IN1010 uke 4

Gruppe 2

Agenda

- Interface
  - Hvorfor interface? - Eksempel
  - Pekere av typen interface
  - Interface vs abstract class vs class
- Mer om exceptions
- Menti quiz om arv & subklasser del I, II & III 🏆
- Repetisjon av static for innlevering 1
- Jobbe med oppgaver/trix/oblig!

Interface

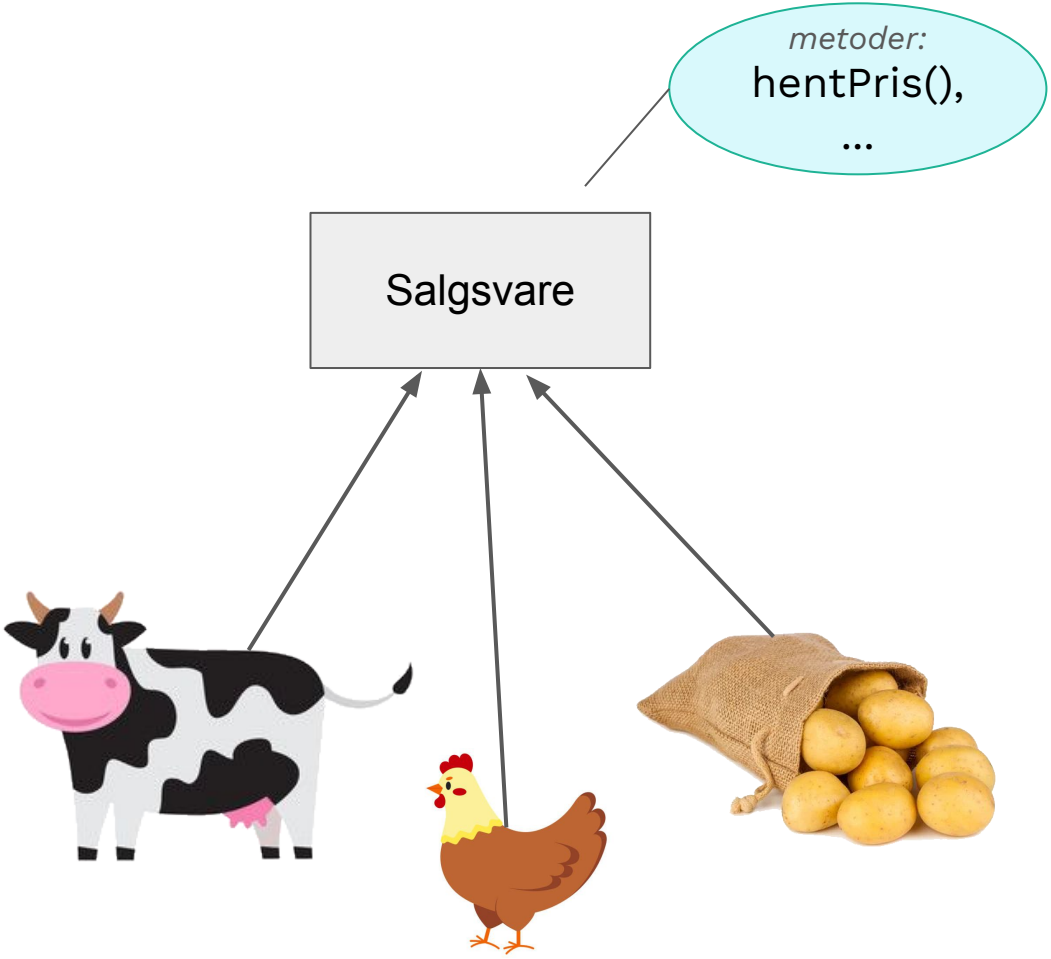
Interface

- Hva/hvorfor?
- Forskjellen på interface/abstrakt klasse/klasse?

Hvorfor interface?

