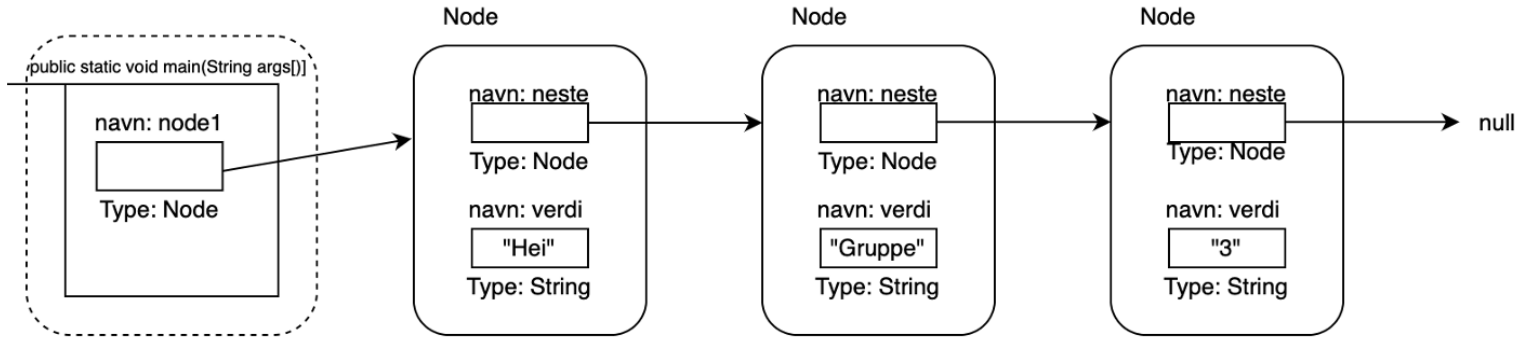
Uke 6

Beholdere 2

Fredagsjava på fredag!

Forrige gang

- Generiske klasser
  - class Klasse<T> {...}
- Lenkelister
- Node som hjelpeklasse/indre klasse

