IN1010

Repetisjon Uke 1

Java for deg som kan Python

(Basert på Dag Langmyhr og Stein Gjessings presentasjoner)

24. januar, 2025

Are Digranes
aredi@ifi.uio.no

Java er ikke Python?



```
print("Jeg er en hund 🐕")
```

Et eksempel på et enkelt program i Python

Alt i Java er klasser

```
class HundeProgram {
  public static void main (String[] args) {
    System.out.println("Jeg er en hund 🐕");
  }
}
```

Samme program i Java

Programblokker

```
class HundeProgram {
  public static void main (String[] args) {
    System.out.println("Jeg er en hund 🐕");
  }
}
```



Python bruker innrykk, men i Java bruker vi krøllparenteser: { }

Programlinjer

```
class HundeProgram {
  public static void main (String[] args) {
    System.out.println("Jeg er en hund 🐕");
  }
}
```



Alle programlinjer som ikke er blokker, må i Java avsluttes med semikolon ;

main

```
class HundeProgram {
   public static void main (String[] args) {
      System.out.println("Jeg er en hund 🐕");
   }
}
```

Java starter å kjøre programmer fra den statiske metoden "main"

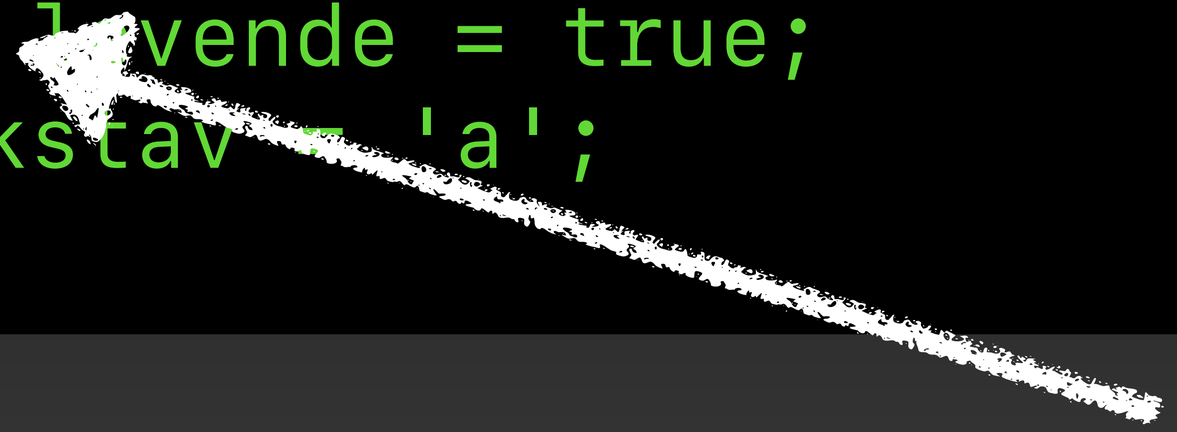
Statisk typet

```
int tall = 420;
double pi = 3.14159;
boolean levende = true;
char bokstav = 'a';
```

I Java så må alle variabler deklareres før bruk

Statisk typet

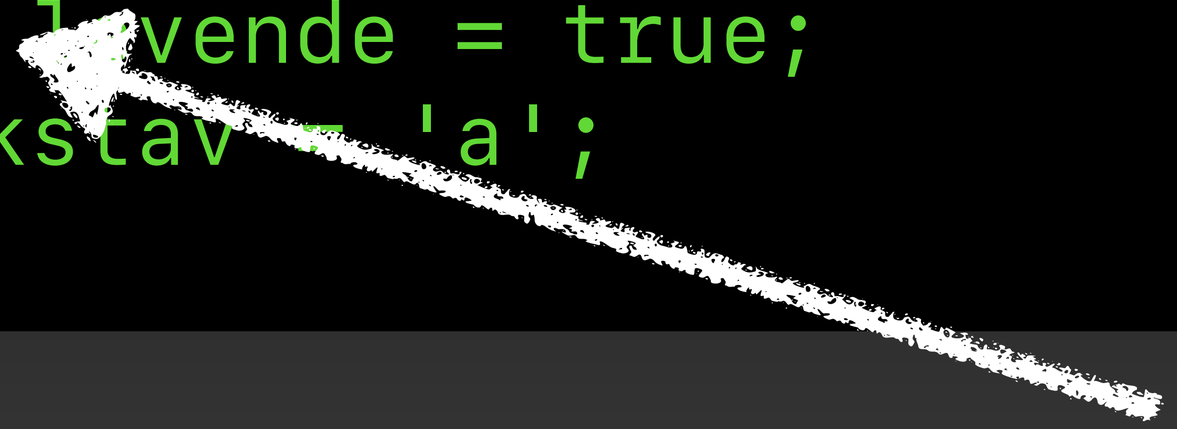
```
int tall = 420;
double pi = 3.14159;
boolean levende = true;
char bokstav = 'a';
```