Eksempel: Bondegårdsystem

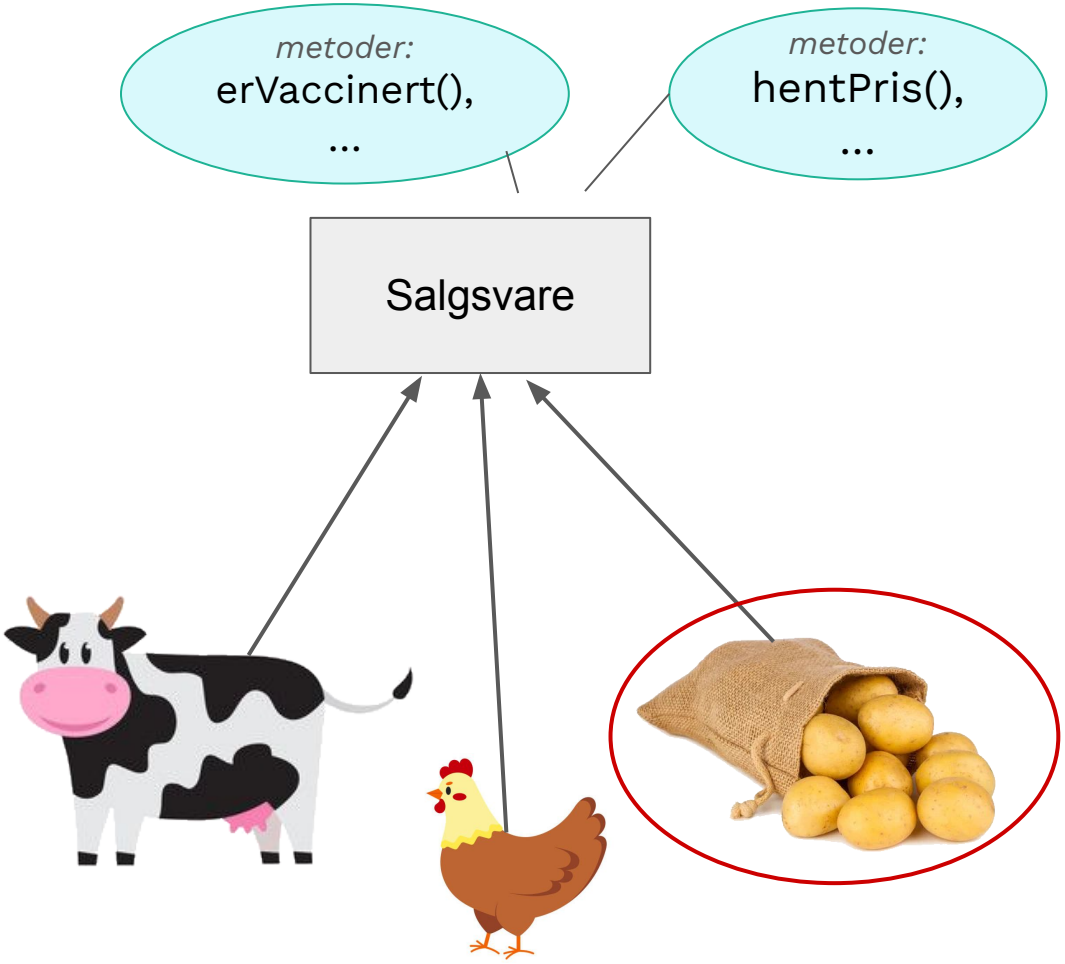
- Både selge dyr og sjekke helsen



Hvorfor interface?