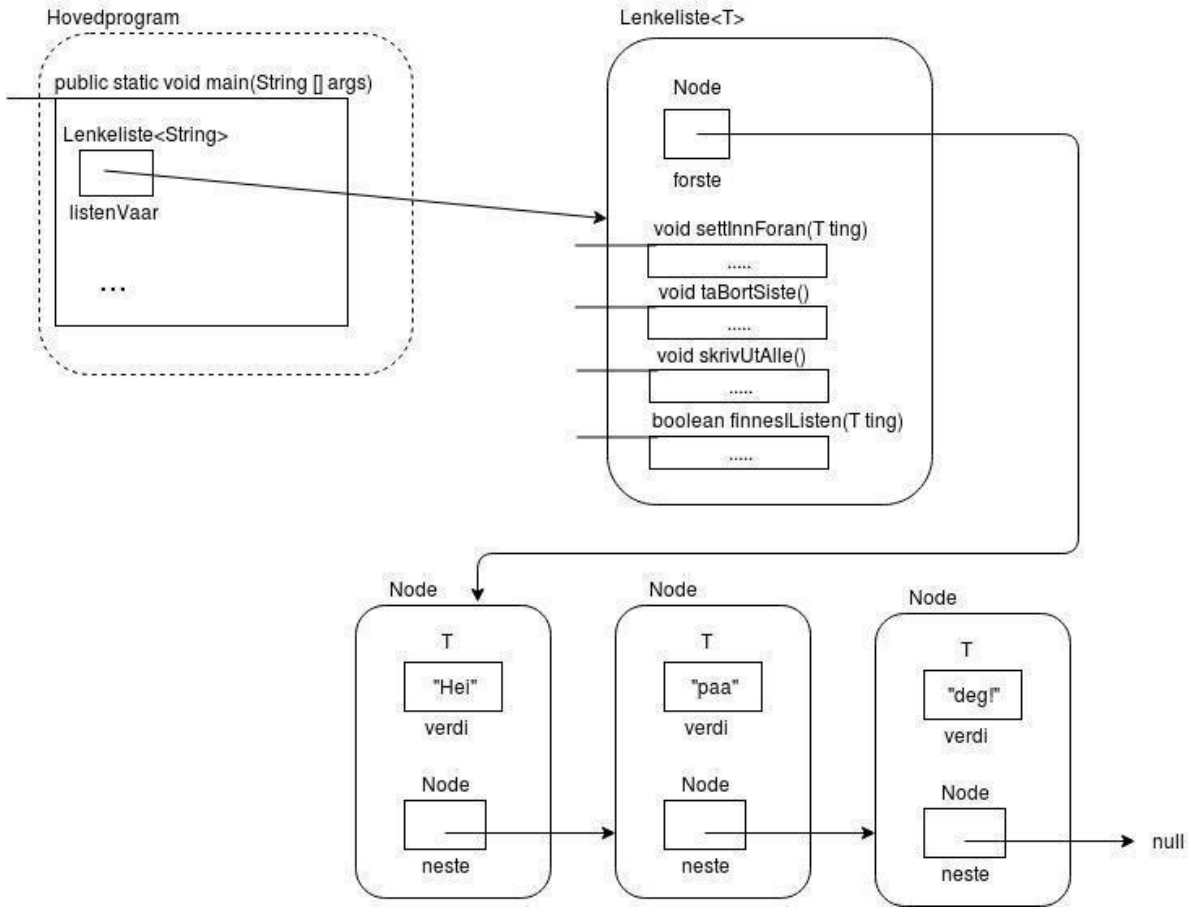


I dag

- Fortsetter med lenkelister
- Enkeltlenket vs dobbeltlenket
- Ulike typer lister
- Interfacet Comparable
- Interfacet Iterable
- 💗 Livekoding 💗

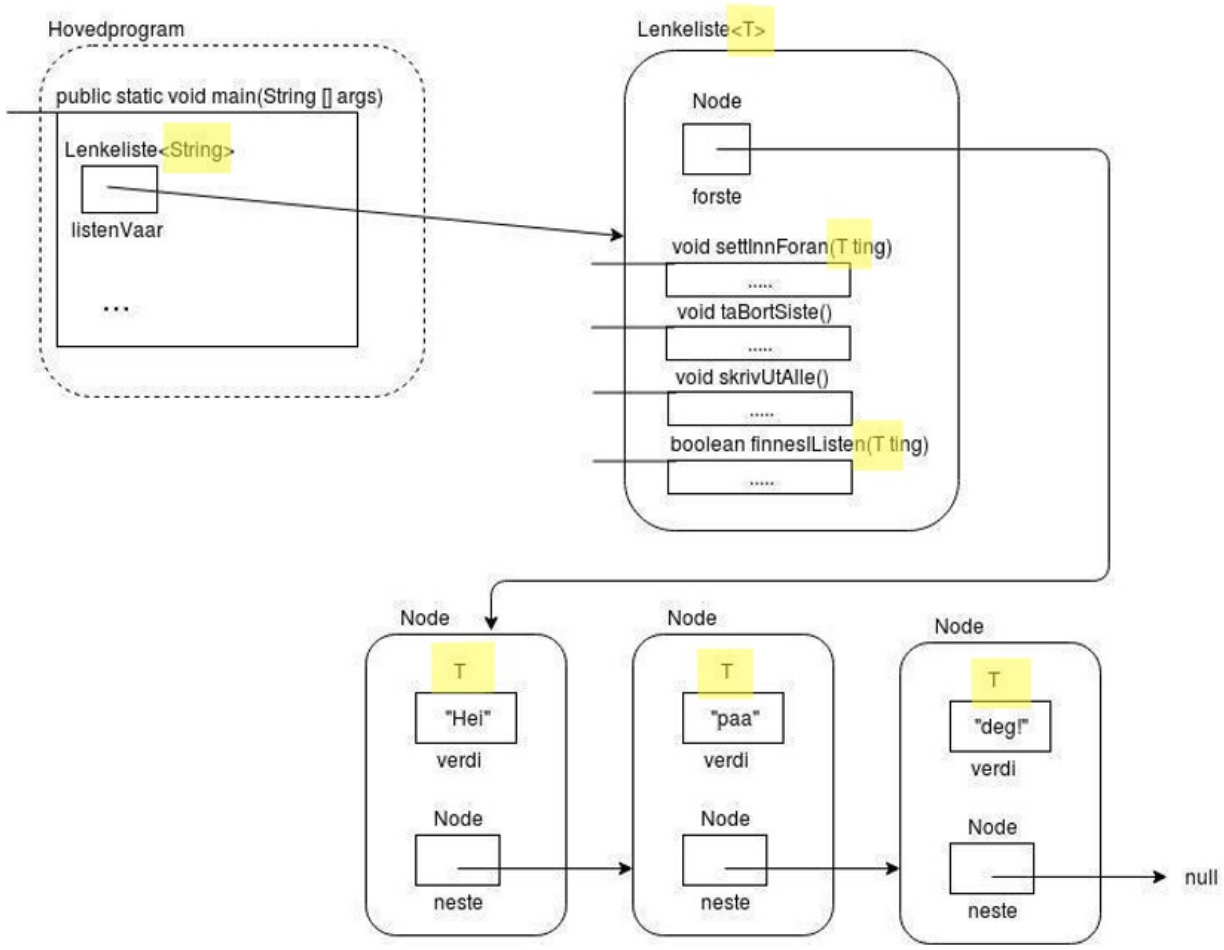
Lenkeliste

- Noder som inneholder data er koblet sammen i en *lenkestruktur* i en lenkeliste.
- Lenkelisten vil håndtere innsetting, sletting og andre operasjoner.
- Vi trenger ikke referanser til nodene i main, men har en referanse til lenkelisten i main.
  - Referanse til første (evt. siste) node ligger inni lenkelisten.
- Gjennom referansen får vi tilgang til listemetoder og dermed data lagret i nodene våre.



