IN1010 - Gruppe 11

Dag 6

Agenda

- Mer om lister og lenkelister
  - Doubly linked list (dobbeltlenket liste)
  - Comparable, compareTo
  - Iterable

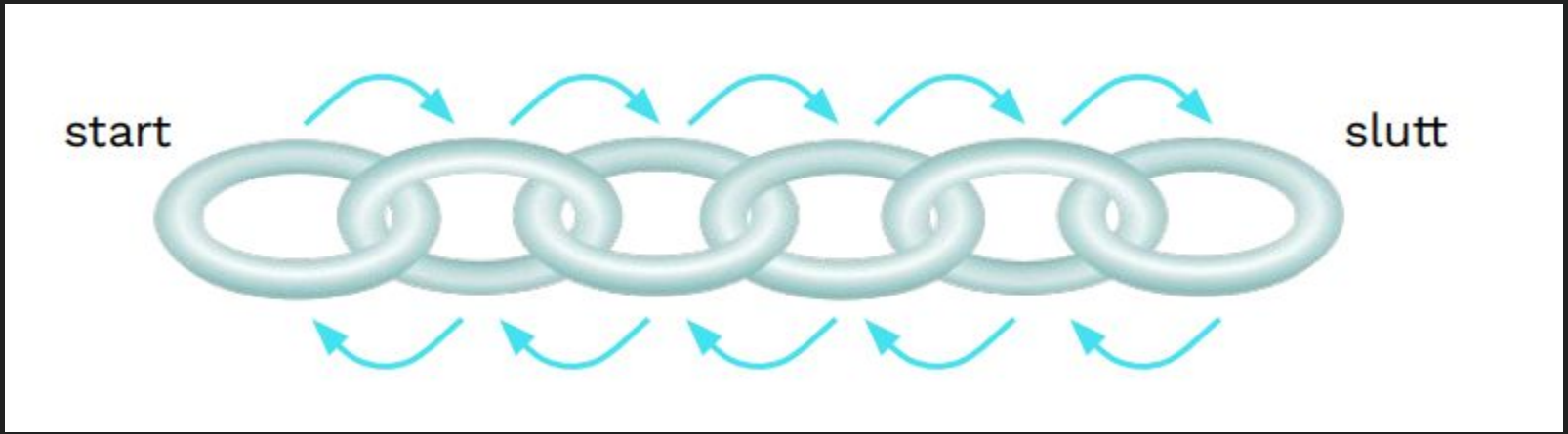
Liten forklaring på en **midtveisoppgave**