Det er en del primitive datatyper, som skrives med små bokstaver

Statisk typet

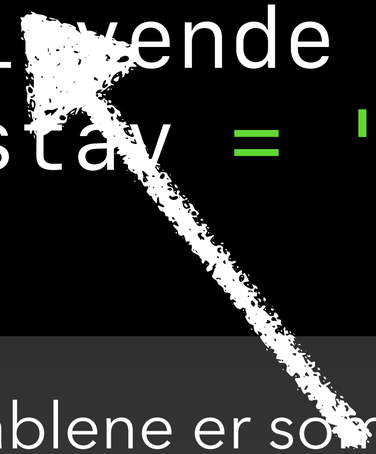
```
int tall = 420;
double pi = 3.14159;
boolean levende = true;
char bokstav = 'a';
```



int = heltall, double = flyttall, boolean = sann/usann, char = enkelttegn

Statisk typet

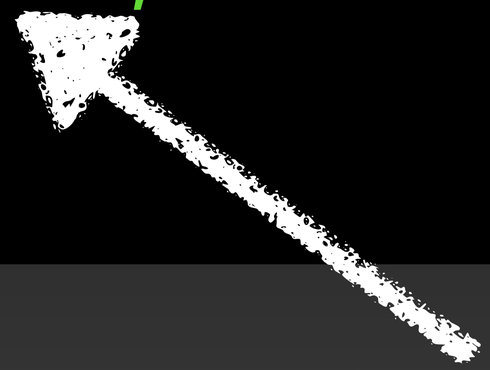
```
int tall = 420;
double pi = 3.14159;
boolean levende = true;
char bokstav = 'a';
```



Navnene på variablene er som i Python, men kan ikke senere i programmet brukes med andre datatyper

Statisk typet

```
int tall = 420;
double pi = 3.14159;
boolean levende = true;
char bokstav = 'a';
```



Verdien du gir variablene må stemme med typen de er definert som

Mellomrom og linjeskift

```
int
tall
=
420
;
```

Java bryr seg ikke om hvordan du bruker linjeskift og innrykk

Mellomrom og linjeskift

```
class
KlassenMin
{
}
```

Men det er likevel smart å bruke for at det skal være lettere å lese for oss!

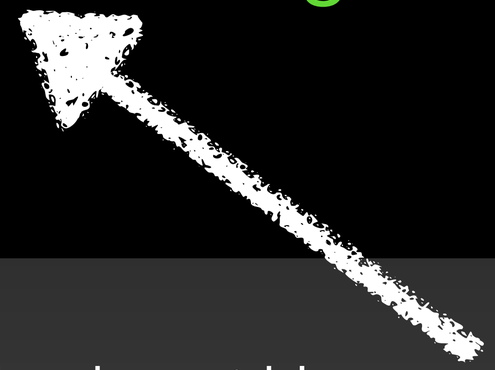
Klasser som typer

```
Integer tall = 420;
String deiligTekst = "Java er best";
```

Mange datatyper er laget som klasser

Klasser som typer

```
Integer tall = 420;
String deiligTekst = "Java er best";
```



For tekst bruker vi klassen String, det betyr at alle strenger er egne objekter

Klasser som typer

```
Integer tall = 420;
String viligTekst = "Java er best";
```



De primitive datatypene har også egne klasser som gir oss metoder for bl. a. konvertering

Variabler og objekter

I minne så er variabler av basistyper representert med verdien direkte.

Men variabler av klasse-objekter er referanser som peker på hvor i minne objektet eksisterer.