Lenkeliste

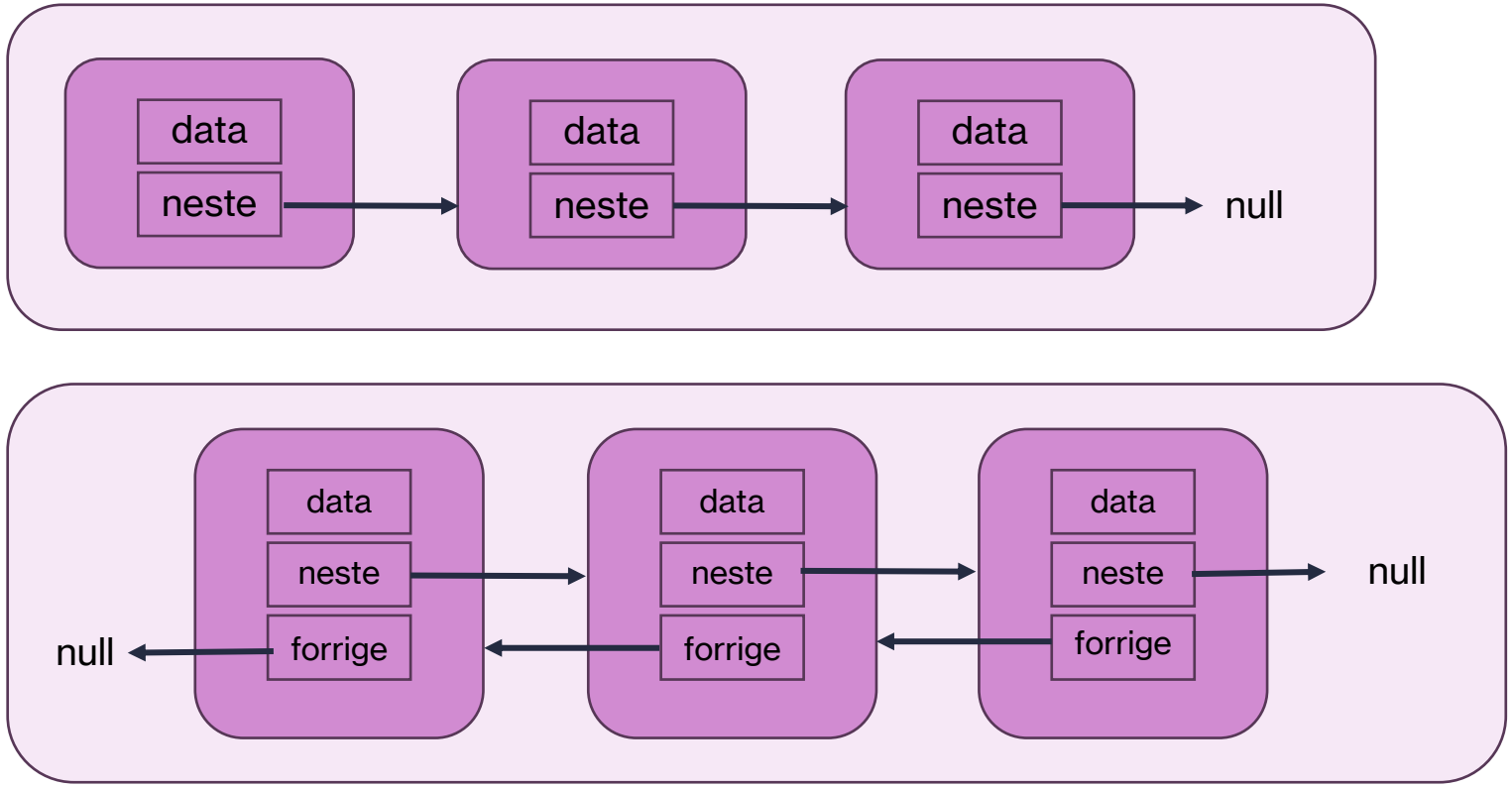
- Lenkelister kan være generiske
  - Klassen tar inn et typeparameter
- Lenkelisten tar inn en generisk type, mens den indre Node-klassen bruker typen.
- *wrapper-klasser (boxing)*
  - Integer, Double, Boolean



Enkeltlenkel liste vs. dobbeltlenket liste

- **Enkeltlenket**
  - Kun neste-peker
  - Siste node: neste peker til null
- **Dobbellenket**
  - Neste-peker og forrige-peker
  - Første node: forrige peker til null
  - Siste node: neste peker til null

- Ofte er enkeltlenket nok, men noen ganger har man behov for å traversere bakover eller ha oversikt over både elementet før og etter.