Eksempel: Bondegårdsystem

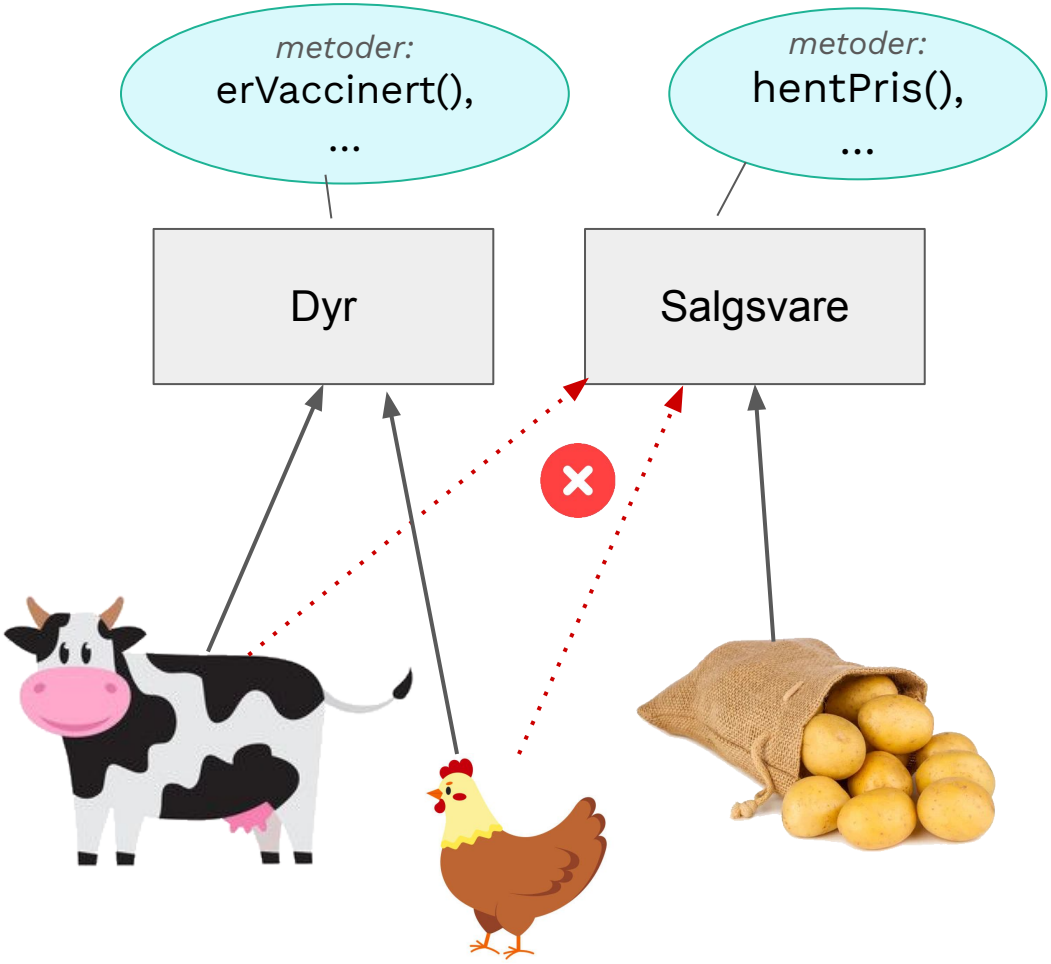
- Både selge dyr og sjekke helsen
  → legg til dette i superklasse??



Hvorfor interface?