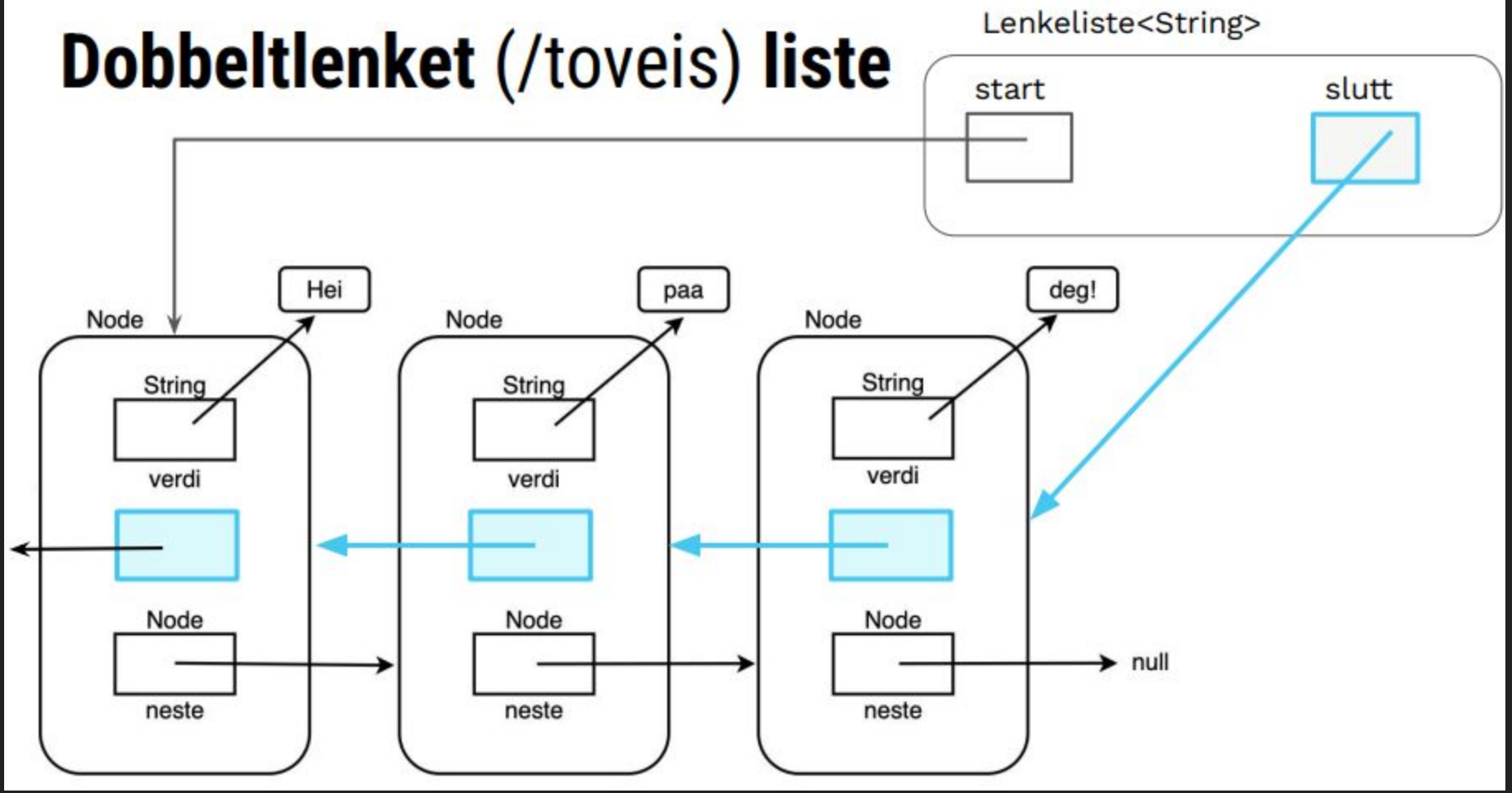
Lenkeliste repetisjon

Dobbeltlenke liste (toveis)

- Kan gå igjennom listen begge retningene
- Lettere sletting av noder som ikke ligger først i listen
- Mer å holde orden på når man skriver metodene



Dobbeltlenket (/toveis) **liste**



Comparable, compareTo

- Comparable: interface med metoden

```
int compareTo(T otherObj);
```

- Implementer hvis vil sammenligne objekter av vår klasse
- **a**.compareTo(**b**) returnerer en int…
  - = 0 hvis lik
  - **positivt** tall hvis **a** er **større** enn **b**
  - **negativt** tall hvis **a** er **mindre** enn **b**

Hva burde skrives ut? (0 / positivt tall /negativt tall)?

- compareTo er implementert i Vare basert på prisen

```
public static void main(String[] args) {
    Vare agurk = new Vare(pris: 20);
    Vare tomat = new Vare(pris: 10);
    Vare melk = new Vare(pris: 20);

    System.out.println(agurk.compareTo(tomat));
    System.out.println(agurk.compareTo(agurk));
    System.out.println(agurk.compareTo(melk));
    System.out.println(tomat.compareTo(melk));
```

Hvordan kan vi implementere compareTo i Vare basert på pris?

```java
class Vare implements Comparable<Vare> {
    int pris;

    public Vare(int pris) {
        this.pris = pris;
    }

    @Override
    public int compareTo(Vare o) {
        return pris - o.pris;
    }
```

```java
class Vare implements Comparable<Vare> {
    int pris;

    public Vare(int pris) {
        this.pris = pris;
    }

    @Override
    public int compareTo(Vare o) {
        if (pris > o.pris) return 1;
        if (pris < o.pris) return -1;
        // pris == o.pris
        return 0;
    }
```

