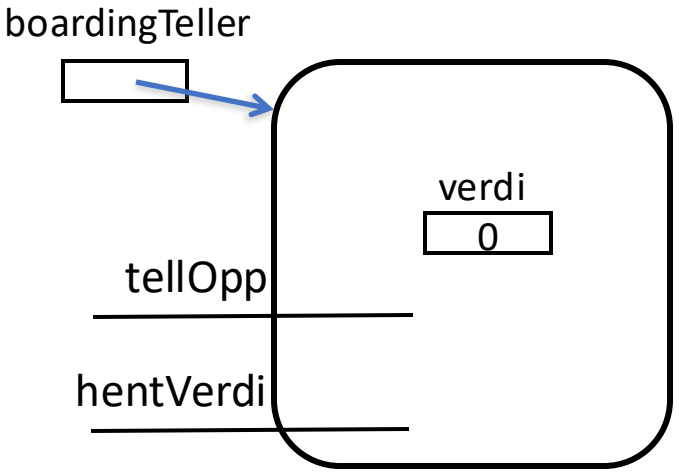
- Er det slik dere har tenkt på objekter i Python?
- Kan vi bruke disse tegningene til å forstå objekter (i Java)?

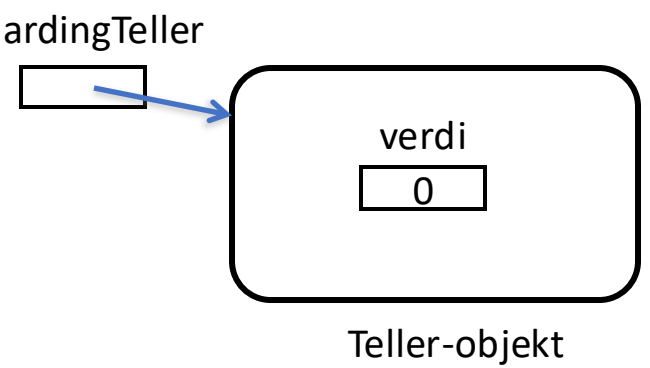


Råd om visualisering av et Teller-objekt

To enklere måter å visualisere det samme objektet på:







Den fulle og hele sannhet

Tegn (og tenk på) så mange detaljer om gangen at du selv skjønner hvordan objektene og variablene er/virker, og slik at du kan forklare hvordan algoritmen virker både for deg selv og for dem du samarbeidet med.

Del 4: "static" i Java

Hva betyr static i

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

En klasse er et mønster for å lage objekter

PLUSS:

- I Java kan vi deklarere egenskaper inne i en klasse som bare finnes èn gang
- Følgelig: Disse egenskapen blir IKKE gjentatt inne i hvert objekt
- Brukes når vi trenger felles egenskaper for ALLE objekter av samme klasse

- Ikke veldig viktig i IN1010, men noe vi bare må kunne

Java: **static**-egenskaper

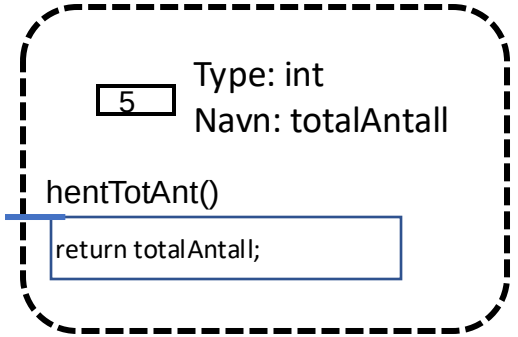

```
class Teller {
      private static int totalAntall = 0;
      private int verdi;
      public static int hentTotAnt ( ) {
            return totalAntall;
      }
      public void tellOpp( ) {
            verdi++;
            totalAntall++;
       }
      public int hentVerdi( ) {
            return verdi;
      }
}
```



