

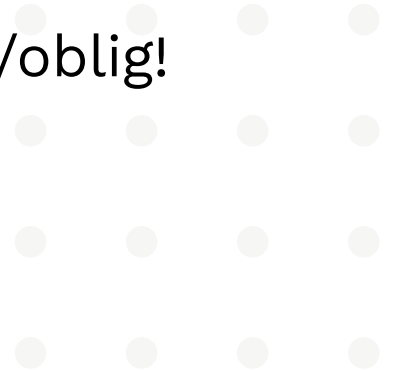


IN1010 uke 5

Gruppe 7

Agenda

- Interface
 - Interface vs abstract class vs class
 - Hvorfor interface? - Eksempel
 - Pekere av typen interface
- Invarianter
- Jobbe med oppgaver/trix/oblig!





















Interface





Interface

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

| | Interface | Abstrakt klasse | Klasse |
|---|--|--|--|
| Skrives | <code>interface X</code> | <code>abstract class X</code> | <code>class X</code> |
| Konstruktør |  |  |  |
| Instansvariabler |  |  |  |
| Lage objekt av |  |  |  |
| Implementere metoder (kode hva de skal gjøre) |  |  |  |
| Abstrakte metoder |  |  |  |
| En klasse kan arve fra flere... |  |  |  |

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

```
abstract class Dyr {  
    protected int alder;  
  
    public Dyr(int alder) {  
        this.alder = alder;  
    }  
  
    abstract void lagLyd();  
  
    public int hentAlder() {  
        return alder;  
    }  
}
```

Interface vs abstract class vs class: koden

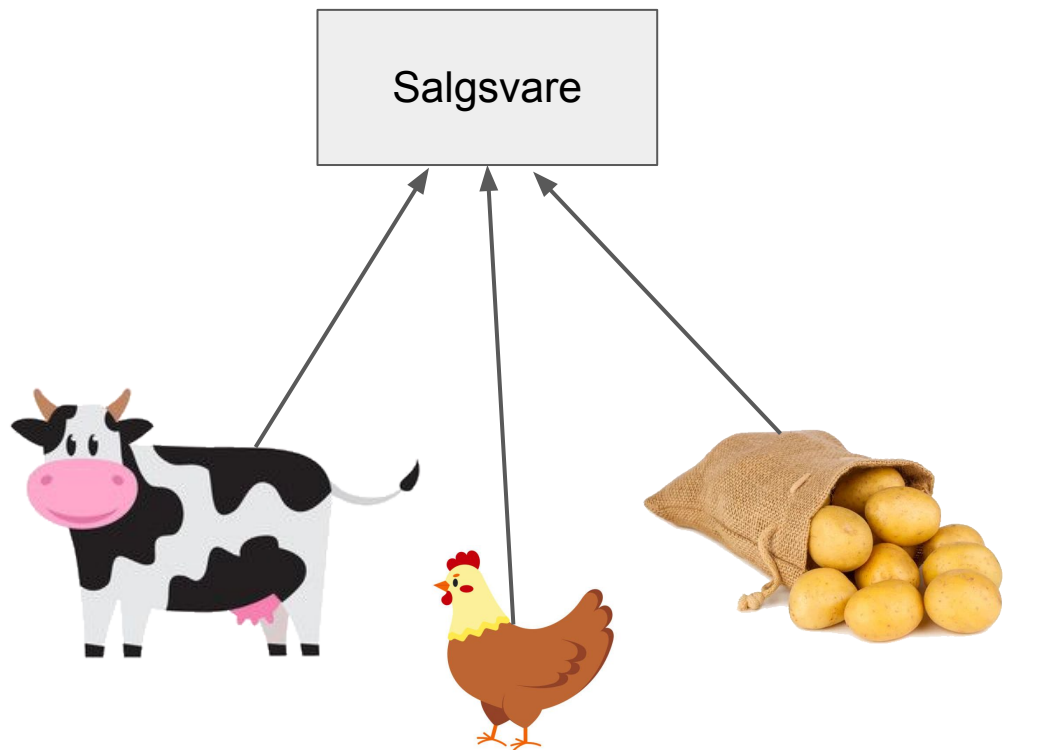
- instansvariabler
- konstruktør
- abstrakte metoder
- implementation av metoder

```
class Bil {  
    private String regNr;  
  
    public Bil(String regNr, String merke) {  
        this.regNr = regNr;  
    }  
  
    public String hentRegNr() {  
        return regNr;  
    }  
}
```

Hvorfor interface?