Iterable

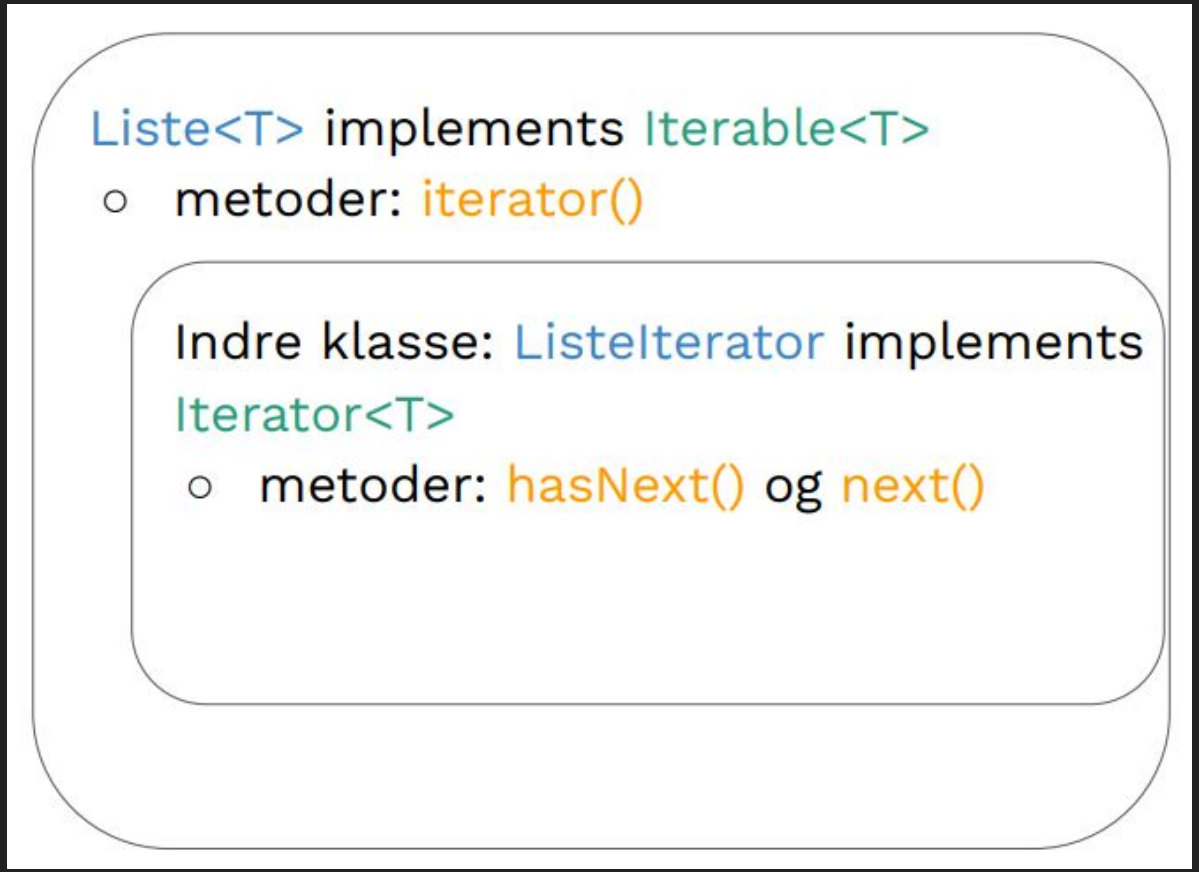
- **Iterable:** interface med metoden

```
public Iterator<T> iterator()
```

- For å kunne gå gjennom listen med en for-each loop

Iterable

- Trenger:

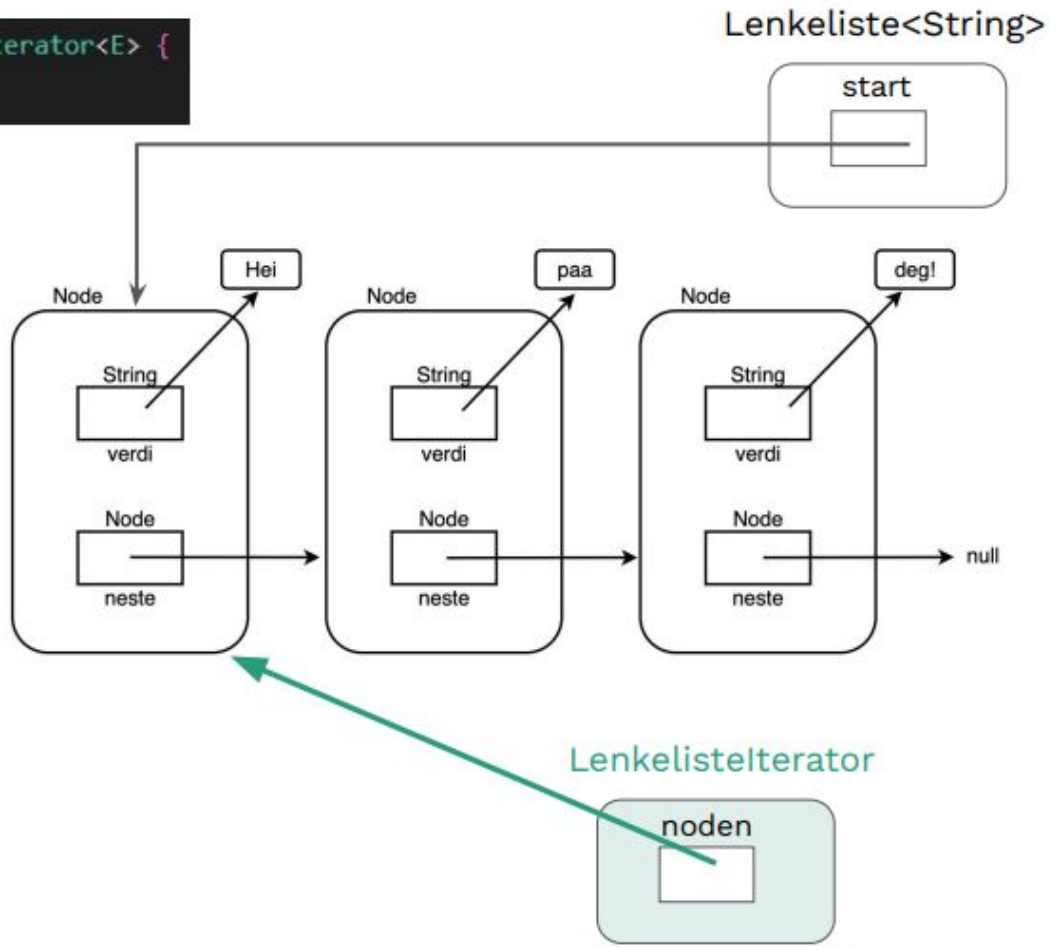


```java
private class LenkelisteIterator implements Iterator<E> {
    Node noden = start;
```

```java
@Override
public boolean hasNext() {
    return noden != null;
}
```

```java
@Override
public E next() {
    if (noden == null) {
        throw new NoSuchElementException(
            s: "next");
    }

    E returVerdi = noden.verdi;
    noden = noden.neste;
    return returVerdi;
}
```