Eksempel: Bondegårdsystem

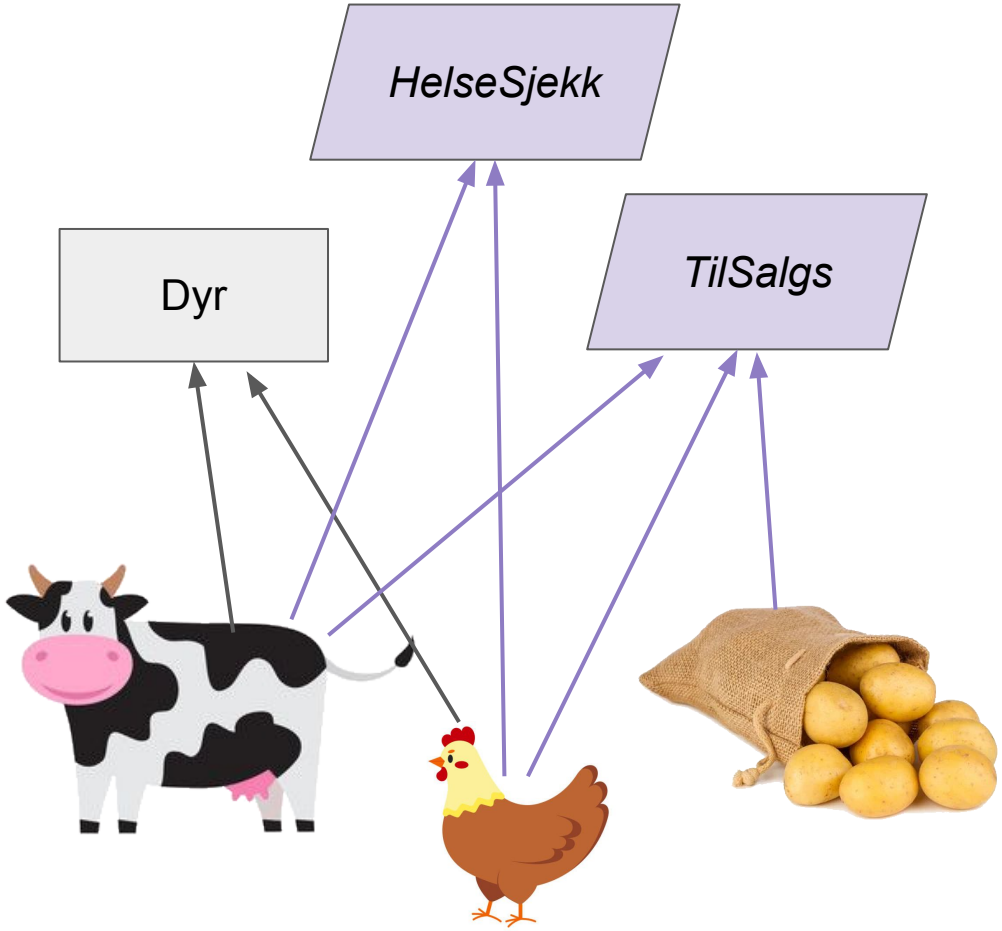
- Både selge dyr og sjekke helsen
  → Arve fra en til klasse?
  - kan bare arve fra én



Hvorfor interface?

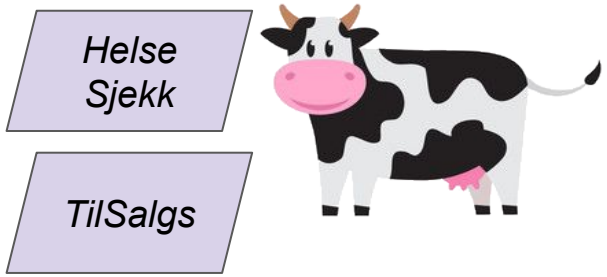
Eksempel: Bondegårdsystem

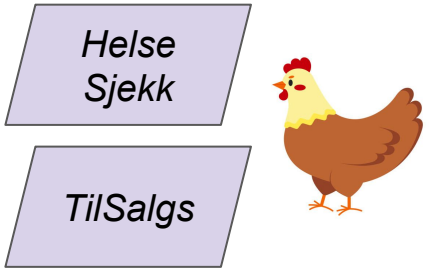
- Både selge dyr og sjekke helsen
  → løsning: interface



Hvorfor interface?