Hvis, dersom, ellers

```
if antall_hester > 100:
  print("For mange hester")
elif antall_hester < 10:
  print("For få hester")
else:
  print("Akkurat passe")
```



If, elif, og else fungerer tilsvarende som i Python. Her er et Python eksempel

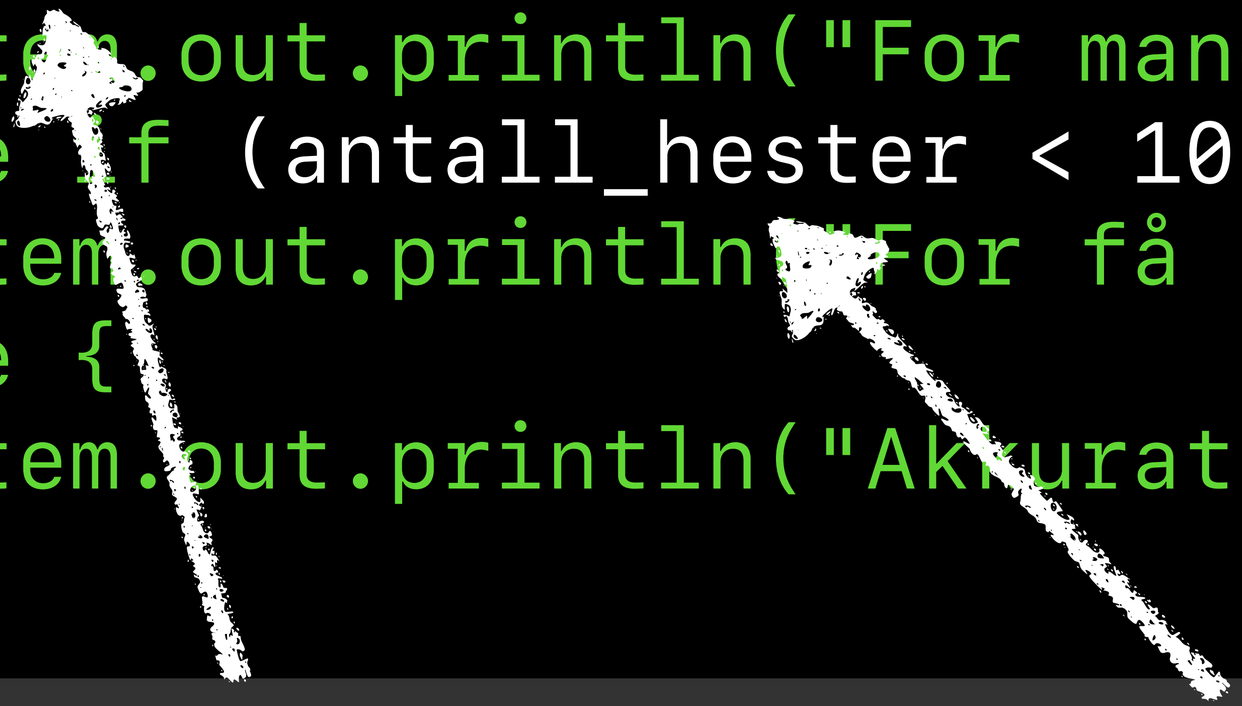
Hvis, dersom, ellers

```
if (antall_hester > 100) {
  System.out.println("For mange hester");
} else if (antall_hester < 10) {
  System.out.println("For få hester");
} else {
  System.out.println("Akkurat passe");
}
```

Og det samme i Java. Forskjellene er som følger

Hvis, dersom, ellers

```
if (antall_hester > 100) {
  System.out.println("For mange hester");
} else if (antall_hester < 10) {
  System.out.println("For få hester");
} else {
  System.out.println("Akkurat passe");
}
```