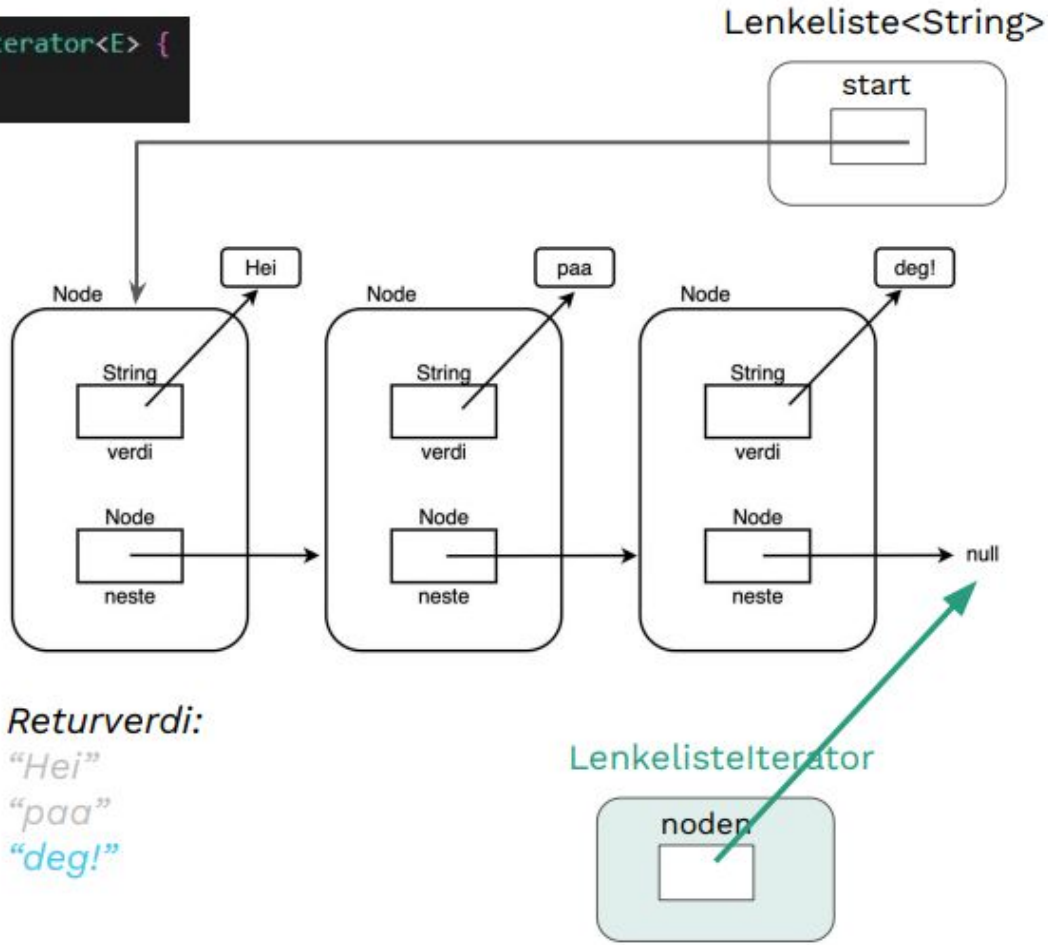


```java
private class LenkelisteIterator implements Iterator<E> {
    Node noden = start;
```

```java
@Override
public boolean hasNext() {
    return noden != null;
}
```

```java
@Override
public E next() {
    if (noden == null) {
        throw new NoSuchElementException(
            s: "next");
    }

    E returVerdi = noden.verdi;
    noden = noden.neste;
    return returVerdi;
}
```



