UNIVERSITETET I OSLO

Endelig versjon, lagt ut etter forelesningen *

Institutt for informatikk

IN1010 - våren 2023

Tirsdag 24. januar

Java
Verdier og variabler
Objekter, klasser og referanser
Metoder og metodeinstanser
static

Stein Gjessing

- På lysark 33 er navnene på to variabler lagt til helt høyre

Agenda

- Rask intro til enkel innlesing og utskrift i terminalvindu (2 lysark)
  - Mer i gruppene og neste mandag
- Verdier og variabler i datamaskinen og i Java - arrayer i Java
- Objekter og klasser i Java
- Metoder og metodeinstanser

Lesing og skriving i terminalvinduet


```java
import java.util.Scanner;

class LesFraTerminal {
    public static void main (String [ ] args) {
        int alder;
        String navn, adresse;
        Scanner minInn =  new Scanner (System.in);
        System.out.print(" Skriv adressen din: ");
        adresse = minInn.nextLine();
        System.out.print(" Skriv fornavnet ditt: ");
        navn = minInn.next();
        System.out.print(" Skriv alder: ");
        alder = minInn.nextInt();
        System.out.println( navn + ", du bor i " +
              adresse + " og er " + alder + " år" );
    }
}
```

Men pass på.
F.eks.
next() og
nextInt()
leser ikke
hele linjen

Mer i gruppene og mandag 30. januar
Midlertidig forklaring neste side

Midlertidig forklaring av noen detaljer

java.util.Scanner	klassen Scanner er definert i biblioteket util
new Scanner()	lager et nytt Scanner-objekt
Scanner minInn = new Scanner (System.in);	lager et nytt Scanner-objekt som leser fra terminalvinduet OG setter variablen minInn til å referere dette objektet
String adresse = minInn.nextLine();	returnerer en referanse til en hel linje fra minInn (mer senere)
String navn = minInn.next();	returnerer en referanse til en tekst fra minInn (mer senere*)
int alder = minInn.nextInt();	returnerer et heltall fra minInn (mer senere)

Google: Java API 8 Scanner

* senere = neste mandag

IN1010: Objektorientert programmering

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

Men først: Hva er en variabel (i Java) ?

```
int antall;
int flere;

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

eller