static-egenskaper

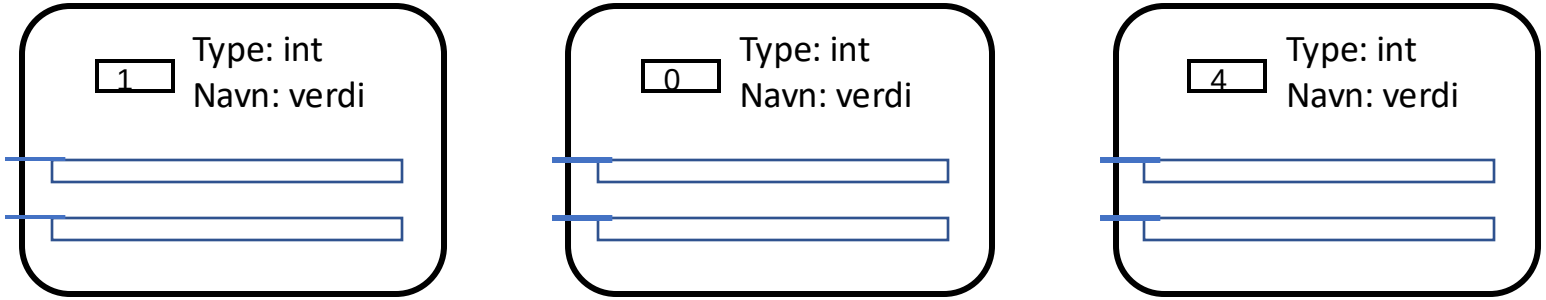

```
class Teller {
    private static int totalAntall =0;
    private int verdi;
    public static int hentTotAnt( ) {
        return totalAntall;
    }
    public void tellOpp( ) {
        verdi ++;
        totalAntall++;
    }
    public int hentVerdi( ) {
        return verdi;
    }
}
```

Når programmet starter lages en *klasse-datastruktur* av alle "static"-egenskapene til alle klassene i programmet



De tegningene vi har sett hittil av variable, metoder, objekter og klassedatastrukturer kaller vi til sammen Java **datastrukturer**

Etter f.eks. 3 kall på new Teller() har vi 3 objekter , og etter 5 kall på tellOpp() har instansvariablene f.eks. disse verdiene:



Fullstendig program


```
class Teller {
      private static int totalAnt = 0;
      private int verdi = 0;
      public void tellOpp( ) {
            verdi = verdi++;
            totalAnt++;
      }
      public int hentVerdi( ) {
            return verdi;
      }
      public static int hentTotAnt( ) {
            return totalAnt;
      }
}

class BrukTellerStatic {
     public static void main (String [ ] arg) {
            Teller tell1 = new Teller( );
            Teller tell2 = new Teller( );
            tell1.tellOpp();
            tell2.tellOpp();
            tell2.tellOpp();
            System.out.println("Teller en er " + tell1.hentVerdi());
            System.out.println("Teller to er " + tell2.hentVerdi());
            System.out.println("Tellerene totalt " + Teller.hentTotAnt());
      }
}
```

Del 5: Metodeinstanser (neste uke)

Et objekt = en instans av en klasse

- Men hva er en instans av en metode?
- Hva skjer når en metode blir kalt og utført?
- Jo, da opprettes det en **metodeinstans**.

La oss modifisere Teller-klassen litt

```
class Teller {
    private int verdi;

    public void tellOpp( ) {
        verdi = verdi + 1;
    }

    public int hentVerdi( ) {
        return verdi;
    }
}
```

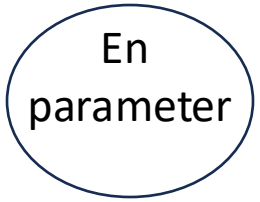


```
class Teller {
    private int verdi;

    public void tellOpp(int antall ) {
        verdi = verdi + antall;
    }

    public int hentVerdi( ) {
        return verdi;
    }
}
```



