UNIVERSITETET I OSLO

Institutt for informatikk

IN1010 - våren 2021

Tirsdag 12. januar

Java
Objekter og klasser

Stein Gjessing

1

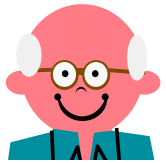
Agenda

- Rask intro til enkel innlesing og utskrift i terminalvindu (2 lysark)
  - Mer i gruppene og neste mandag

- Objekter og klasser i Java

Lesing og skriving i terminalvinduet


```
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( fornavn + ", 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 18. 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 terminalvindu OG setter variablen minInn til å referere dette objektet
adresse = minInn.nextLine();	returnerer en hel linje fra minInn som tilordnes adresse (mer senere)
navn = minInn.next();	returnerer en tekst fra minInn som tilordnes navn (mer senere)
alder = minInn.nextInt();	returnerer et heltall fra minInn som tilordnes alder (mer senere)

Google: Java API 8 Scanner

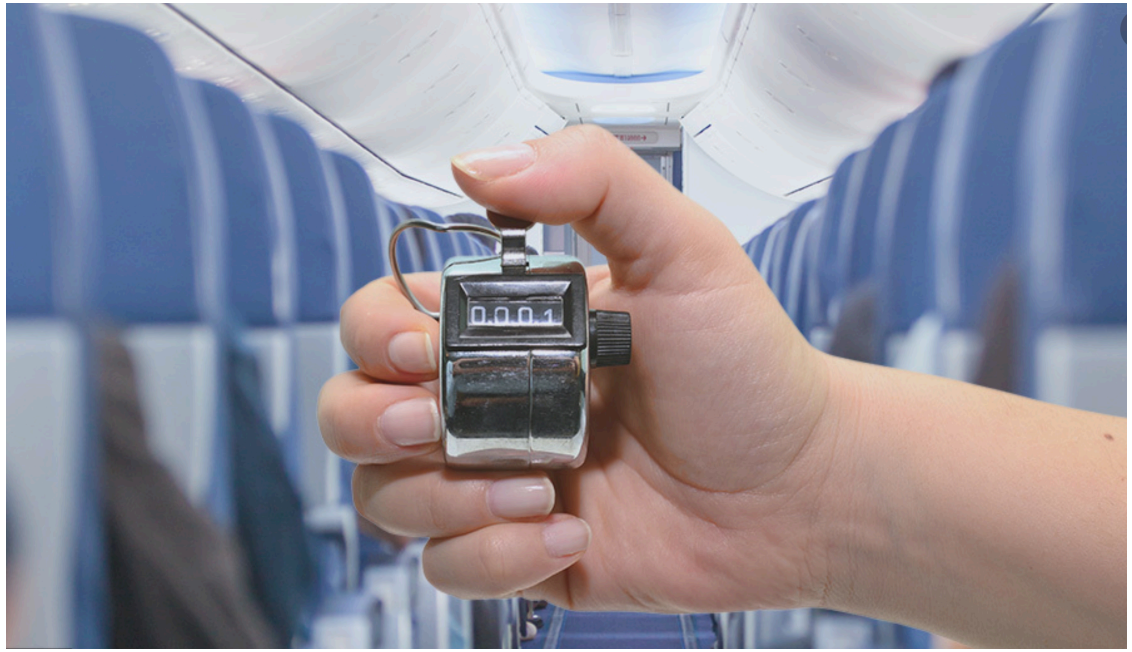
senere = neste mandag

IN1010: Objektorientert programmering

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

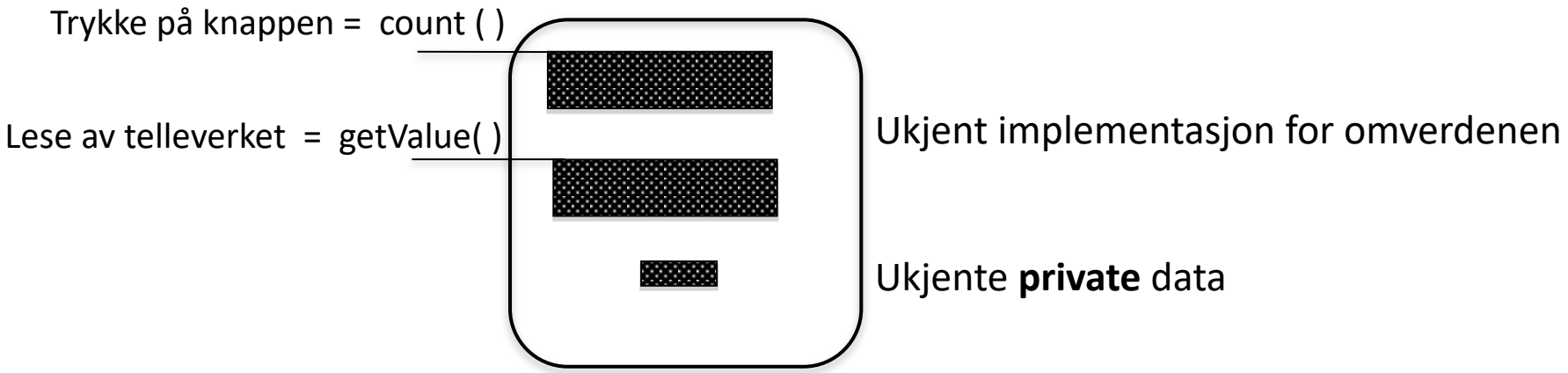
En teller (engelsk: counter)

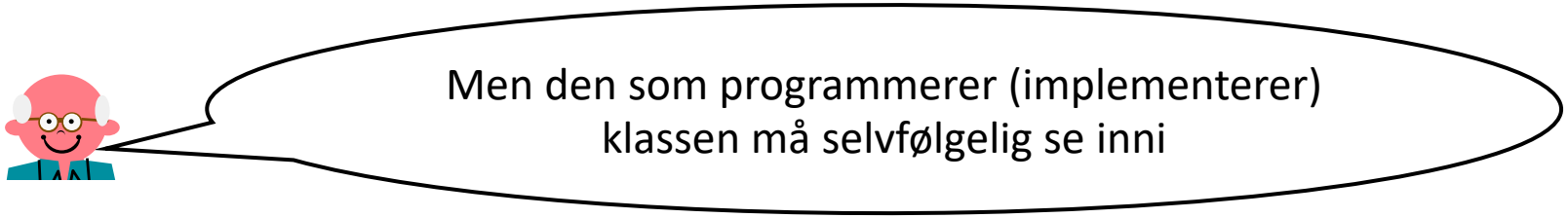
- Aller enkleste eksempel (Horstmann kap 8.2):
  - En teller (som f.eks. betjeningen på et fly bruker)
    - Tell én opp
    - Les av telleren
  - Starter på null
- Vi skal programmere en slik



Objekter i den virkelige verden modelleres av programmet inne i datamaskinen