- En klasse kan implementere flere interface
- Spesifiserer rollen/grensesnittet til klassen
  - “Utenifra”: Vet hva klassen gjør uten å måtte se implementasjonen
  - Programmereren: Vet hva klassen må kunne gjøre men har frihet til å implementere det (skrive koden) som man vil







```java
class Poteter implements TilSalgs{
    private double vekt;
    private String dato;
    private int kgPris = 8;

    public Poteter(double vekt, String dato) {
        this.vekt = vekt;
        this.dato = dato;
    }

    @Override
    public int hentPris() {
        return (int) Math.round(vekt * kgPris) ;
    }

    @Override
    public void settPris(int nyPris) {
        kgPris = nyPris;
    }
```

```java
interface TilSalgs {
    int hentPris();
    void settPris(int nyPris);
}
```



```java
class Ku extends Dyr implements TilSalgs, HelseSjekk {
    private String navn;
    private double vekt;
    private int pris = 11000;
    private boolean vaksinert = false;

    public Ku(String navn, int alder, double vekt) {
        super(alder);
        this.navn = navn;
        this.vekt = vekt;
    }

    @Override
    public int hentPris() {
        return pris;
    }

    @Override
    public void settPris(int nyPris) {
        pris = nyPris;
    }
```

```java
interface TilSalgs {
    int hentPris();
    void settPris(int nyPris);
}
```

- Ulik implementering!



```java
@Override
public boolean normalVekt() {
    return vekt > 500 && vekt < 750;
}

@Override
public boolean erVaksinert() {
   return vaksinert;
}

@Override
public void giVaksine() {
    if (!erVaksinert()) {
        vaksinert = true;
        System.out.println("Vaksinerte kuen" + navn);
    }
}
```

```java
interface HelseSjekk {
    boolean normalVekt();
    boolean erVaksinert();
    void giVaksine();
}
```



```java
class BondegaardSystem {
    Run | Debug
    public static void main(String[] args) {
        Poteter potet1 = new Poteter(2.5, "08.08.2022");
        Poteter potet2 = new Poteter(2.3, "08.08.2022");
        Ku ku = new Ku("Molly", 3, 550);

        TilSalgs[] salgListe = {potet1, potet2, ku};

        for (TilSalgs vare : salgListe) {
            System.out.println(vare + ": " + vare.hentPris() + "kr");
        }
    }
}
```

```
Poteter 2.5 kg: 20kr
Poteter 2.3 kg: 18kr
Kuen Molly, 3 aar: 11000kr
```

Vi kan gjøre dette fordi vi har peker av type TilSalgs!

```java
class Hund {
    public void bjeff() {
        System.out.println("Voff");
    }
}
```

```java
class GoldenRetriever extends Hund implements TilSalgs {
    private int pris = 8000;
    public void vaerGlad() {
        System.out.println("Jeg er saaa glad!");
    }

    @Override
    public int hentPris() {
        return pris;
    }

    @Override
    public void settPris(int nyPris) {
        pris = nyPris;
    }
}
```

Hva skjer hvis vi kjører hovedprogrammet?

```java
public static void main(String[] args) {
    GoldenRetriever r = new GoldenRetriever();
    TilSalgs salgsR = new GoldenRetriever();

    System.out.println(r.hentPris());
    System.out.println(salgsR.hentPris());

    r.bjeff();
    salgsR.bjeff();

    r.vaerGlad();
    salgsR.vaerGlad();
}
```

```java
class Hund {
    public void bjeff() {
        System.out.println("Voff");
    }
}
```

Hva skjer hvis vi kjører hovedprogrammet?

```java
class GoldenRetriever extends Hund implements TilSalgs {
    private int pris = 8000;
    public void vaerGlad() {
        System.out.println("Jeg er saaa glad!");
    }

    @Override
    public int hentPris() {
        return pris;
    }

    @Override
    public void settPris(int nyPris) {
        pris = nyPris;
    }
}
```