ENKELTLENKET

```
public class Enkeltlenket<T> {
    Node forste, siste;

    class Node {
        T data;
        Node neste;
        Node(T data) {
            this.data = data;
        }
    }
}
```

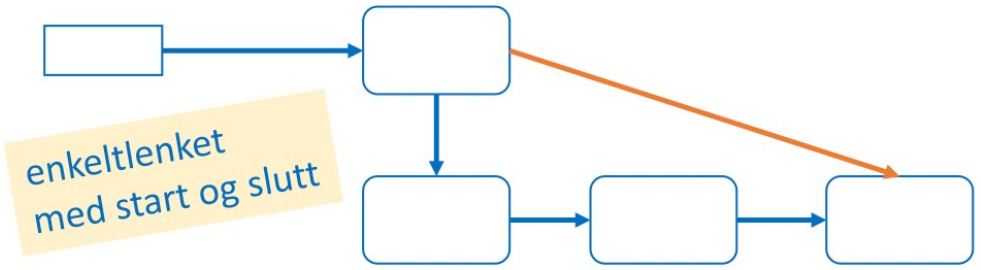
DOBBELTLENKET

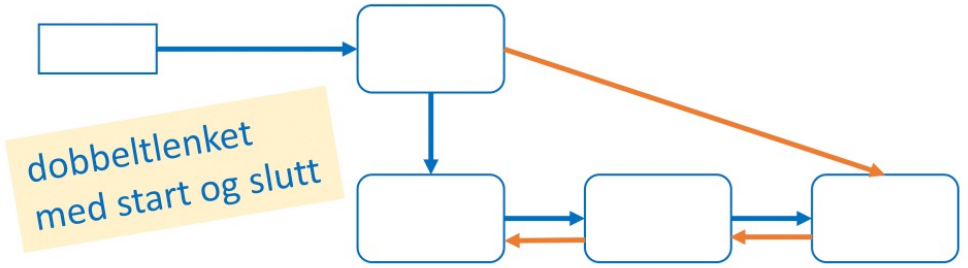
```
public class Dobbeltlenket<T> {
    Node forste, siste;

    class Node {
        T data;
        Node forrige, neste;
        Node(T data) {
            this.data = data;
        }
    }
}
```

Lenkeliste

- Ofte lurt å ha en peker til **første** node, og **siste** node.
  - Hvis man skal ha tak i siste element/legge til noe bakerst, men kun har peker til første node må man traversere hele lenkelisten.





Ulike typer lister

- ***Stabel/stack*** (LIFO) - elementer settes inn og tas ut foran
  - (f.eks. tallerkenstabel)
- ***Kø*** (FIFO) - elementer settes inn bakerst og tas ut foran
  - (f.eks. kassekø i butikken)
- ***Prioritetskø*** - elementer settes inn og tas ut etter ønsket prioritet
  - (f.eks. eldre først)
  - Elementene er sortert etter det de prioriteres på, og man henter ut det første elementet.

- Metode for å legge til og fjerne elementer fra listen er implementert annerledes ut i fra hvilken type

Interfacet Comparable

- Brukes for å sammenligne objekter på en ønsket måte.
- Praktisk i prioritetskøer
  - Klassen Pasient med en variabel for hvor syk den er, og dermed prioritet
  - Klassen Person som prioriteres ut i fra alderen.
- Comparable er generisk: har klasseparameter der vi sender med typen som skal sammenlignes

Comparable sin compareTo

compareTo-metoden må implementeres i klasser som implementerer Comparable

- Leksikografisk (alfabetisk) bokstavsammenligning
- Returnerer et positivt heltall når objektet vi sammenligner med er “mindre”
- Returnerer et negativt heltall når objektet vi sammenligner med er “større”
- Returnerer 0 når objektene vi sammenligner er like

```
public interface Comparable<T> {

    public int compareTo(T o);

}
```

Objektet vi sammenligner med er det som sendes inn i compareTo()

Interfacet Iterable

- når vi vil iterere over elementer i lister og andre beholdere med en for each-løkke
- hashNext() – sjekker om det er et til element å iterere over.
- next() – går til neste element.
- hasNext-metoden og next-metoden kan tilpasses nøyaktig slik vi ønsker å ha dem

Livekoding

- Enkeltlenket lenkeliste av typen Kø (FIFO)
- Implementere Iterable i en lenkeliste
- Comparable