```
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;
```

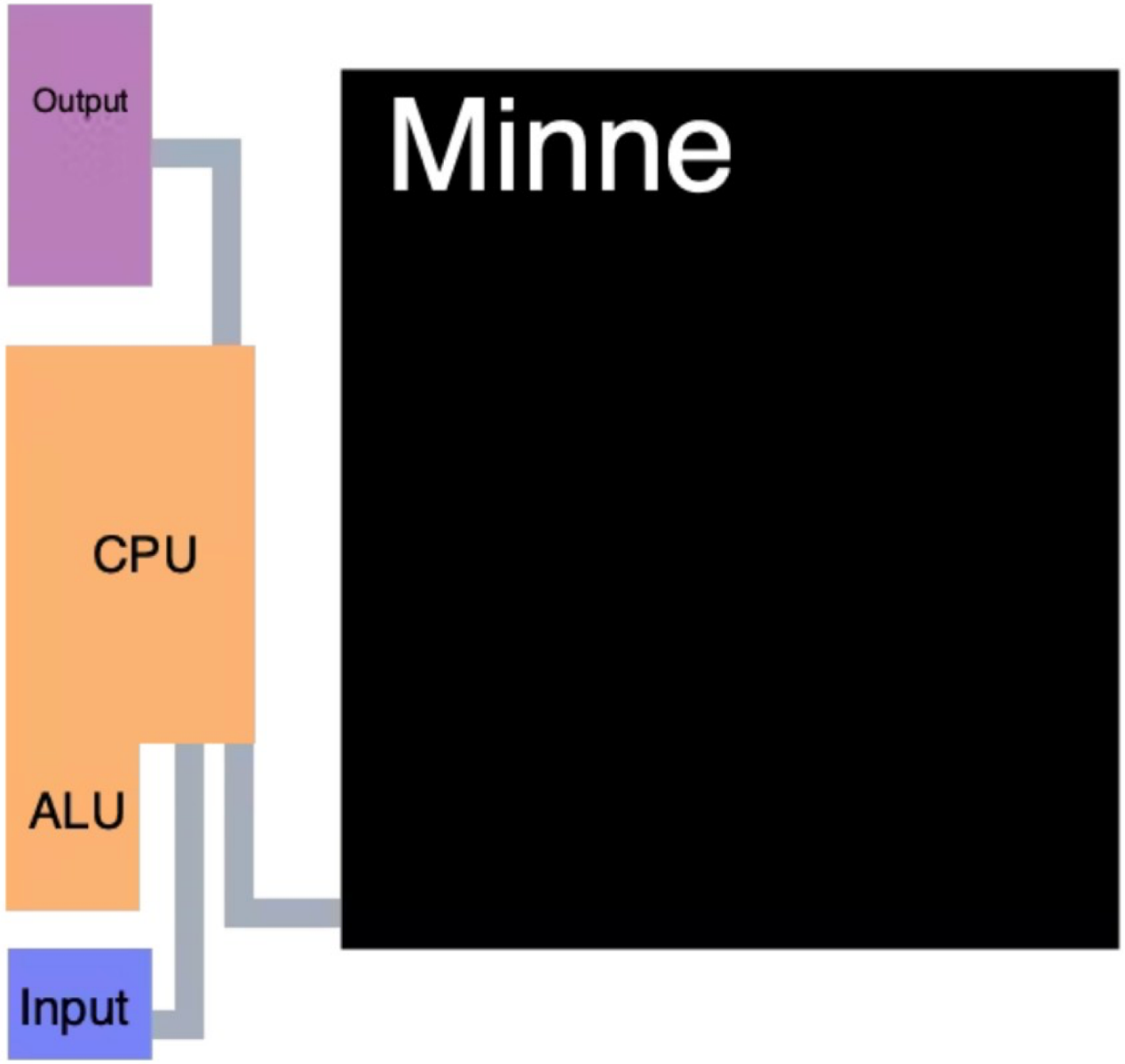
antall 00000000000000000000000000000110

flere 00000000000000000000000000000100

UNIVERSITETET I OSLO

Institutt for informatikk

Repetisjon IN1020:

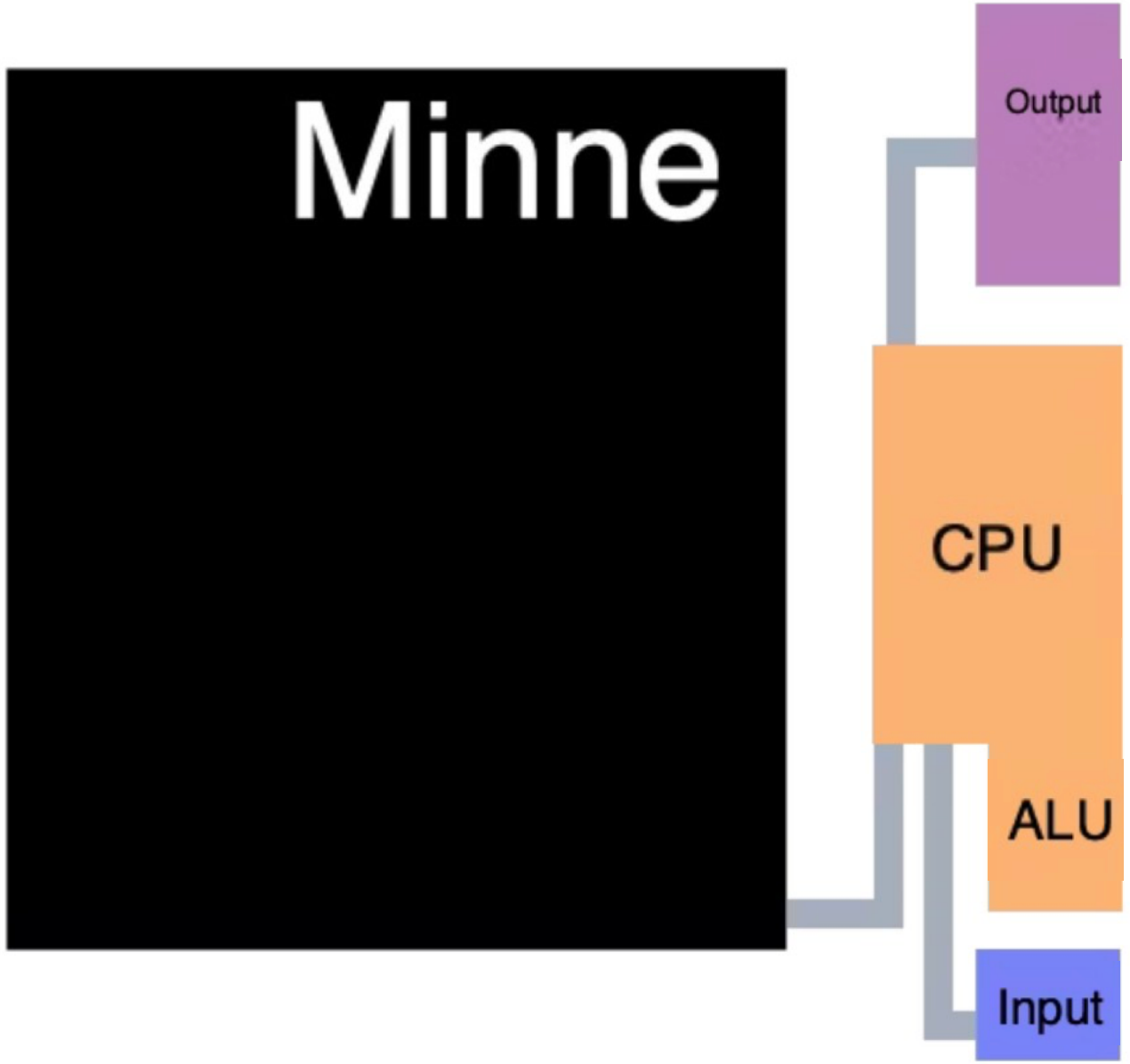


LMC - forenklet

Dette er en datamaskin

La oss speilvende den

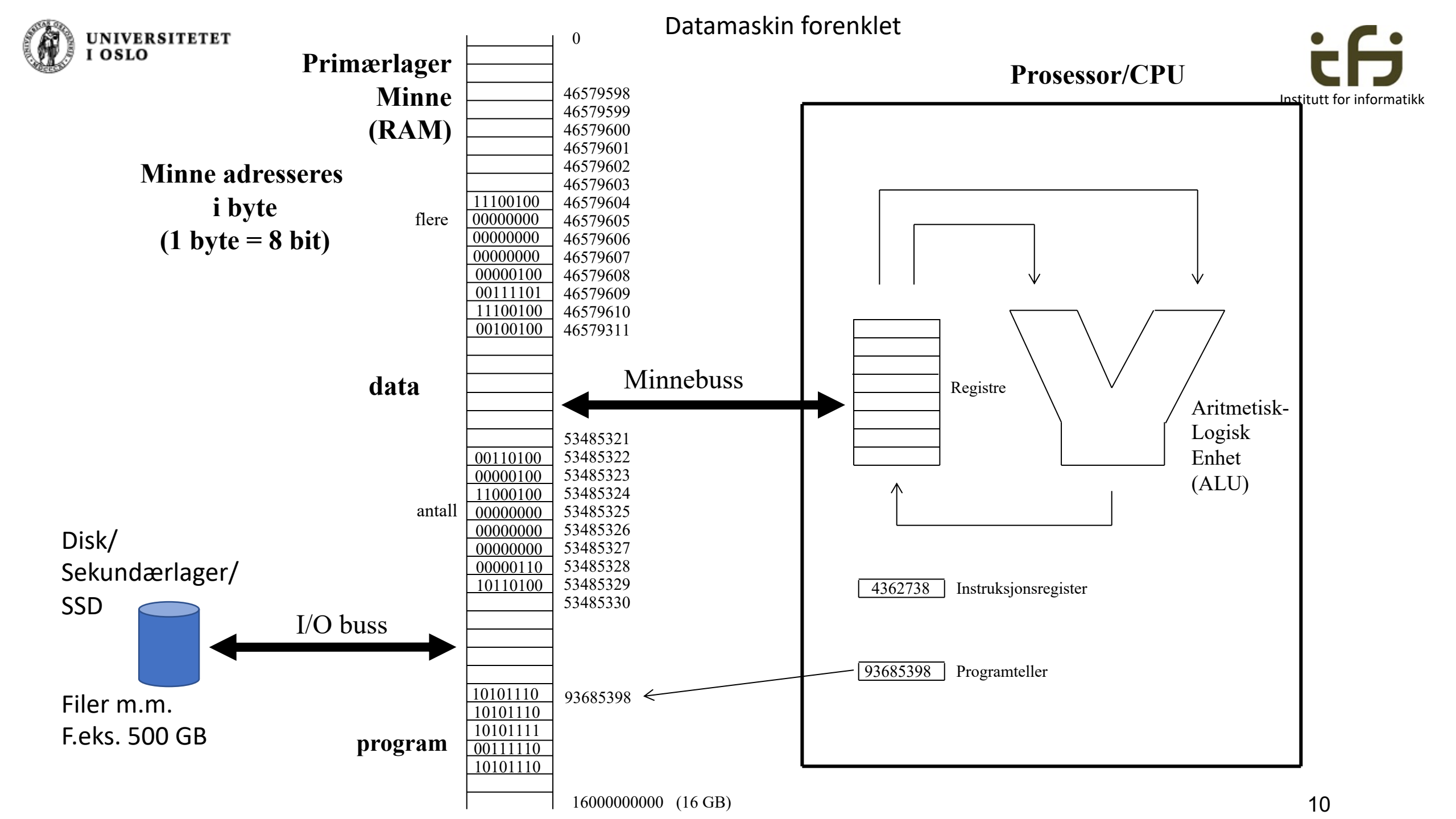
NESTE SIDE



Samme figur
på neste side

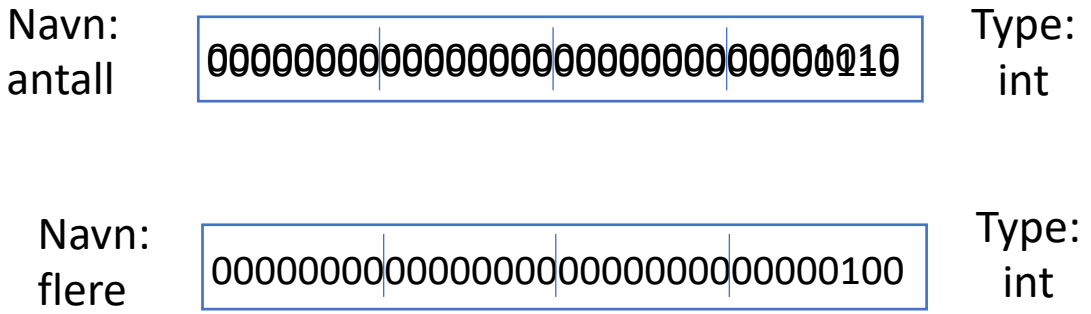
LMC - forenklet = Datamaskin forenklet

Datamaskin forenklet



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 ikker navnet kan forandres).
Innholdte til en variable forandres ved **tilordning**

Javas verdier

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