⊗ The method bjeff() is undefined for the type TilSalgs Java

⊗ The method vaerGlad() is undefined for the type TilSalgs

```java
public static void main(String[] args) {
    GoldenRetriever r = new GoldenRetriever();
    TilSalgs salgsR = new GoldenRetriever();

    System.out.println(r.hentPris());
    System.out.println(salgsR.hentPris());

    r.bjeff();
    salgsR.bjeff();

    r.vaerGlad();
    salgsR.vaerGlad();
}
```

	Interface	Abstrakt klasse	Klasse
Skrives	`interface X`	`abstract class X`	`class X`
Konstruktør	✗	✓	✓
Instansvariabler	✗	✓	✓
Lage objekt av	✗	✗	✓
Implementere metoder (kode hva de skal gjøre)	✗	✓	✓
Abstrakte metoder	✓	✓	✗
En klasse kan arve fra flere…	✓	✗	✗

Interface vs abstract class vs class: koden

- instansvariabler
- konstruktør
- abstrakte metoder
- implementation av metoder

```java
interface TilSalgs {
    int hentPris();
    void settPris(int nyPris);
}
```

```java
abstract class Dyr {
    protected int alder;

    public Dyr(int alder) {
        this.alder = alder;
    }

    abstract void lagLyd();

    public int hentAlder() {
        return alder;
    }

}
```

```java
class Bil {
    private String regNr;

    public Bil(String regNr, String merke) {
        this.regNr = regNr;
    }

    public String hentRegNr() {
        return regNr;
    }
}
```

Mer om exceptions

(Men først litt repetisjon!)

Exceptions Repetisjon

- Hva skjer hvis filen vi vil lese fra ikke finnes?

```
public static void main(String[] args) {
    Scanner fil = new Scanner(new File(pathname: "minfil.txt"));
}
```

Exceptions - *throws*

Repetisjon

```java
public static void main(String[] args) throws FileNotFoundException {
    Scanner fil = new Scanner(new File(pathname: "minfil.txt"));
}
```

→ Kan kjøre filen, men får “stygg” feilmelding hvis ikke filen finnes:

```
Exception in thread "main" java.io.FileNotFoundException: minfil.txt (The system cann
ot find the file specified)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:216)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.util.Scanner.<init>(Scanner.java:639)
        at Matbutikk.main(Matbutikk.java:7)
```

Exceptions - *try, catch*

Repetisjon

```
Scanner fil = null;
try {
    fil = new Scanner(new File(pathname: "minfil.txt"));
} catch (Exception e) {
    System.out.println(x: "Kunne ikke lese filen minfil.txt!");
    System.exit(status: 1);
}
```



```
Kunne ikke lese filen minfil.txt!
```

```java
public static void main(String[] args) {
    Scanner fil = null;
    int[] tall = new int[3];
    try {
        fil = new Scanner(new File("minfil.txt"));
        tall[4] = 1;
    } catch (Exception e) {
        System.out.println("Kunne ikke lese filen!");
        System.exit(1);
    }

    while (fil.hasNextLine()) {
        System.out.println(fil.nextLine());
    }
}
```

minfil.txt

```
1    linje 1
2    linje 2
3    linje 3
```

Hva skjer når vi kjører programmet?

```java
public static void main(String[] args) {
    Scanner fil = null;
    int[] tall = new int[3];
    try {
        fil = new Scanner(new File("minfil.txt"));
        tall[4] = 1;
    } catch (Exception e) {
        System.out.println("Kunne ikke lese filen!");
        System.exit(1);
    }

    while (fil.hasNextLine()) {
        System.out.println(fil.nextLine());
    }
}
```

minfil.txt

```
1    linje 1
2    linje 2
3    linje 3
```

Hva skjer når vi kjører programmet?

```
Kunne ikke lese filen!
```

```java
import java.io.FileNotFoundException;
```

```java
public static void main(String[] args) {
    Scanner fil = null;
    int[] tall = new int[3];
    try {
        fil = new Scanner(new File("minfil.txt"));
        tall[4] = 1;
    } catch (FileNotFoundException e) {
        System.out.println("Kunne ikke lese filen!");
        System.exit(1);
    }

    while (fil.hasNextLine()) {
        System.out.println(fil.nextLine());
```