Testene må være inne i parenteser ( ) og resultatet i de må bli en *boolean*

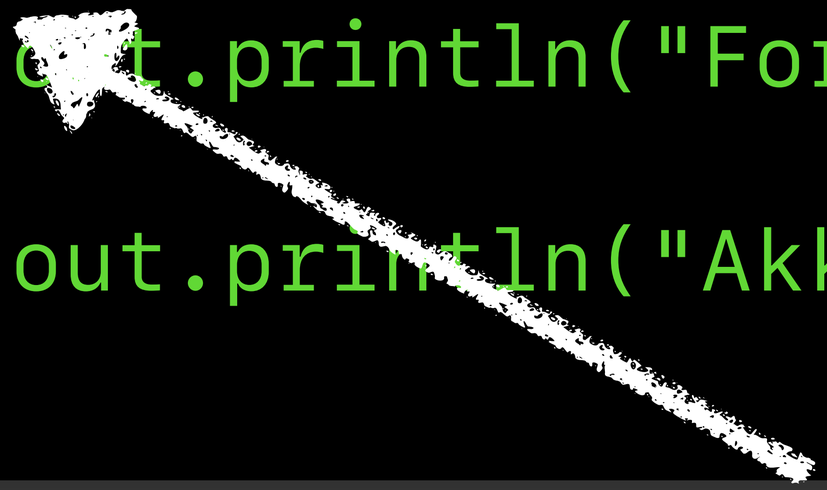
Hvis, dersom, ellers

```
if (antall_hester > 100) {
   System.out.println("For mange hester");
} else if (antall_hester < 10) {
   System.out.println("For få hester");
} else {
   System.out.println("Akkurat passe");
}
```

Programkoden som kjøres som følge av testene settes inn i blokker

Hvis, dersom, ellers

```
if (antall_hester > 100) {
  System.out.println("For mange hester");
} else if (antall_hester < 10) {
  System.out.println("For få hester");
} else {
  System.out.println("Akkurat passe");
}
```



Og *elif* skrives *else if*

Hvis, dersom, ellers

```
==  er lik (primitiv)      &&  logisk og
!=  er ikke lik            ||  logisk eller
<   mindre enn             !   ikke
>   større enn             ---------------------------
<=  mindre eller lik       x.equals(y) test om et
>=  større eller lik       objekt er lik et annet
```

Arrays

I Python så har vi mange typer samlinger, som lister, mengder, ordbøker osv.

Det har vi også i Java, men først må vi vite om en viktig basistype som vi kaller *array*.

Det er rett og slett en samling av samme datatype.

Siden de alle har samme størrelse og plasseres sammenhengende* i minne er de raske i bruk og oppslag.

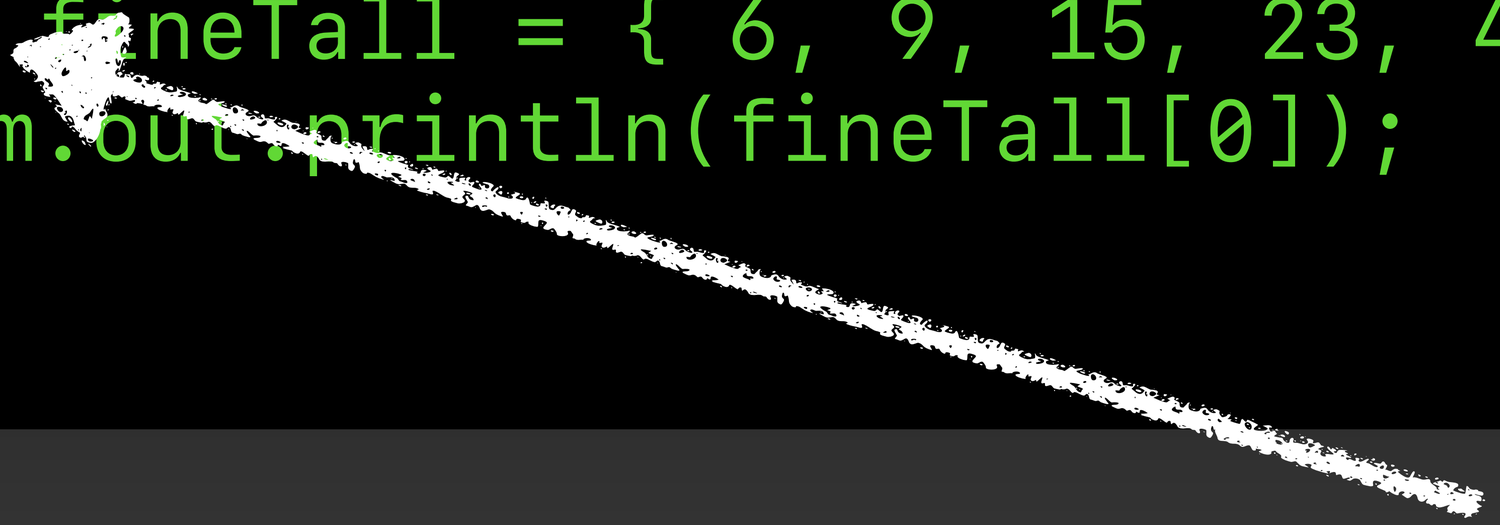
Arrays

```
int[] fineTall = { 6, 9, 15, 23, 42, 69 };
System.out.println(fineTall[0]);
```