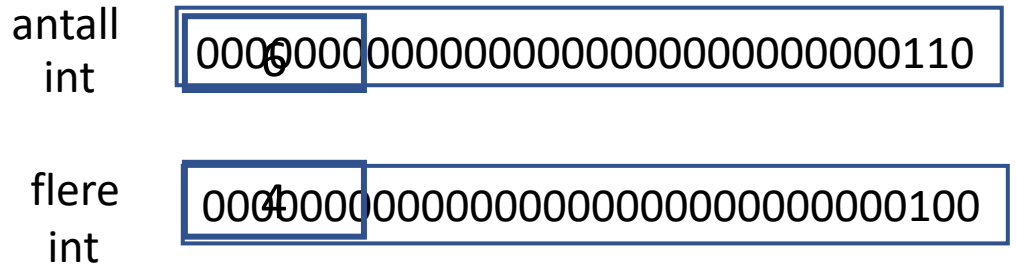
boolean	~~1~~ 00000001	8 bit	(Det er for upraktisk å lagre (sammen med andre data), pakke ut og inn og teste ett bit)
byte	01001110	8 bit	
char	0100111010110110	16 bit	
short	0100100101101110	16 bit	
int	01010010110110101000100101101110	32 bit	
long	0101001011010011010110100010101001010110100110101000100101101110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

IEEE standard 754

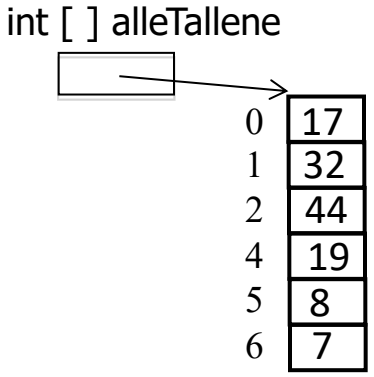
Det er disse verdiene som kan ligge i en primitiv variabel

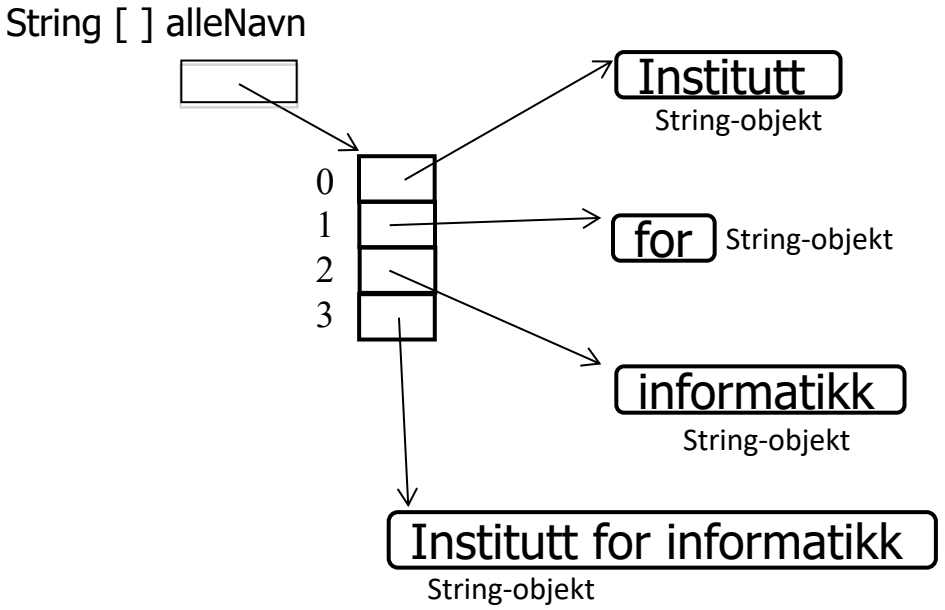
Slik tenker vi på og tegner en variabel i Java

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



Flere sammenhengende variabler er en **array**

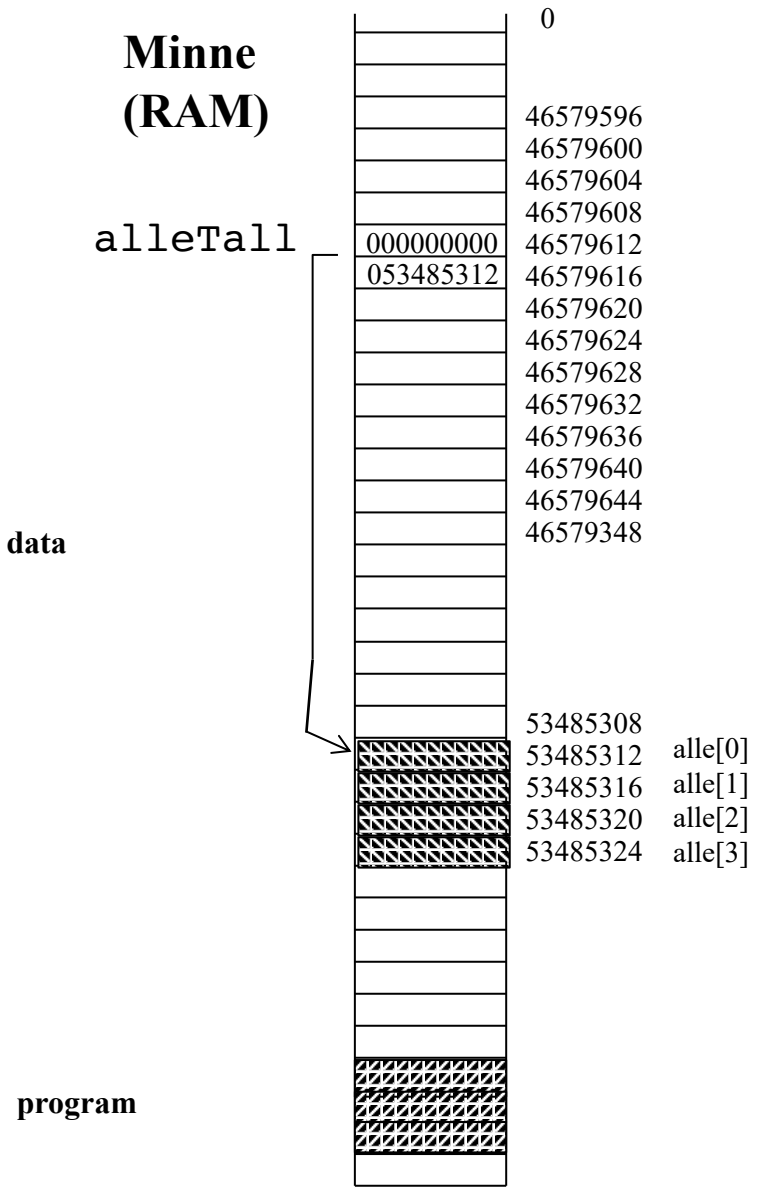