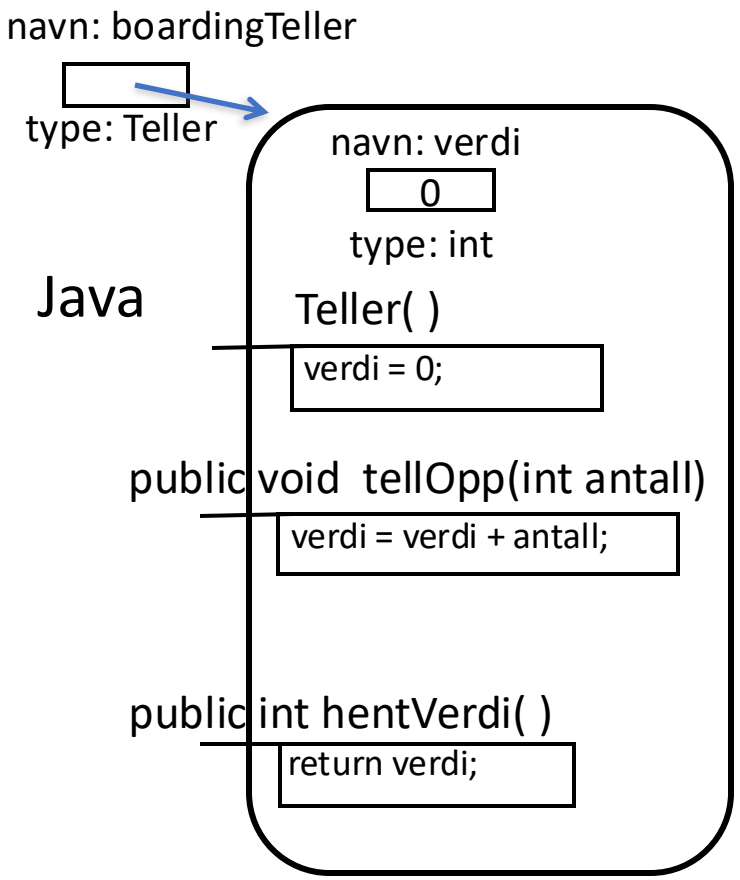



```
Teller boardingTeller = new Teller ( );

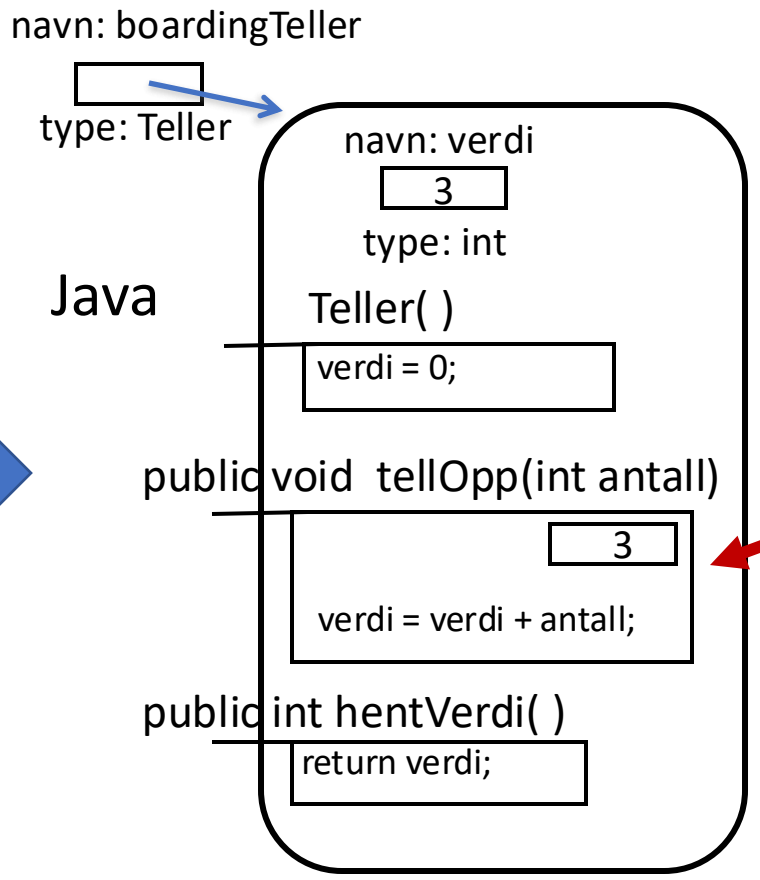
boardingTeller.tellOpp(3);

System.out.println(boardingTeller.hentVerdi( ));
```