En array av heltall i Java. Antall elementer er bestemt på forhånd.

Arrays

```
int[] fineTall = { 6, 9, 15, 23, 42, 69 };
System.out.println(fineTall[0]);
```



Vi bruker hakeparenteser [ ] etter typen for å lage en array.

Arrays

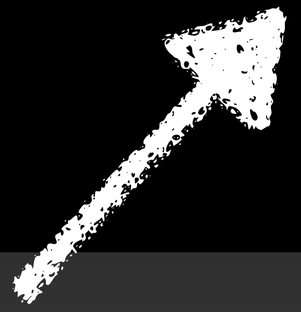
```
int[] fineTall = { 6, 9, 15, 23, 42, 69 };
System.out.println(fineTall[0]);
```



Elementene legges inn ved hjelp av krøllparenteser { } og komma , mellom verdiene

Arrays

```
int[] fineTall = { 6, 9, 15, 23, 42, 69 };
System.out.println(fineTall[0]);
```



Som i Python så begynner vi på 0, så her skriver vi ut første element

Arrays

```
String[] hunderaser = {
  "Puddel",
  "Dachshund"
};
System.out.println(hunderaser[0]);
```

Vi kan også lage arrays av objekter, selve arrayet er da referanser til objektene

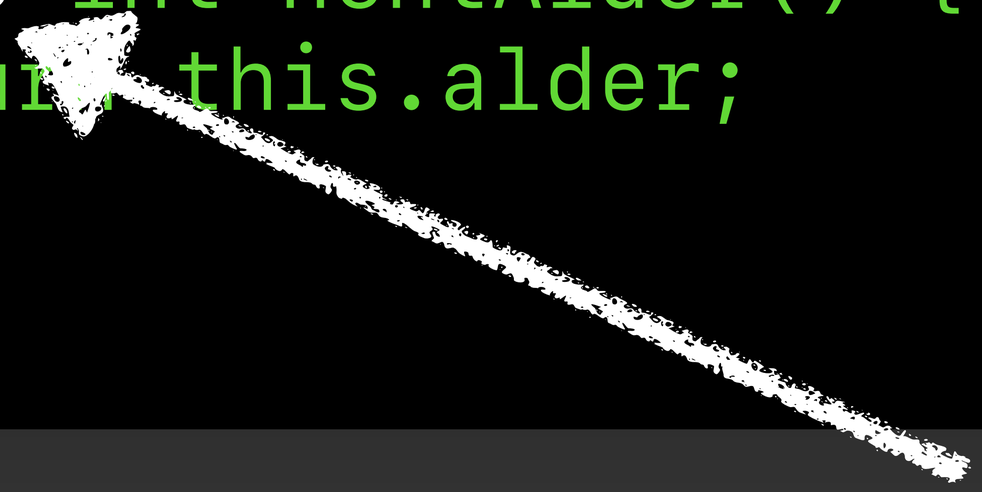
Metoder i Java

```
public int hentAlder() {
  return this.alder;
}
```