```
int [ ] alleTallene;
alleTallene = new int[7];
// Men innholdet nå er bare 0-er
```

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

Detaljer i RAM (men ikke viktig i IN1010)




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

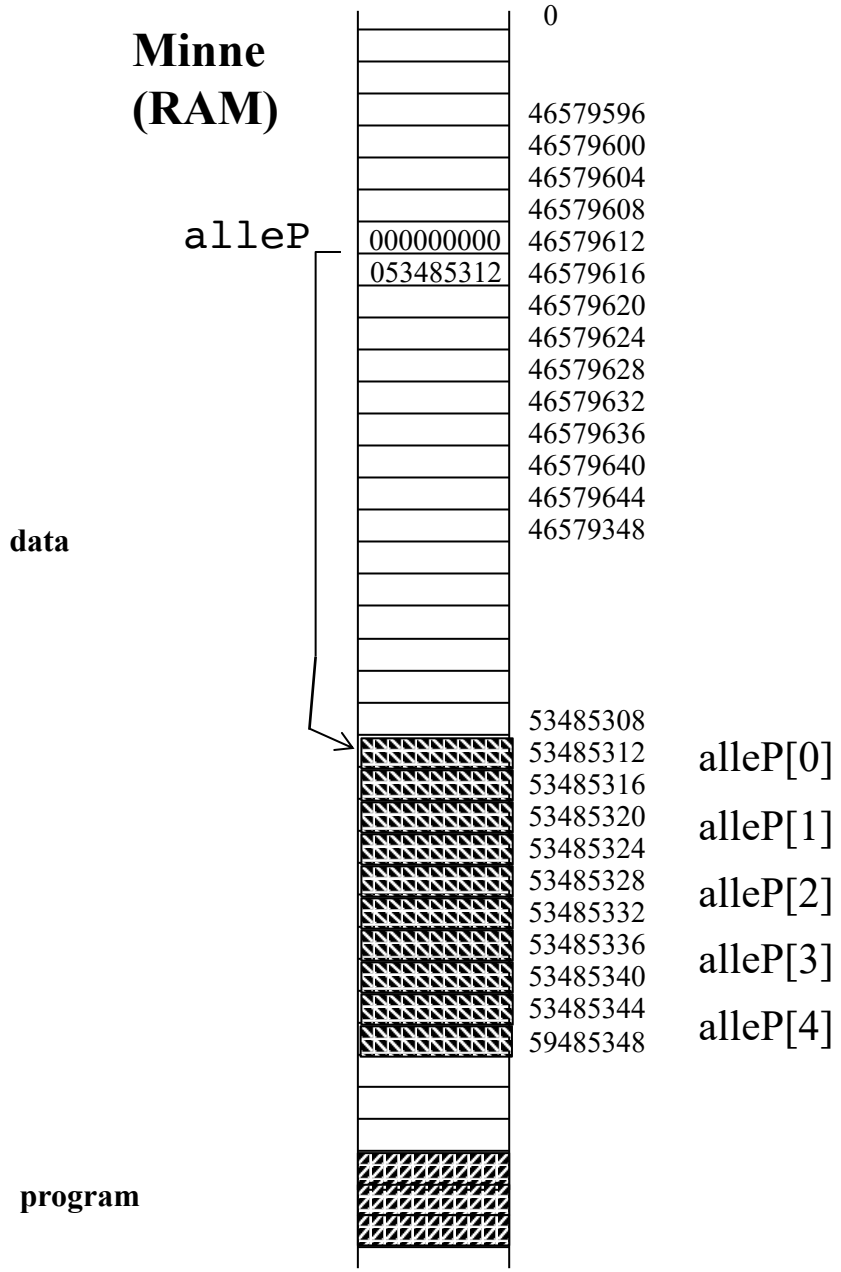
Array er iboende i datamaskinen og i Java

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 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

Noen programsnutter med **array** i