Eksempel: Bondegårdsystem

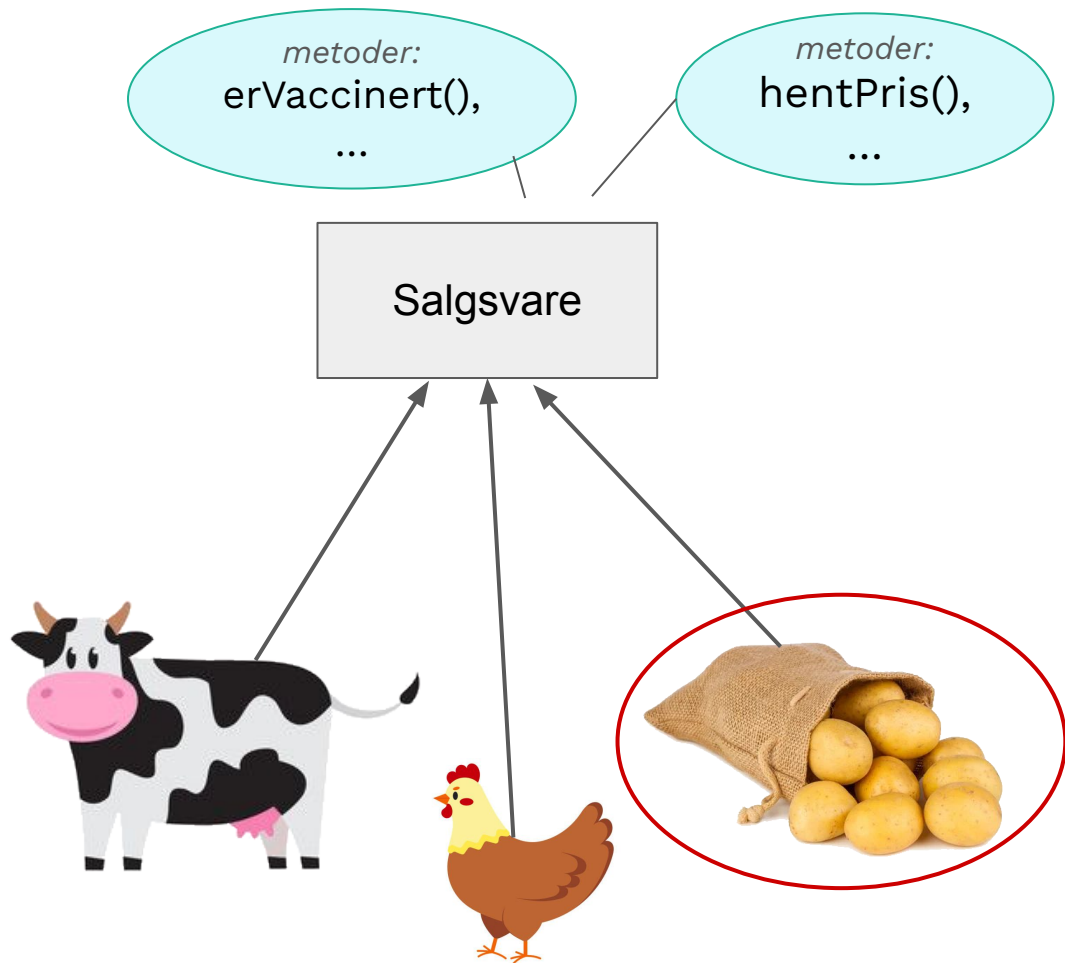
- Både selge dyr og sjekke helsen



Hvorfor interface?