Et objekt er en sort boks + grensesnittet mot omverdenen





En teller

- Hvordan skal count() og getValue() 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:

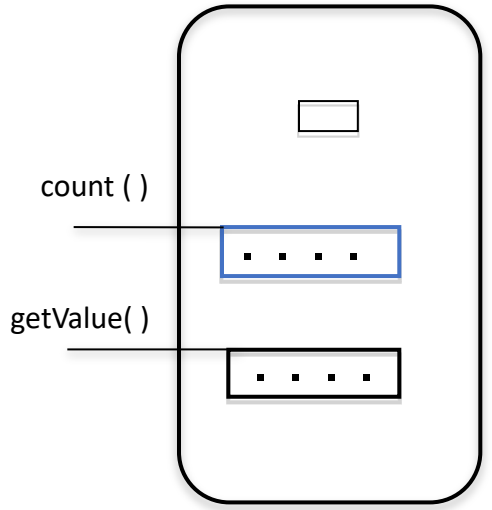
```
public void count ( )
public int    getValue ( )
```

- Hvordan skal count() og getValue() 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?

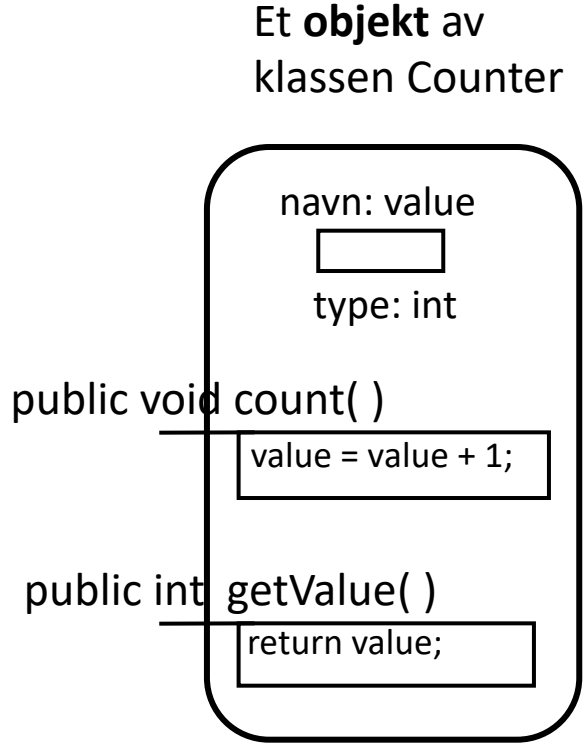
- Er det nok at
- count() teller opp denne verdien med èn og at
- getValue() leser av denne verdien



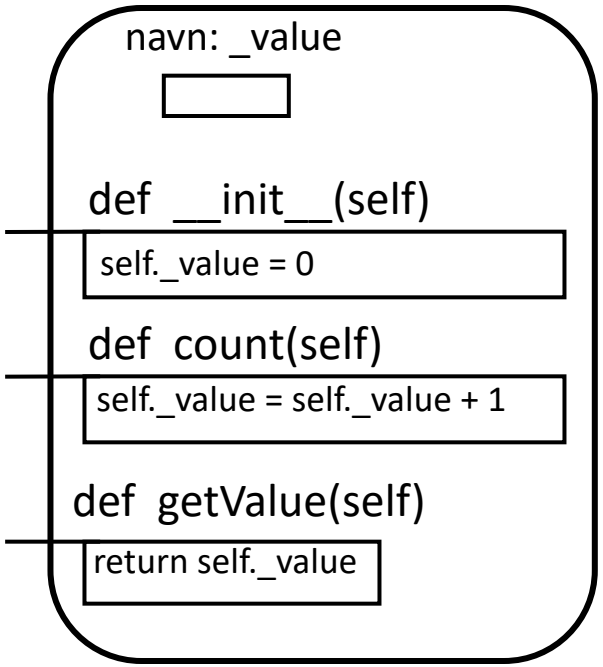
Det er dette som er å programmere !!
Men vanligvis er problemet vanskeligere !!

Et forslag til et teller-objekt

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

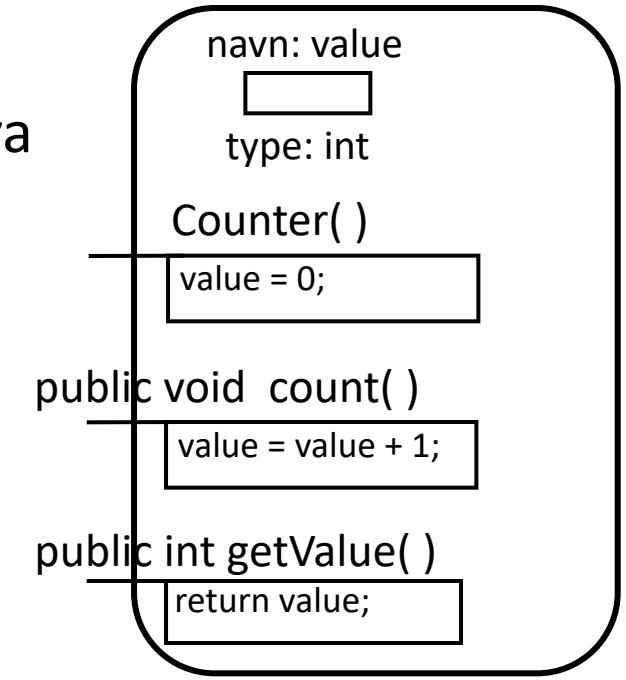


Python