```
int[ ] alleTall;
alleTall = new int[7];

//noen helt tøysete tilordninger:

alleTall[3] = 7;
alleTall[2] = 4;
alleTall[1] = alleTall[3] + alleTall[2] + alleTall[0];


// Når en array deklareres kan den gis verdier:
int[ ] noenTall = { 1, 2, 3, 5, 7, 11 } ;
```

```
String[ ] alleNavn = new String[4];

//enda flere tøysete tilordninger mm:

alleNavn[0] =  “Institutt”;
alleNavn[2] =  “informatikk”;
alleNavn[1] =   “for”;

alleNavn[3] =  alleNavn[0] + “ “ + alleNavn[1]
                          + “ ” + alleNavn[2];

System.out.println(“Navnet er: “ + alleNavn[3]);
```

Når en array lages med “new” blir innholdet i arrayen alltid standardverdien til typen (f.eks. 0 eller null)

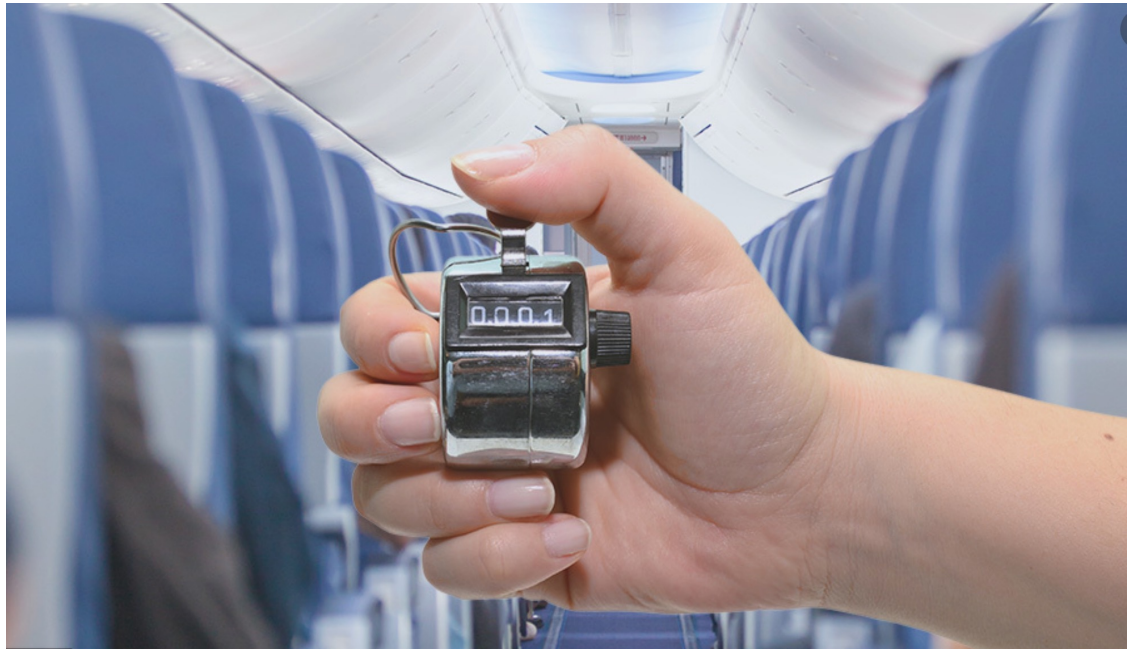
Nå, over til

Objektorientert programmering

- 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

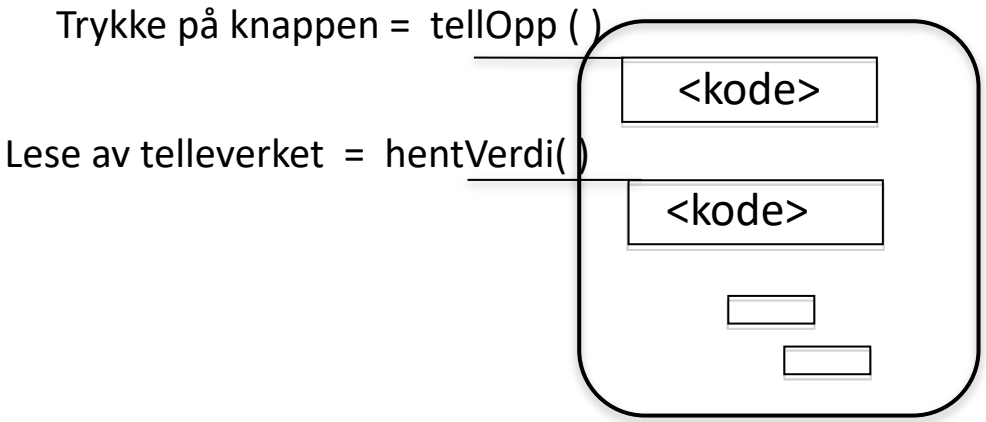


Objekter i den virkelige verden modelleres av objekter inne i datamaskinen

Et objekt tilbyr tjenester til omverdenen

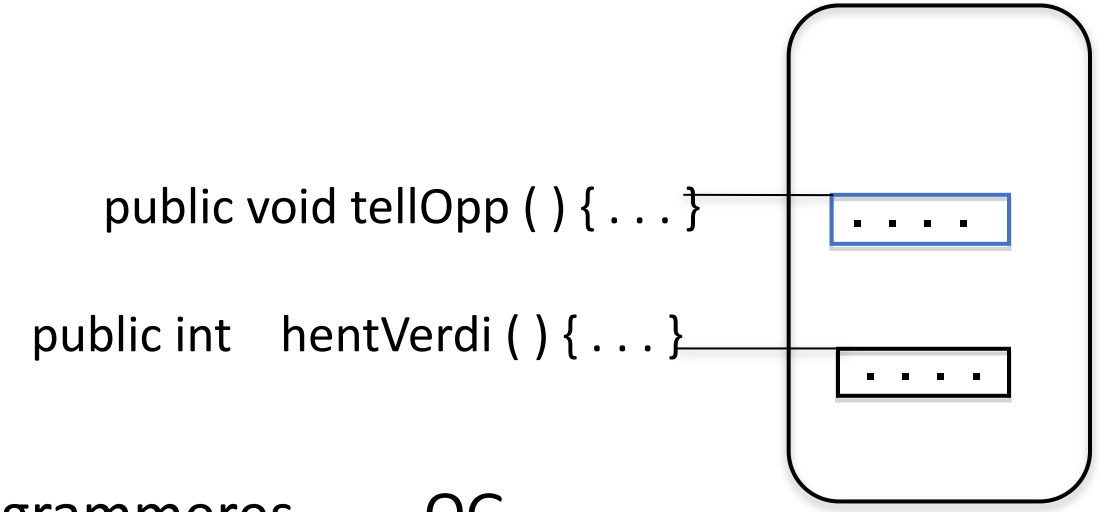
Et objekt er en “sort boks” + grensesnittet mot omverdenen

Et objekt er en samling av data og handlinger



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:

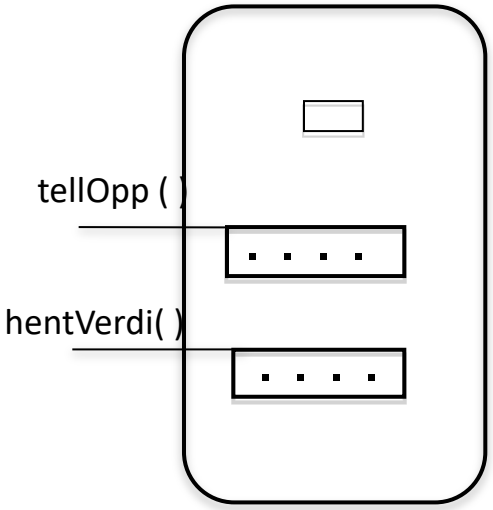


- Hvordan skal tellOpp() og hentVerdi() programmeres OG
- Hvilke private **data** må være inne i et slikt teller-objekt?

Da må vi se for oss et teller-objekt

- Er det nok med èn variabel inne i objektet?

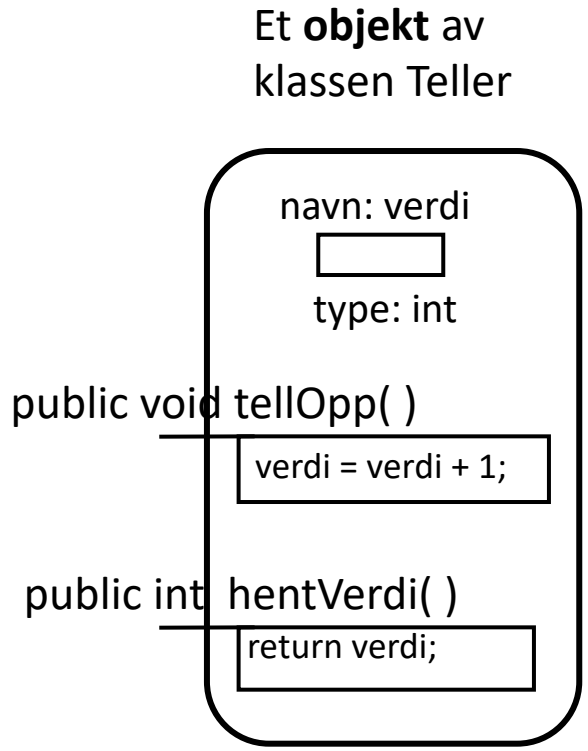
- Er det nok at
- tellOpp() teller opp denne verdien med èn og at
- hentVerdi() leser av denne verdien ?



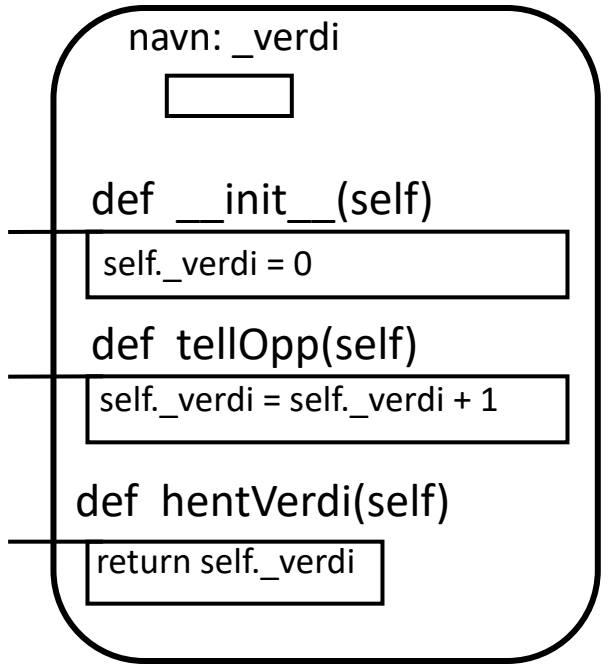
Det er dette som er å programmere !!
Men vanligvis er problemet vanskeligere !!

Et forslag til et teller-objekt

- Generelt innholder objekter
  - **Metoder – tjenester – handlinger**
    - public (som regel)
    - 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)

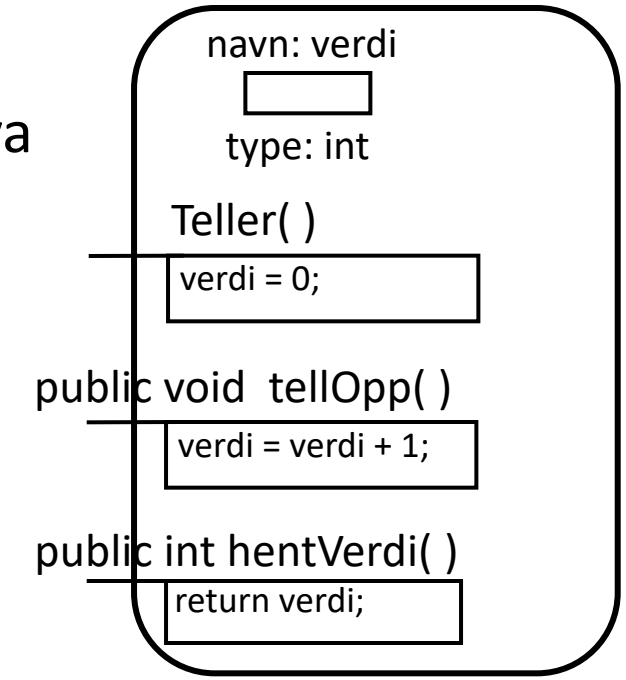


Python




