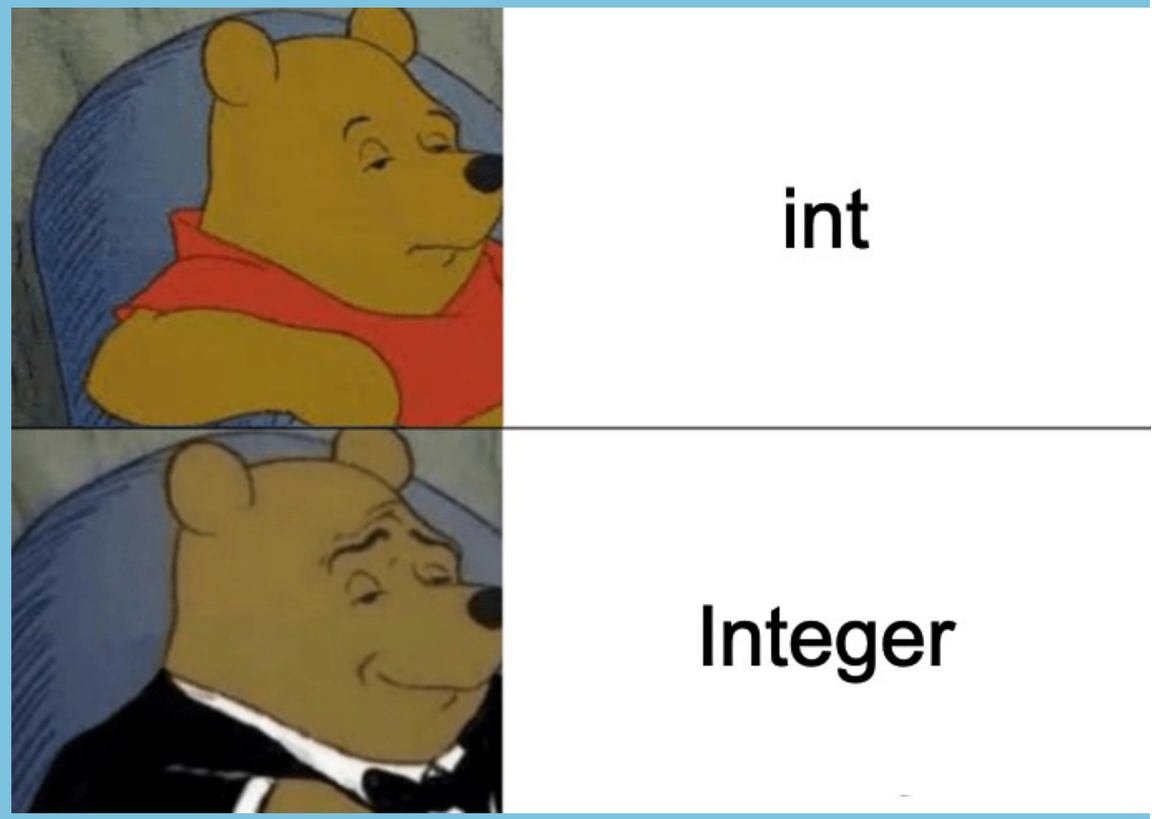
Uke 6 - Beholdere

Gruppe 11 - jonasbny@ifi.uio.no

Agenda



Noder

lenkelister

enkeltlenket vs dobbeltlenket

comparable interfacet

iterable og iterator interfacet

FredagsJava

Noder

Hva er en node og hvorfor bruker vi de

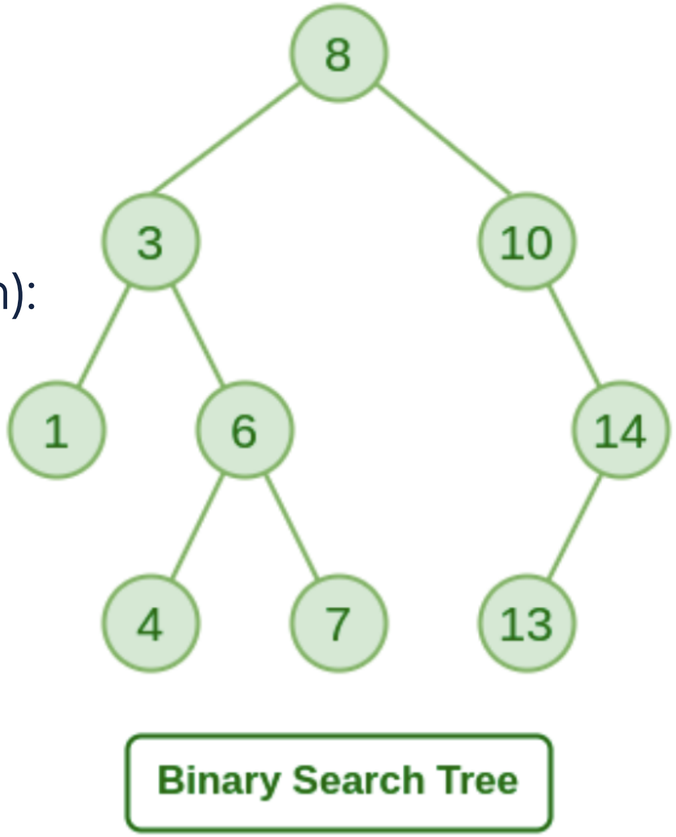