```
class Counter:

   def __init__(self):
      self. _value = 0

   def count(self):
       self._value = self. _value + 1

   def getValue(self):
       return self._value
```

Java

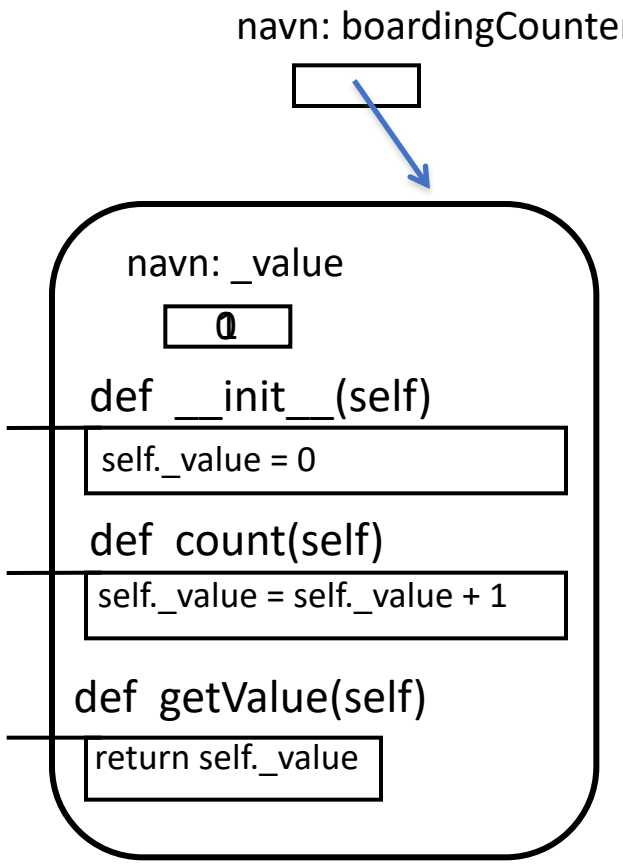



```
class Counter {
    private int value;
    public Counter() {
        value = 0;
    }
    public void count( ) {
        value = value + 1;
    }
    public int getValue( ) {
        return value;
    }
}
```

Neste skritt

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

Python

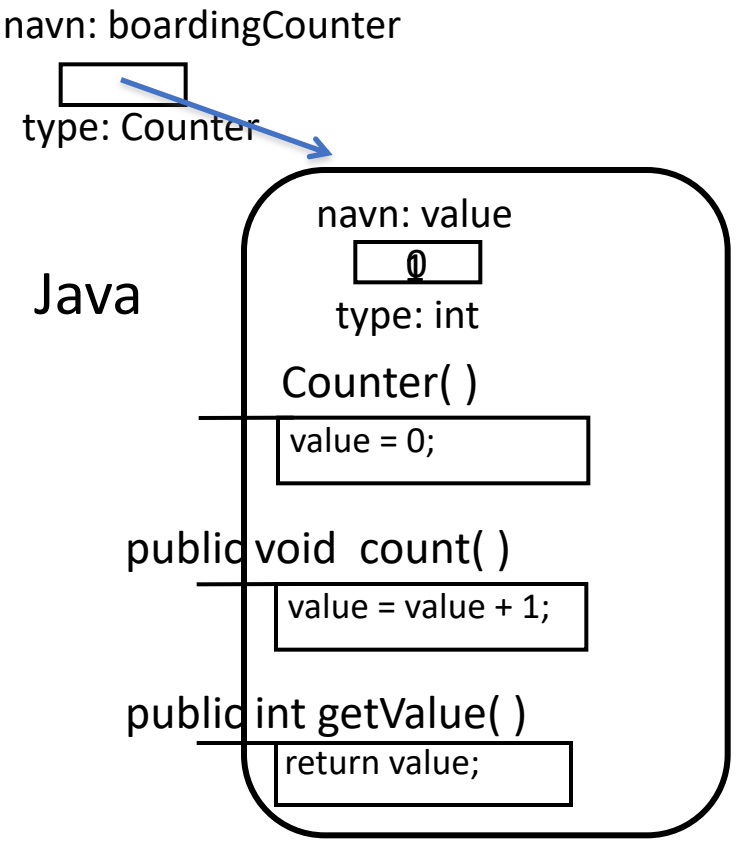



```
boardingCounter = Counter ( );

boardingCounter.count( );

tall = boardingCounter.getValue( );
```

navn: tall

1

Java




```
Counter boardingCounter = new Counter ( );

boardingCounter.count( );

int tall = boardingCounter.getValue( );
```

navn: tall

1

type: int

Fult program, to filer (i samme katalog)

```
class Counter {
    private int value;
    public Counter() {
        value = 0;
    }
    public void count( ) {
        value = value + 1;
    }
    public int getValue( ) {
        return value;
    }
}
```

```
class BrukCounter {
   public static void main (String[ ] args) {
        Counter boardingCounter = new Counter ( );
        boardingCounter.count( );
        int tall = boardingCounter.getValue( );
    }
}
```