```
class Teller:

   def __init__(self):
      self. _verdi = 0

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

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

Java

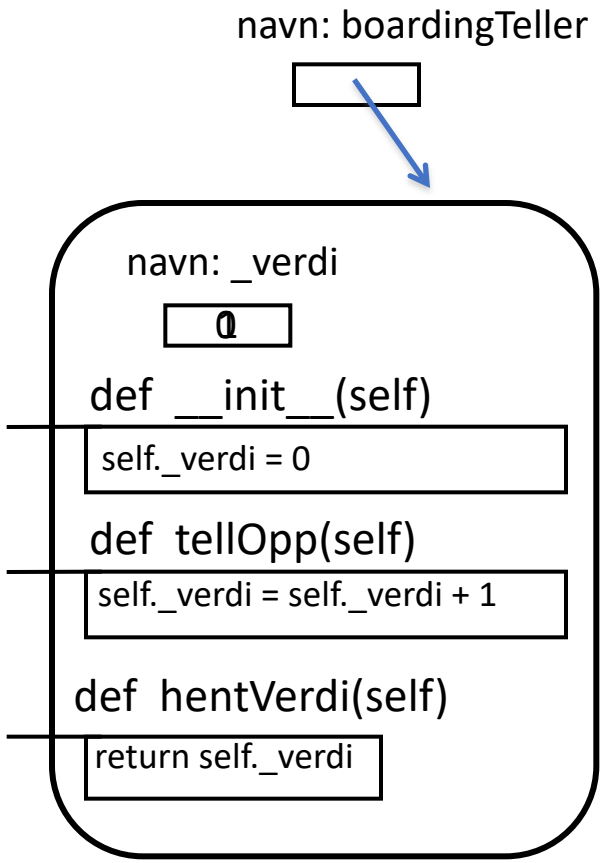



```
class Teller {
    private int verdi;
    Teller() {
        verdi = 0;
    }
    public void tellOpp( ) {
        verdi = verdi + 1;
    }
    public int hentVerdi( ) {
        return verdi;
    }
}
```

Neste skritt

1. Lage et pogram 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

Python

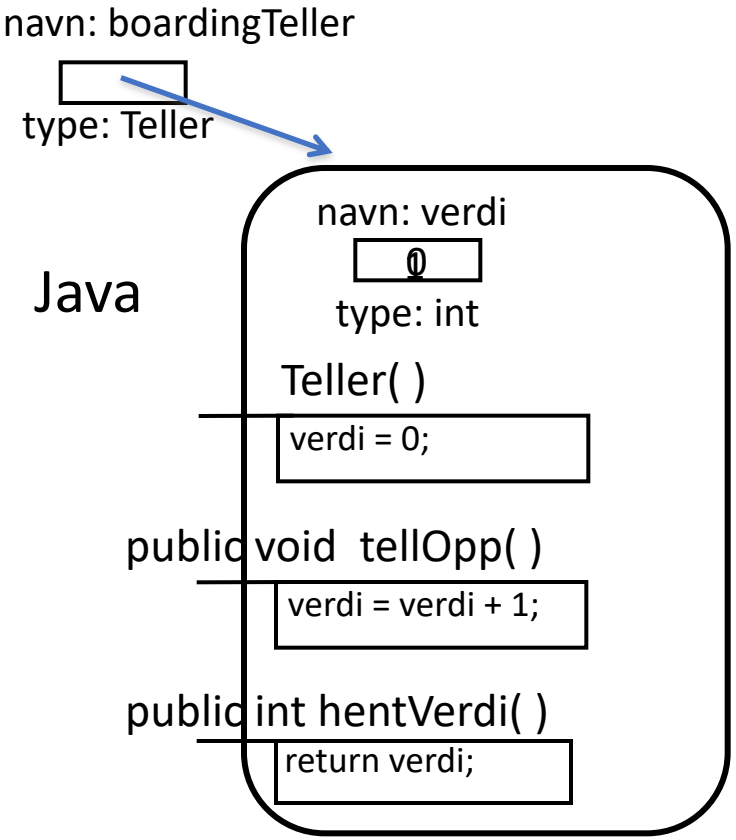



```
boardingTeller = Teller ( );

boardingTeller.tellOpp( );

tall = boardingTeller.hentVerdi( );
```

navn: tall

1

Java




```
Teller boardingTeller = new Teller ( );

boardingTeller.tellOpp( );

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

navn: tall

1

type: int

Fullstendig program (- +)


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

class BrukTellerBasis {
     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());
      }
}
```

Nytt tema: Java og «static»

- 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

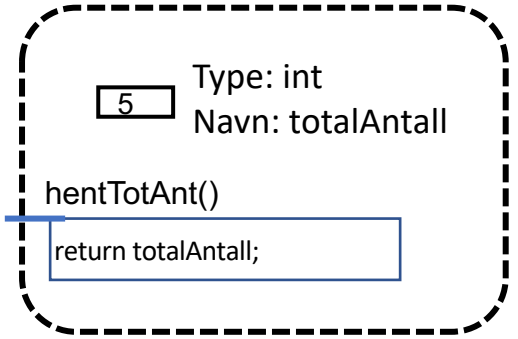
Java: **static**-egenskaper


```
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;
      }
}
```

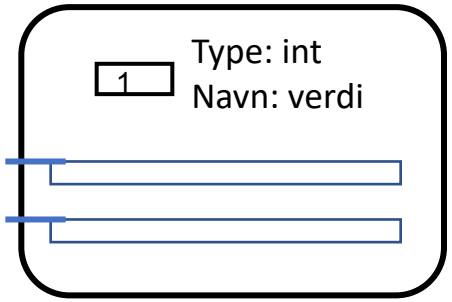
static-egenskaper

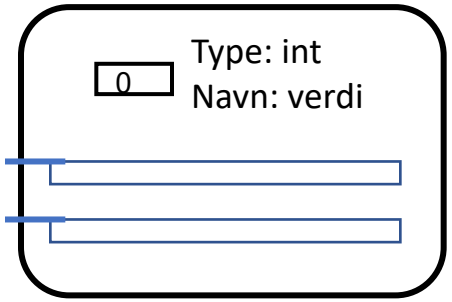

```
class Teller {
    private static int totalAntall =0;
    private int verdi;
    public static int totalAntall ( ) {
        return totalAntall;
    }
    public void tellOpp( ) {
        verdi ++;
        totalAntall++;
    }
    public int hentVerdi( ) {
        return verdi;
    }
}
```

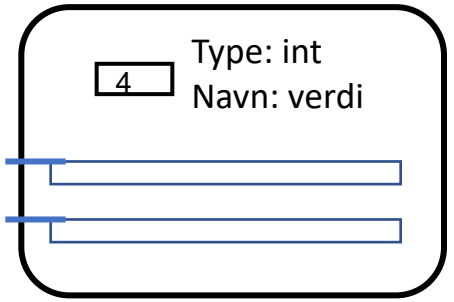
Når programmet starter lages en *klasse-datastruktur* av alle "static"-egenskapene til alle klassene i programmet