Eksempel: Bondegårdsystem

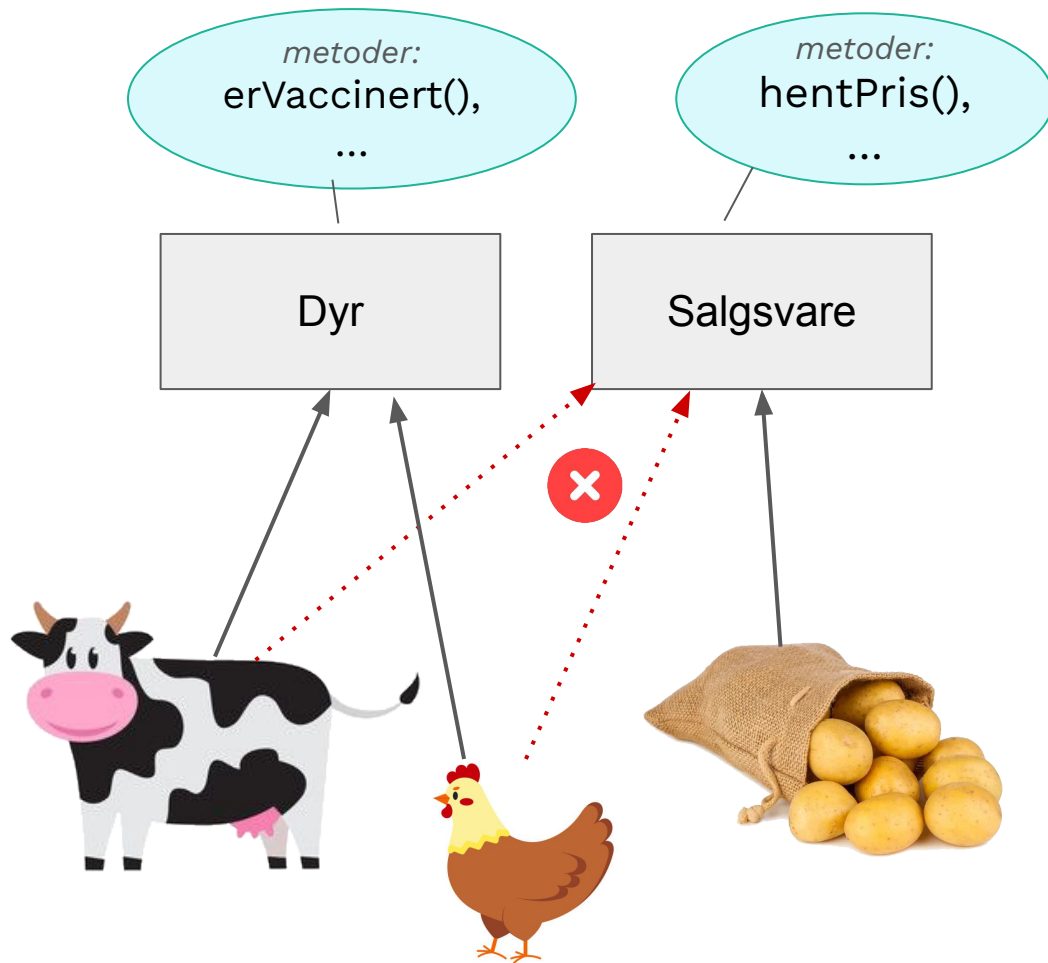
- Både selge dyr og sjekke helsen
→ legg til dette i superklasse??



Hvorfor interface?