```
> javac BrukCounter.java
> java BrukCounter
>
```

```
> javac  *.java
> java BrukCounter
>
```

Java: «static»

- En klasse er et mønster for å lage objekter

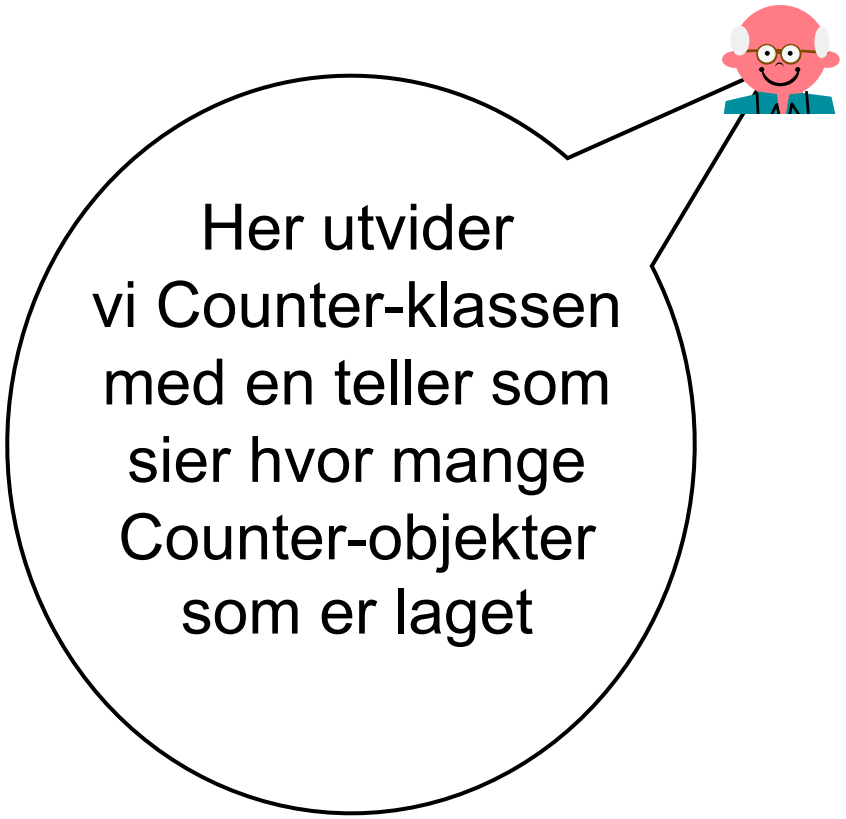
  PLUSS:

- 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

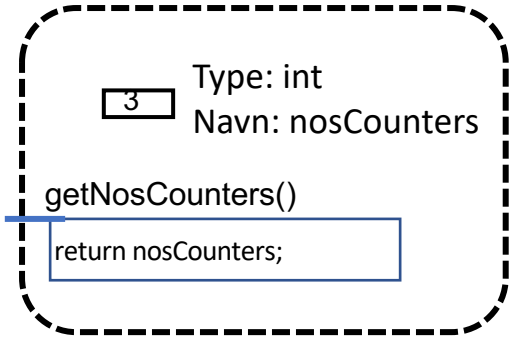

```
class Counter {
    private static int nosCounters =0;
    private int value;
    public Counter( ) {
        value = 0;
        nosCounters ++;
    }
    public static int getNosCounters ( ) {
          return nosCounters;
    }
    public void count( ) {
        value = value + 1;
    }
    public int getValue( ) {
        return value;
    }
}
```



static-egenskaper

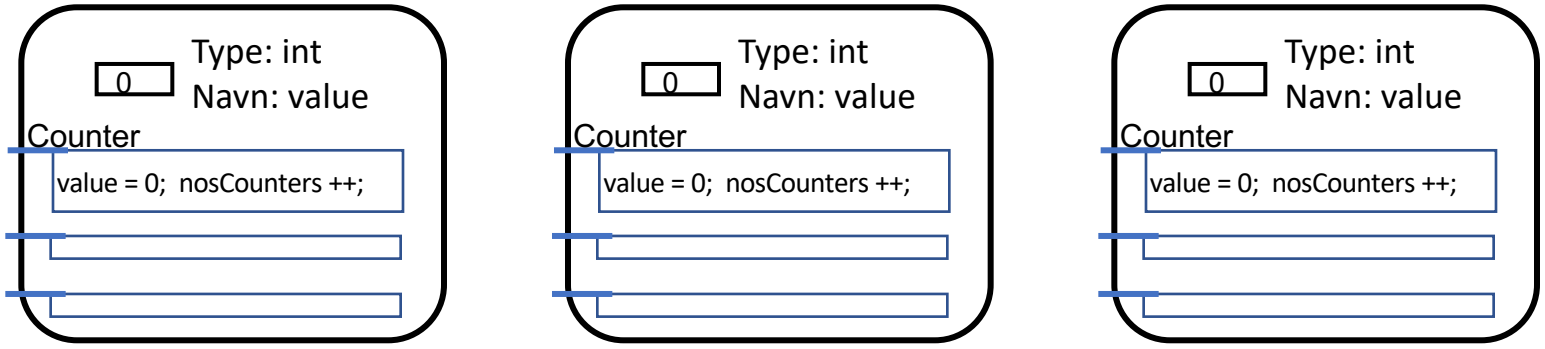