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







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

Fullstendig program

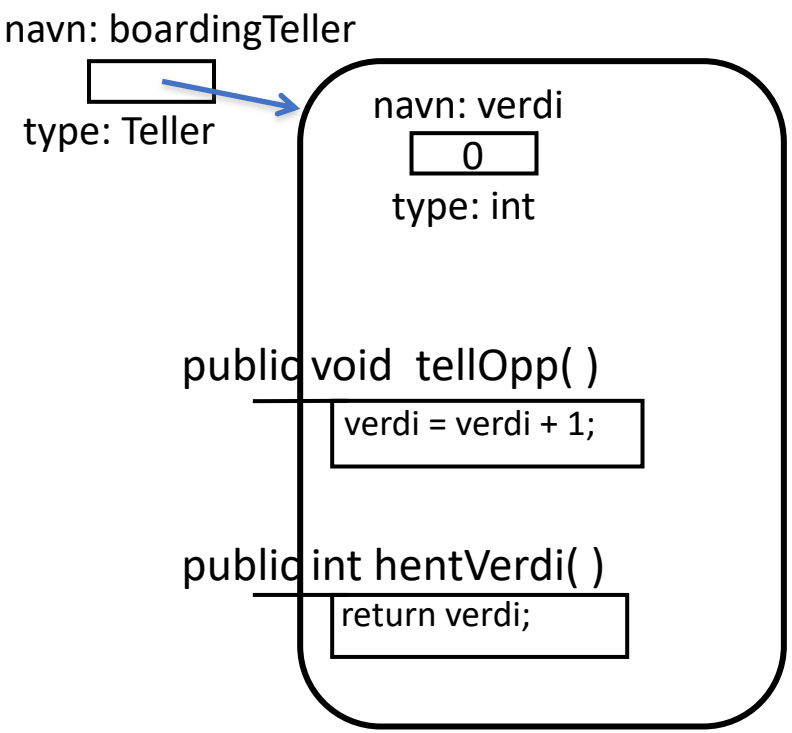

```
class Teller {
      private static int totalAnt = 0;
      private int verdi = 0;
      public void tellOpp( ) {
            verdi = verdi++;
            totalAnt++;
      }
      public int hentVerdi( ) {
            return verdi;
      }
      public static int hentTotAnt( ) {
            return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

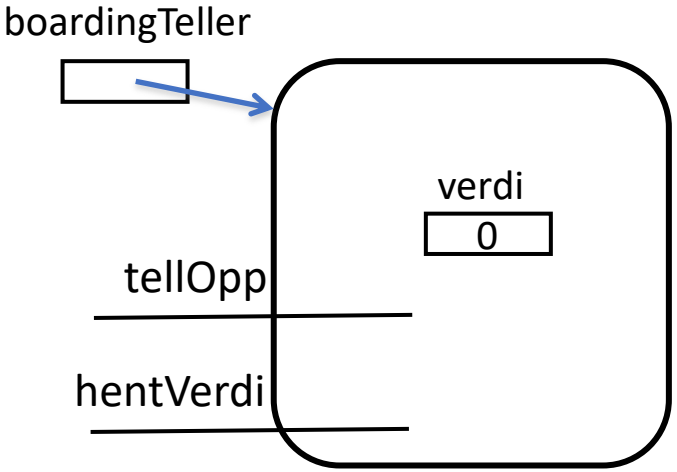
Innleveringsoppgave 0: Hvor nøyaktig skal jeg tegne datastrukturen ?

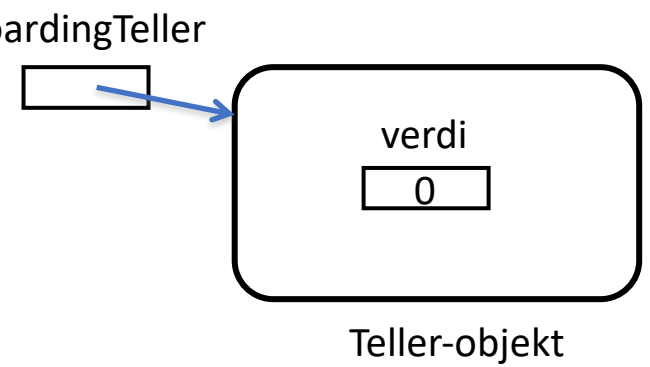
- **Svar:**
- Generelt:
  Så nøyaktig som det er nødvendig for at du selv eller dem du samarbeider med skal skjønne hva som skjer med datastrukturen når programmet (algoritmen) utføres
- I innleveringsoppgave 0:
  Så nøyaktig at din retter skjønner at du skjønner hvordan datastrukturen ser ut.

Eksempel fra teller-programmet

To måter å tegne det samme objektet på:







Den fulle og hele sannhet
Ikke tegn slik i innleveringsoppgave 0

Tegn så mange detaljer 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.
Det finnes ingen fasit for hvordan dette skal gjøres.
Men referanser mellom objekter bør (nesten) alltid med, se notatet «Objekter og klasser i Java»

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**

navn: boardingTeller

type: Teller

Java

navn: verdi

0

type: int

Teller( )

verdi = 0;

public void tellOpp(int antall)

verdi = verdi + antall;

public int hentVerdi( )

return verdi;

Objekt med illustrasjon av metode som også **er kalt**

navn: boardingTeller

type: Teller

Java

navn: verdi

3

type: int

Teller( )

verdi = 0;

public void tellOpp(int antall)

3

verdi = verdi + antall;

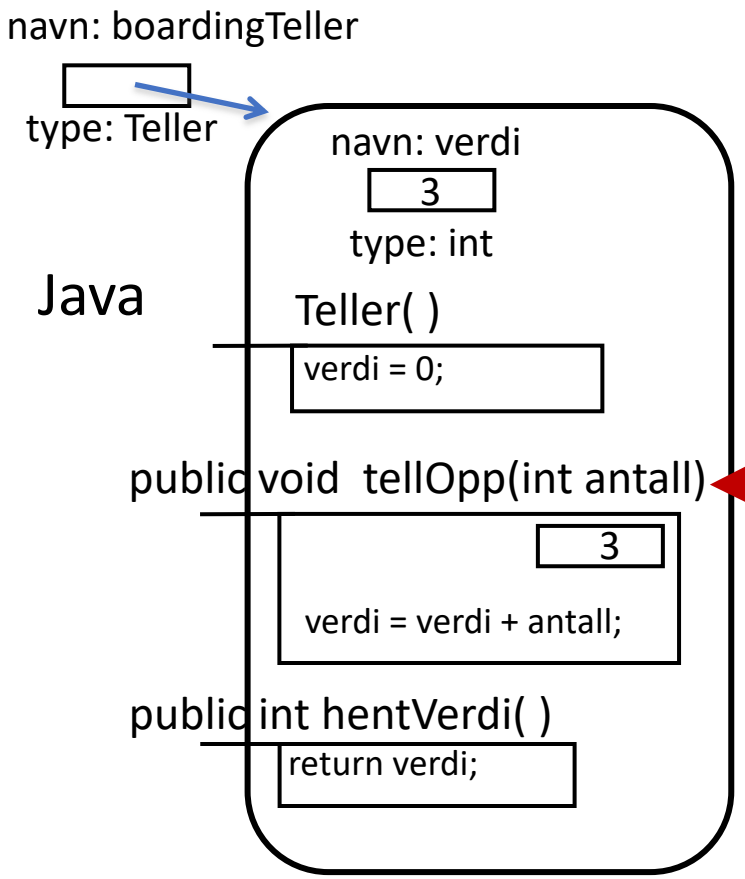
public int hentVerdi( )

return verdi;

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 klassen)
  - 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. Dataen 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 (1/3)

Programmering er å løse problemer ved hjelp av datastrukturer og algoritmer

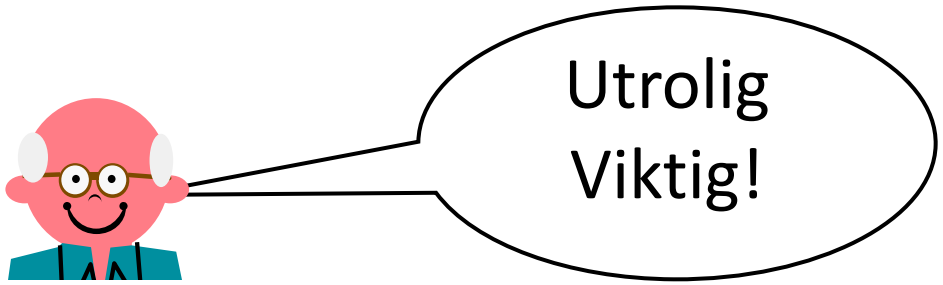
- Når vi skal løse et problem må vi tenke oss en **datastruktur** som løser problemet
- Selve løsningen av problemet er manipulering av denne datastrukturen (en **algoritme**)
- Når du skal løse et problem:
  - Tenk deg og eventuelt tegn først datastrukturen
  - Deretter kan du skrive koden (algoritmen) som lager og manipulerer datastrukturen og løser problemet