Midtveiseksamen v25 - Kanin

```java
abstract class Kanin implements Comparable<Kanin>{
    final String navn; protected int vekt; protected int alder;
    Kanin(String nv, int v, int a) {navn=nv; vekt=v; alder=a;}
    @Override
    public int compareTo(Kanin med) {return vekt - med.vekt;}
    public static Kanin største(Kanin en, Kanin to) {
        if  (en.compareTo(to) > 0) return en; else return to;
    }
}
class Barnekanin extends Kanin {
    Barnekanin(String nv, int v, int a) {super(nv,v,a);}
    @Override
    public int compareTo(Kanin med){return alder-med.alder;}
}
class Hunnkanin extends Kanin {
    Hunnkanin(String nv, int v, int a) {super(nv,v,a);}
    @Override
    public int compareTo(Kanin med){return navn.compareTo(med.na
}
class Hannkanin extends Kanin {
    Hannkanin(String nv, int v, int a) {super(nv,v,a);}
}
```

```java
class HovedSammenlign {
    public static void main (String[] args) {
        Kanin en = new Barnekanin("Puck", 6, 2);
        Kanin to =  new Barnekanin("Kølle", 5, 3);
        Kanin tre = new Hannkanin("Carsten", 52, 40);
        Kanin fire = new Hannkanin("Jabben", 41, 51);
        Kanin fem = new Hunnkanin("Kine", 36, 32);
        Kanin seks = new Hunnkanin("Anne", 48, 28);
        Kanin syv,åtte,ni,ti,elleve,tolv;
        syv = Kanin.største(en,to);
        åtte = Kanin.største(tre,fire);
        ni = Kanin.største(fem,seks);
        ti = tre.største(en,tre);
        elleve = ti.største(fire,fem);
        tolv = Hunnkanin.største(seks,new Hunnkanin("Eva", 32, 27));

        System.out.println("Kanin 7: " + syv.navn);
        System.out.println("Kanin 8: " + åtte.navn);
        System.out.println("Kanin 9: " + ni.navn);
        System.out.println("Kanin 10: " + ti.navn);
        System.out.println("Kanin 11: " + elleve.navn);
        System.out.println("Kanin 12: " + tolv.navn);
    }
}
```

7 Hva skriver dette programmet ut?

Hint: Husk at f.eks. "Stein".compareTo("Dag") > 0

Hjelp

Kanin 7: [ ]

Kanin 8: [ ]

Kanin 9: [ ]

Kanin 10: [ ]

Kanin 11: [ ]

Kanin 12: [ ]

Carsten | Puck | Anne | Kine | Jabben | Eva | Kølle

Løsning

- **Hovedregel**: **største(en,to)** returnerer objektet der **compareTo(...) > 0.**
- Sammenligningsregler
  - Barnekanin → sammenligner alder
  - Hunnkanin → sammenligner navn (alfabetisk)
  - Hannkanin → sammenligner vekt

```
public static Kanin største(Kanin en, Kanin to) {
    if (en.compareTo(to) > 0) return en;
    else return to;
}
```

Den gjør:

hvis en.compareTo(to) > 0 → returner en ellers → returner to

Altså: den returnerer den som blir "størst" i sammenligningen.

Kanin 7

- Begge er **Barnekanin → alder sammenlignes**

  Puck: alder **2**
  Kølle: alder **3**

```
syv = Kanin.største(en,to);
```

Res: 2 - 3 = -1
→ negativ → Kølle

Kanin 7 er altså **Kølle**

Kanin 8

- Begge er Hannkanin → vekt sammenlignes
  - Carsten: 52
  - Jabben: 41

  Res: 52 - 41 = 11

  → positiv → Carsten
  Kanin 8 er altså **Carsten**

```
åtte = Kanin.største(tre,fire);
```

Kanin 9

- Begge er Hunnkanin → navn sammenlignes
  `"Kine".compareTo("Anne") > 0`

  `K kommer etter A → Kine`
  Kanin 9 er altså **Kine**

```
ni = Kanin.største(fem,seks);
```

Kanin 10

- Her brukes fortsatt `største(en,tre)`
  Sammenligning skjer fra `en.compareTo(tre)`

```
ti = tre.største(en,tre);
```

en er Barnekanin → alder
Res: 2 - 40 = -38
→ negativ → Carsten

Kanin 10 er altså **Carsten**

Kanin 11

- **fire** er Hannkanin → vekt
  41 - 36 = 5
  → positiv → Jabben

  Kanin 11 er altså **Jabben**

```
elleve = ti.største(fire,fem);
```

Kanin 12

- Hunnkanin → navn
- `"Anne".compareTo("Eva") < 0`
  Eva

```
tolv = Hunnkanin.største(seks,new Hunnkanin("Eva", 32, 27));
```

Kanin 12 er altså Eva

Huskeregel

1. Finn hvilken **klasse objektet er**
2. Se hva `compareTo` **sammenligner**
3. Regn ut forskjellen / alfabetisk rekkefølge
4. `> 0` → første objekt
   Ellers → andre objekt

Lab!