Metoder i Java har mye likt med Python, men noen viktige forskjeller

Metoder i Java

```
public int hentAlder() {
    return this.alder;
}
```



Vi må bestemme om metoden kan brukes utenfor klassen

Metoder i Java

```
public int hentAlder() {
   return this.alder;
}
```



Vi må også deklarere hvilken datatype som metoden skal returnere

Metoder i Java

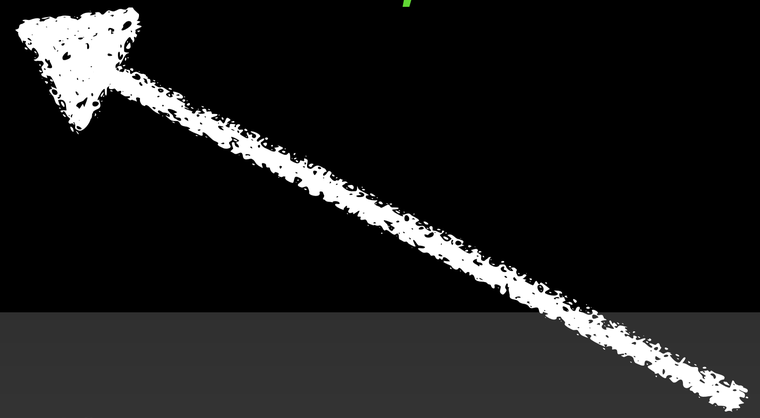


```
public int hentAlder() {
   return this.alder;
}
```

Datatypene til *return* må matche den som er deklarert

Metoder i Java

```
public int hentAlder() {
  return this.alder;
}
```



I Python så brukte vi *self*, i Java som bruker vi *this* for å referere objektet vi "er" i

Konstruktør

```
class Hund {
  private String navn;
  public Hund(String navn) {
    this.navn = navn;
  }
}
```

Hver gang et objekt opprettes kjøres konstruktøren. Hvis klassen ikke har en konstruktør, så lager Java en for deg.

Konstruktør



```
class Hund {
   private String navn;
   public Hund(String navn) {
      this.navn = navn;
   }
}
```

Navnet er alltid det samme som klassen. Dette er ikke en metode og har *ikke* en returverdi. Obs! Hvis du legger til en returverdi, så fungerer den ikke som en konstruktør.

static

I klasser har vi i blant behov for metoder og variabler som er felles for alle objektene.

Og siden alt er klasser i Java, så må vi også kunne utføre ting som ikke allerede er et objekt.

Til det har vi *static*.

Statiske metoder

```
class HundeProgram {
  public static void main (String[] args) {
    System.out.println("Jeg er en hund 🐕");
  }
}
```

Alle Java-programmerere drømmer om public static void main

Statiske metoder

```
class HundeProgram {
  public static void main (String[] args) {
    System.out.println("Jeg er en hund 🐕");
  }
}
```

Men det er en vanlig metode, som er static siden vi ikke kan lage et objekt før programmet kjører.

Utskrift til terminalen

```
System.out.println("Her er hunden!");

System.out.print("Hund");
System.out.printf("Jeg heter %s\n", navn);
System.err.println("Feilmelding: Feil");
```

System-klassen har metoder for utskrift

Utskrift til terminalen

```
System.out.println("Her er hunden!");

System.out.print("Hund");
System.out.printf("Jeg heter %s\n", navn);
System.err.println("Feilmelding: Feil");
```



For å skrive ut en vanlig linje så bruker vi denne

Utskrift til terminalen

```
System.out.println("Her er hunden!");

System.out.print("Hund");
System.out.printf("Jeg heter %s\n", navn);
System.err.println("Feilmelding: Feil");
```



Hvis vi vil skrive ut noe uten å legge til linjeskift kan vi bruke denne

Utskrift til terminalen

```
System.out.println("Her er hunden!");

System.out.print("Hund");
System.out.printf("Jeg heter %s\n", navn);
System.err.println("Feilmelding: Feil");
```

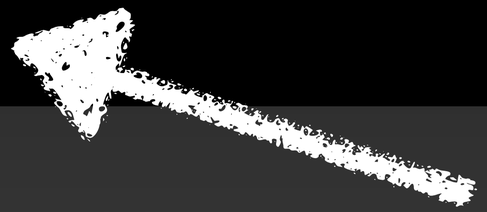


Det finnes også en som ligner litt på Python f-string, som C/C++-folk kjenner igjen

Utskrift til terminalen

```
System.out.println("Her er hunden!");

System.out.print("Hund");
System.out.printf("Jeg heter %s\n", navn);
System.err.println("Feilmelding: Feil");
```



For feilmeldinger så bør man bruke denne