Vi skiller mellom

- Klasse-deklarasjonen i programteksten. Den er et mønster som brukes både når klassedatastrukturen lages (i det programmet starter opp) og senere når nye objekter og metodeinstanser lages.

- Klasse-datastrukturen, dvs. den (statiske) datastrukturen som lages i det programmet starter.

- Objekt-datastrukturen (også kalt klasse-instanser, klasse-objekter eller bare objekter) som lages hver gang vi sier new.



I dag har vi lært (3/3)

- Hva en verdi er
- Hva en variabel er
- Hva et objekt er
- Hva en klasse er
- Hva en metodeinstans er
- Hva en metode er
- Hvordan vi visualiserer / tegner variabler, metoder og objekter
- Hva "static" betyr i Java og hvordan vi visualiserer / tegner klassedatastrukturer

Et litt større eksempel til egetstudium

- Hensikten med dette eksemplet er at dere skal lære Java.
- Hvordan ser et Java-program ut?
- Hva skjer når et Java-program kjører?

- Bildet på siste side kan animeres i PowerPoint
  - Koden ligger under ressurser uke 1.

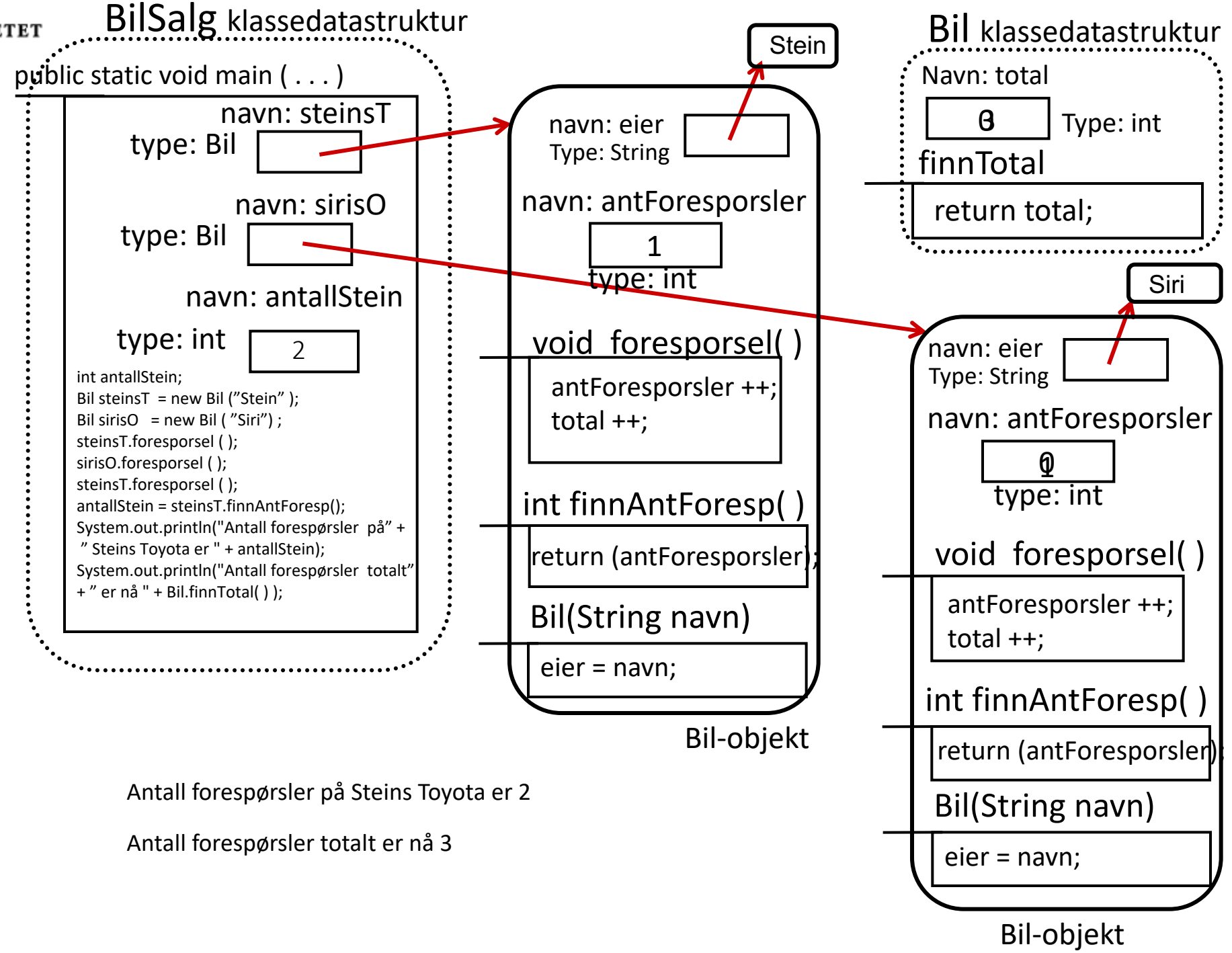
Et litt større eksempel

- Du har en venn som er bruktbilselger, og du skal hjelpe ham med å lage et program for å holde orden på hvor mange som er interessert i de enkelte bilene han har til salgs.
- Først tegner du to bil-objekter, så lager du et program som lager og bruker disse to bil-objektene.
  Dette programmet skal du senere utvide . . .
- Etter at du først tegnet datastrukturen og så programmerte en stund kom du fram til programmet på neste side
- Hvordan tenkte du?
- Hvordan virker dette programmet?

Program for salg av biler.


```
public class BilSalg{
  public static void main (String [ ] args) {
     int antallStein;
     Bil steinsT  = new Bil (”Stein”);
     Bil sirisO   = new Bil (”Siri”) ;
     steinsT.foresporsel ( );
     sirisO.foresporsel ( );
     steinsT.foresporsel ( );
     antallStein = steinsT.finnAntForesp();
     System.out.println("Antall forespørsler  på” +
          ” Steins Toyota er " + antallStein);
     System.out.println("Antall forespørsler  totalt” +
          ” er nå " + Bil.finnTotal( ) );
  }
}
```

```
class Bil {
  private static int total = 0;
  private String eier;
  private int antForesporsler = 0;

  public Bil (String  navn) {
       eier = navn;
  }
  public static int finnTotal ( ) {
       return total;
   }
   public void foresporsel ( ) {
       antForesporsler ++;
       total ++;
   }
  public int finnAntForesp ( ) {
        return  antForesporsler;
   }
}
```



Antall forespørsler på Steins Toyota er 2

Antall forespørsler totalt er nå 3