Eksempel: Bondegårdsystem

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

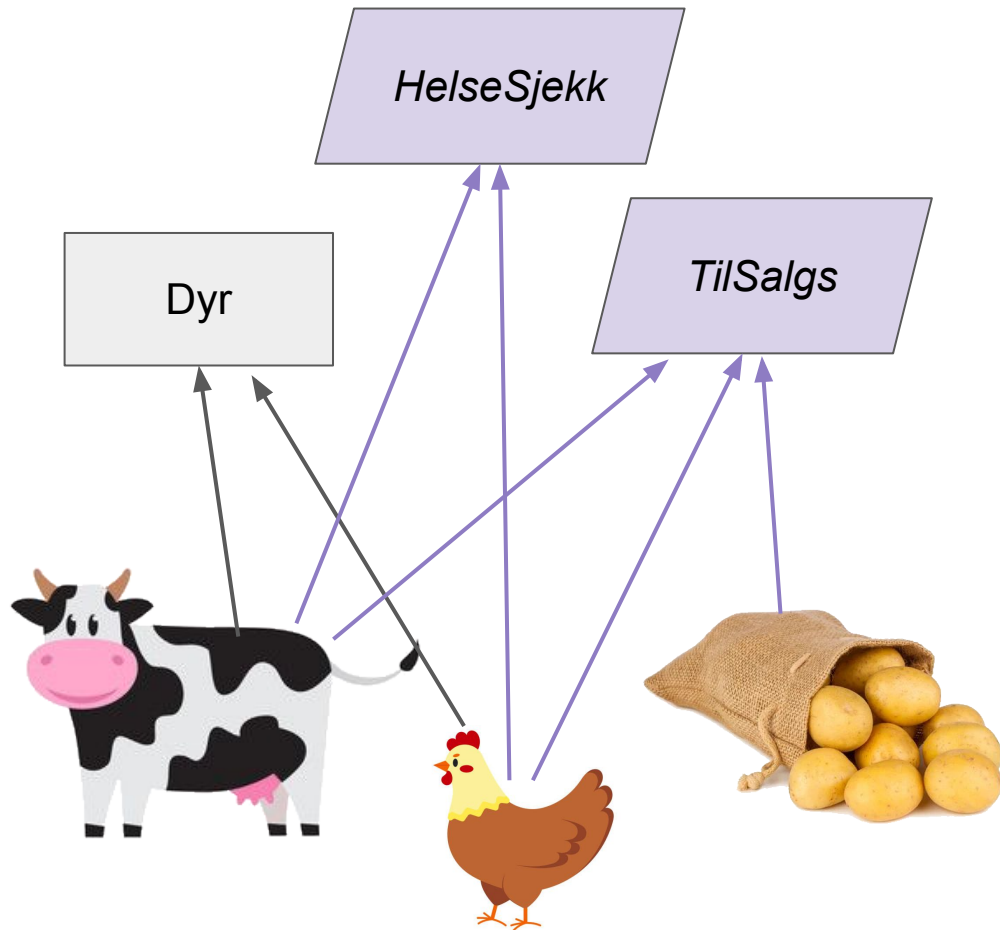


Hvorfor interface?

Eksempel: Bondegårdssystem

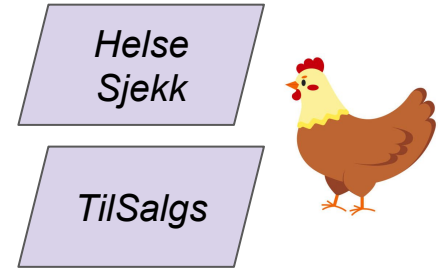
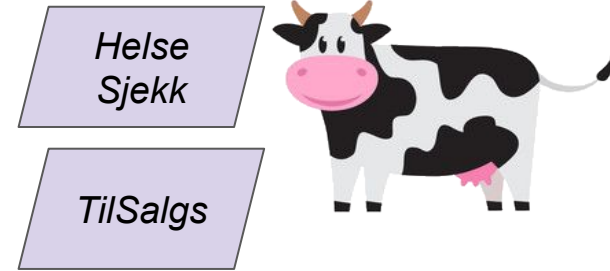
- Både selge dyr og sjekke helsen

→ løsning: interface



Hvorfor interface?

- En klasse kan arve fra 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



```
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;
    }
}
```

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



```
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;  
    }
```

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

- Ulik implementering!



```
@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);
    }
}
```

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



```
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");
    }
  }
}
```

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

```
Poteter 2.5 kg: 20kr
Poteter 2.3 kg: 18kr
Kuen Molly, 3 aar: 1100kr
```


Hva skjer hvis vi kjører hovedprogrammet?

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

```
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;  
    }  
}
```

```
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();  
}
```


Hva skjer hvis vi kjører hovedprogrammet?

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

```
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

```
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();  
}
```

Kort om invarianter



Invariant

- = regel for hvilke verdier en instansvariabel for ha
 - “Pris kan ikke være < 0 ”
- Skal gjelde før og etter du kaller på en metode
- Må tenke på dette når man skriver metodene

```
public void senkPris() {  
    pris -= 100;  
}
```



```
public void senkPris() {  
    if (pris - 100 > 0) {  
        pris -= 100;  
    } else {  
        pris = 0;  
    }  
}
```

Jobb med oppgaver!

IN1010 Emnesiden → Grupper →
Gruppe 7 → Uke5