```
class Counter {
    private static int nosCounters =0;
    private int value;
    public Counter() {
        value = 0;
        nosCounters ++;
    }
    public static int getNosCounters ( ) {
        return nosCounters;
    }
    public void count( ) {
        value = value + 1;
    }
    public int getValue( ) {
        return value;
    }
}
```

Når programmet starter lages en *klasse-datastruktur* av "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 Counter() har vi 3 objekter (*objekt-datastruktur*):



Programmering er å løse problemer ved hjelp av datastrukturer

- 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

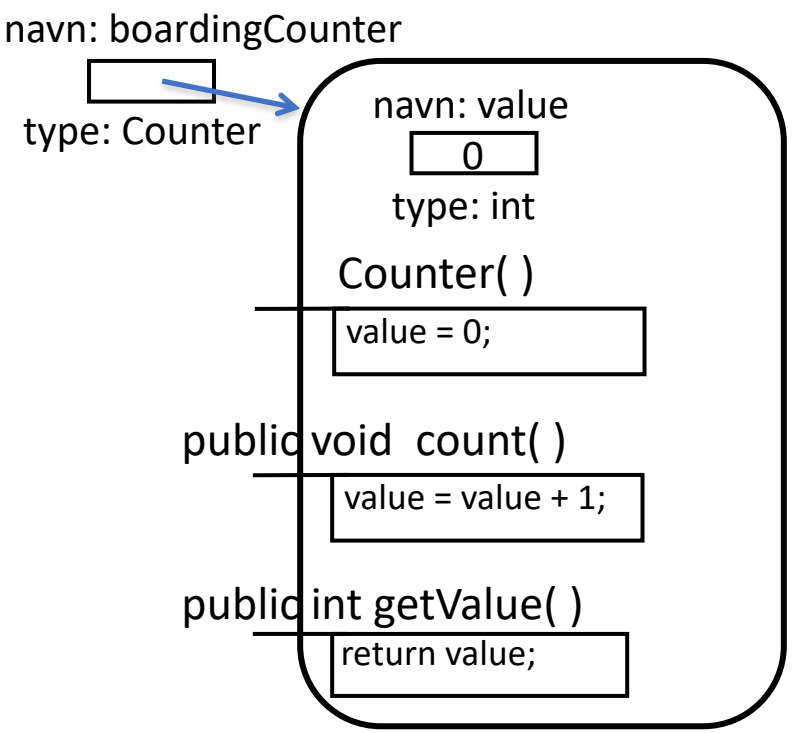
Hvor nøyaktig skal jeg tegne Java datastrukturer ?

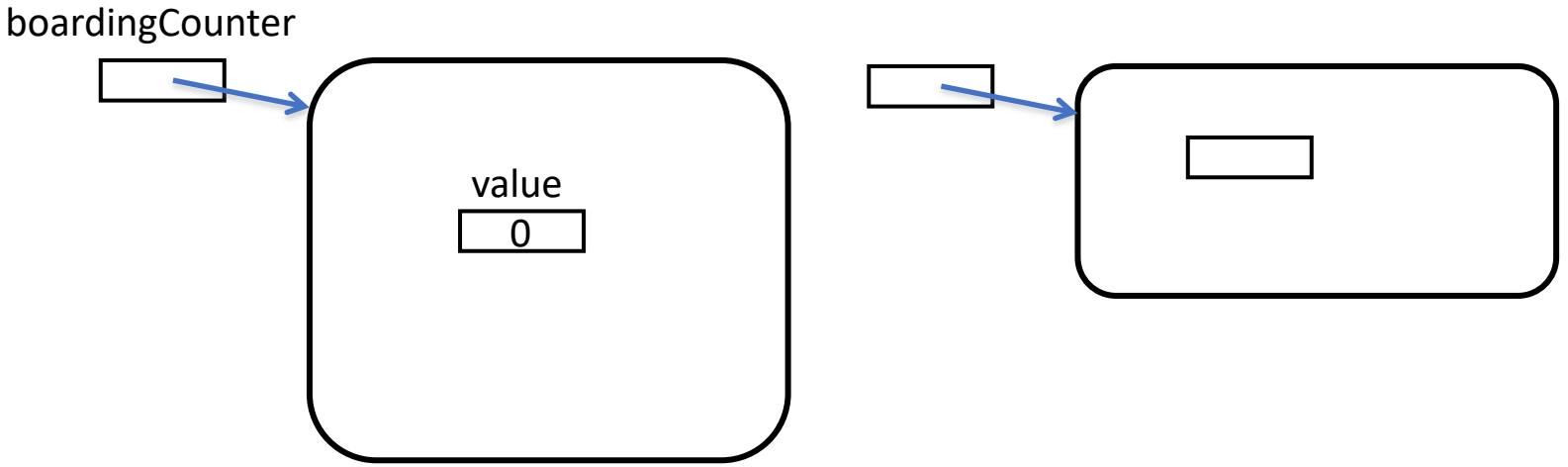
- Svar:
- Så nøyaktig som det er nødvendig for at du selv eller dem du samarbeider med skal skjønne hva som skjer med datastrukturen når programmet (algoritmen) utføres

- Du må gjerne tegne det på en annen måte enn slik vi gjør i IN1010, men da er det ikke sikkert vi andre skjønner deg

Eksempel fra teller-programmet

Tre måter å tegne det samme objektet på:





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.

Et litt større eksempel

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

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?