Objekt med illustrasjon av metoder som **kan kalles**



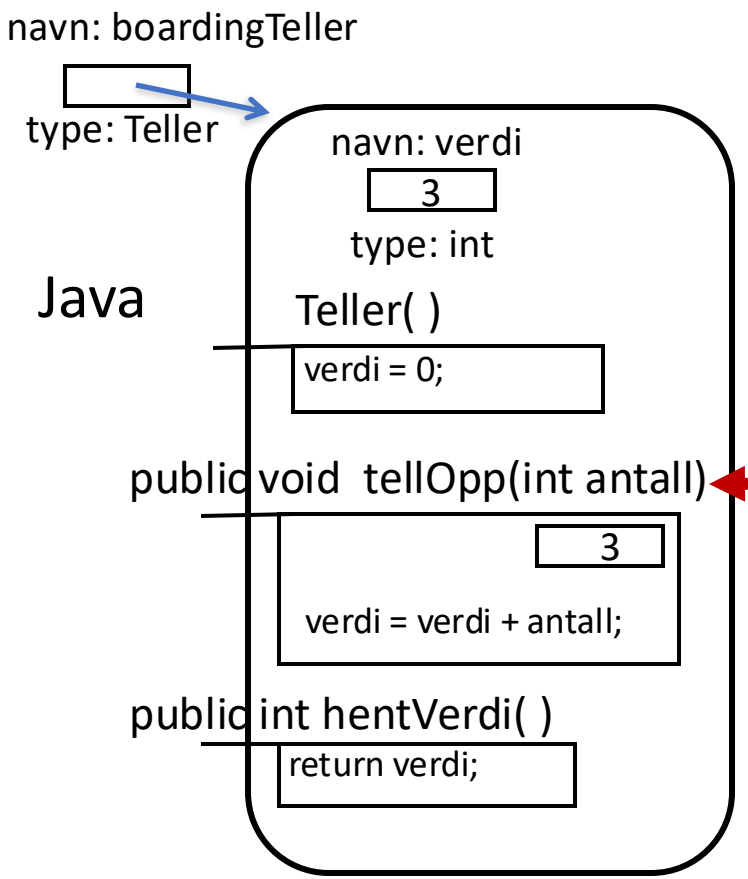
Objekt med illustrasjon av metode som også **er kalt (som er under utførelse)**



Dette er en **metodeinstans**

boardingTeller.tellOpp(3);

Metodekall - metodinnstanser



Når en metode kalles beregnes uttrykket som er den **aktuelle parameteren** på **kallstedet**

Verdien av uttrykket (parameterverdien) tas med til metoden og det opprettes en **metodeinstans** der den **formelle parameteren** blir en **lokal variable** med parameterverdien som startverdi

Formell parameter

Aktuell parameter (som er et uttrykk)

boardingTeller.tellOpp(3);

```
int ant = 2;
boardingTeller.tellOpp(ant + 1);
```

Les mer i notatet: “Mer om objekter og klasser i Java”

Objekter og metodeinstanser

- Det settes av plass i minnet til instansvariabler hver gang vi sier new . . .
  og det opprettes et **objekt** (= instans av en klasse)
  - Den plassen i minnet som brukes til dette kalles ”haugen” (eng: the heap)

- Det settes av plass i minnet til lokale variabler (inkludert aktuelle parametre) hver gang vi kaller en metode og det opprettes en **metodeinstans**
  - I IN1010 kan vi forestille oss at en metodeinstans oppstår inne i et objekt
  - I objektorientert tankegang gir dette oss en god tankemodell for hvilke egenskaper i objektet metoden har tilgang til (skop)
    - Men objektet blir ikke utvidet med mer data. Dataene til metodeinstansen blir lagt på stakken (se under)
  - Men i virkeligheten kalles den plassen i minnet som brukes til lokale variabler i metodeinstanser for stakken (eng: the stack)
  - Les mer om hvordan metoder kalles og plass til lokale variabler settes av på stakken i notatet “Mer om objekter og klasser i Java”.

I dag har vi lært

- At objektorientering gir oss muligheten til å modellere (en del av) verden inne i datamaskinen
- At en variable har et navn, en type og en verdi
  - At verdien forandres ved tilordning
- At et objekt innholder data (variabler som vanligvis er private) og metoder som vanligvis er public (objektets grensesnitt)
- At static variabler og metoder bare finnes ett sted
- At når en metode blir kalt inne i et objekt tenker vi oss at det oppstår en metodeinstans inne i objektet
  - At uttrykket som er aktuell parameter beregnes og gir startverdi til den formellle parameteren som blir en lokal variabel i metodeinstansen