minfil.txt

```
1   linje 1
2   linje 2
3   linje 3
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
 Index 4 out of bounds for length 3
```

```java
public static void main(String[] args) {
    Scanner fil = null;
    int[] tall = new int[3];
    try {
        fil = new Scanner(new File("minfil.txt"));
        tall[4] = 1;
    } catch (FileNotFoundException e) {
        System.out.println("Kunne ikke lese filen!");
        System.exit(1);
    } catch (IndexOutOfBoundsException e) {
        System.out.println("Kunne ikke sette inn i tall-array!");
        System.exit(1);
    }

    while (fil.hasNextLine()) {
        System.out.println(fil.nextLine());
    }
}
```

minfil.txt

```
1   linje 1
2   linje 2
3   linje 3
```

Altså: Ved try/catch, catch helst en *spesifikk* exception og ikke alle (*Exception*)

```
Kunne ikke sette inn i tall-array!
```

Menti quiz 🏅

Om arv og subklasser del I, II & III
(uke 2-4)

Gå til www.mentimeter.com

Repetisjon: static

Relevant for innlevering 1

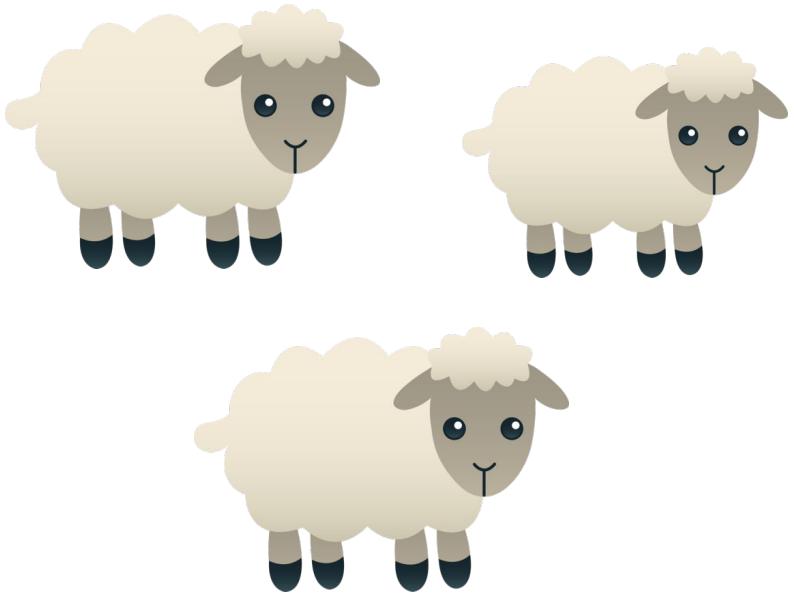
Static - eksempel

Uten static

```
class Sau {
    private int sauTeller = 0;
    private int sauNr;
    private String navn;
    public Sau(String navn) {
        this.navn = navn;
        sauTeller++;
        sauNr = sauTeller;
    }

    public int hentTeller() {
        return sauTeller;
    }
}
```

- Hva er verdien til sauTeller etter at vi har laget 3 Sau-objekter?