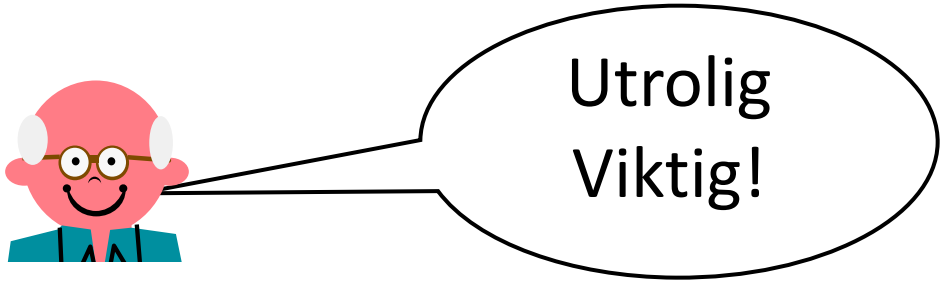
Program for salg av biler.


```
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;
   }
}
```

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 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.



BilSalg klassedatastruktur

public static void main (. . .)

navn: steinsT
type: Bil

navn: sirisO
type: Bil

navn: antallStein
type: int
2

```
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( ) );
```

Stein

navn: eier
Type: String

navn: antForesporsler
1
type: int

void foresporsel( )
antForesporsler ++;
total ++;

int finnAntForesp( )
return (antForesporsler);

Bil(String navn)
eier = navn;

Bil-objekt

Bil klassedatastruktur

Navn: total
0
Type: int

finnTotal
return total;

Siri

navn: eier
Type: String

navn: antForesporsler
0
type: int

void foresporsel( )
antForesporsler ++;
total ++;

int finnAntForesp( )
return (antForesporsler);

Bil(String navn)
eier = navn;

Bil-objekt

Antall forespørsler på Steins Toyota er 2

Antall forespørsler totalt er nå 3

I dag har du lært

- Hva et objekt er
- Hva en klasse er
- Hvordan vi visualiserer / tegner variabler og objekter
- Hva "static" betyr i Java og hvordan vi visualiserer / tegner klassedatastrukturer

- At programmering er å finne på en datastruktur og lage et program som oppretter og manipulerer denne datastrukturen