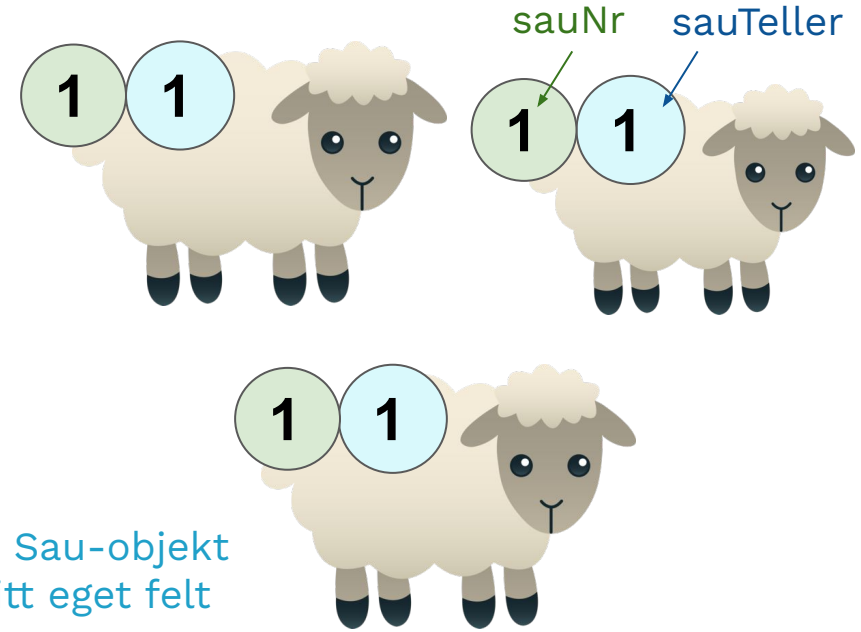
Static - eksempel

Uten static

```
class Sau {
    private int sauTeller = 0;
    private int sauNr;
    private String navn;
    public Sau(String navn) {
        this.navn = navn;
        sauTeller++;
        sauNr = sauTeller;
    }

    public int hentTeller() {
        return sauTeller;
    }
}
```

- Felter *(instansvariabler)* - *sauTeller* blir aldri mer enn 1



Hvert Sau-objekt har sitt eget felt

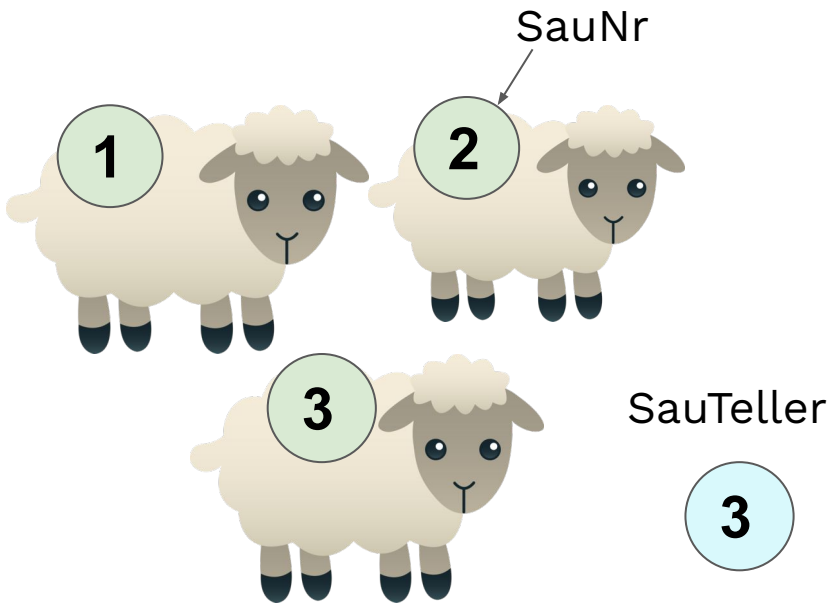
Static - eksempel

Med static

```
class Sau2 {
    private static int sauTeller = 0;
    private int sauNr;
    private String navn;
    public Sau2(String navn) {
        this.navn = navn;
        sauTeller++;
        sauNr = sauTeller;
    }

    public static int hentTeller() {
        return sauTeller;
    }
}
```

- static variabel/metode tilhører *klassen*
- Kan også telle 0 sauer



Jobb med oppgaver!

IN1010 Emnesiden → Grupper →
Gruppe 2 → Uke4

Husk å ta pause ☕