I informatikk refererer en “node” ofte til en grunnleggende enhet i datastrukturer

Inneholder vanligvis data og en eller flere pekere som peker på andre noder

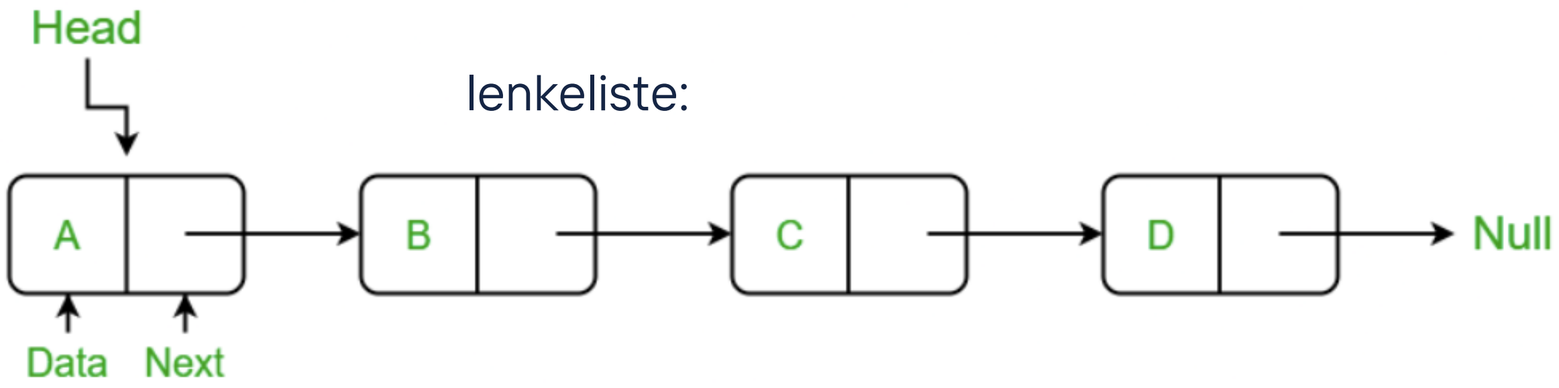
En byggestein som brukes i ulike sammenhenger for å organisere og behandle data.

eksempler på datastrukturer:

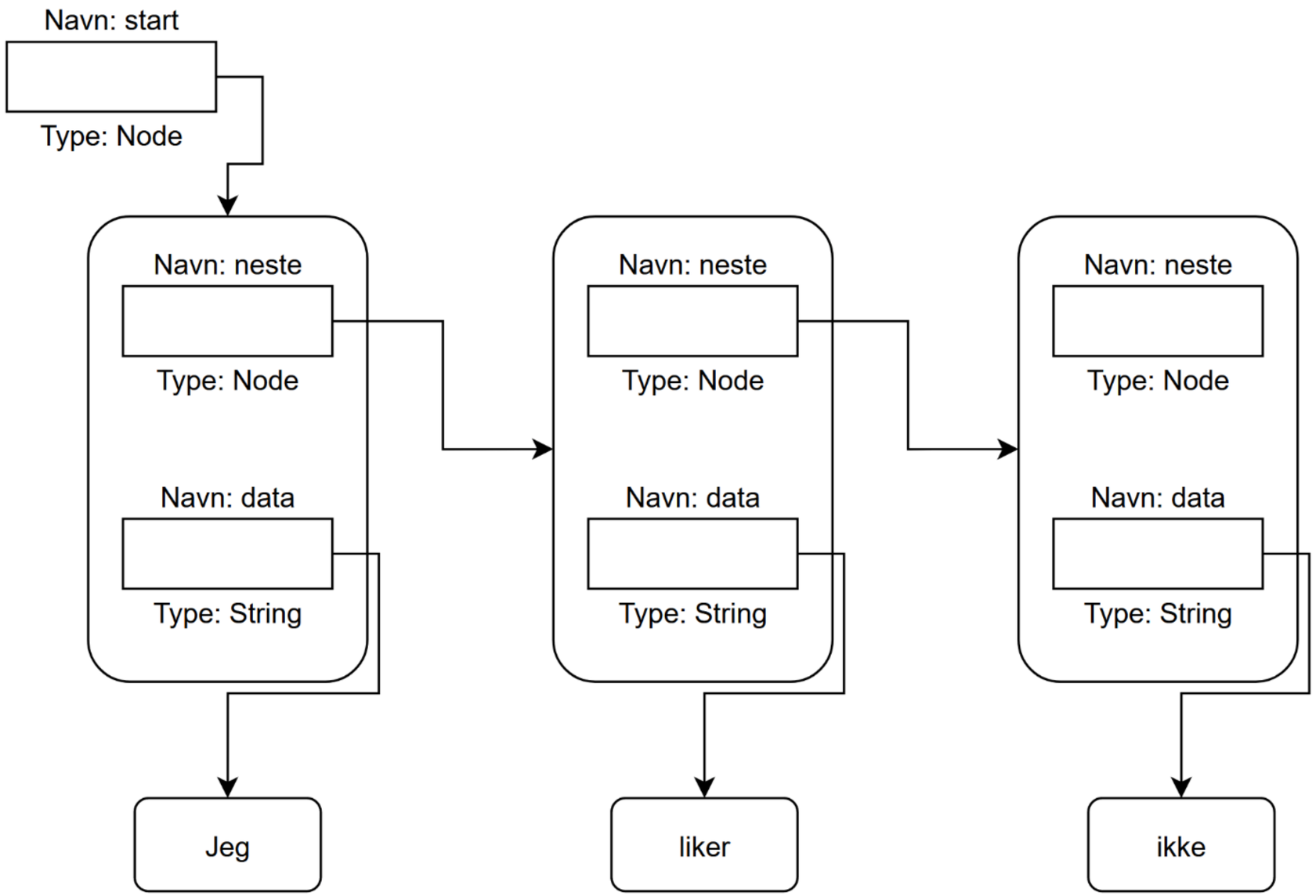
binært søketre(ikke pensum):



lenkeliste:



Lenkeliste


Navn: start
Type: Node
Navn: neste
Type: Node
Navn: data
Type: String
Navn: neste
Type: Node
Navn: data
Type: String
Navn: neste
Type: Node
Navn: data
Type: String
Jeg
liker
ikke

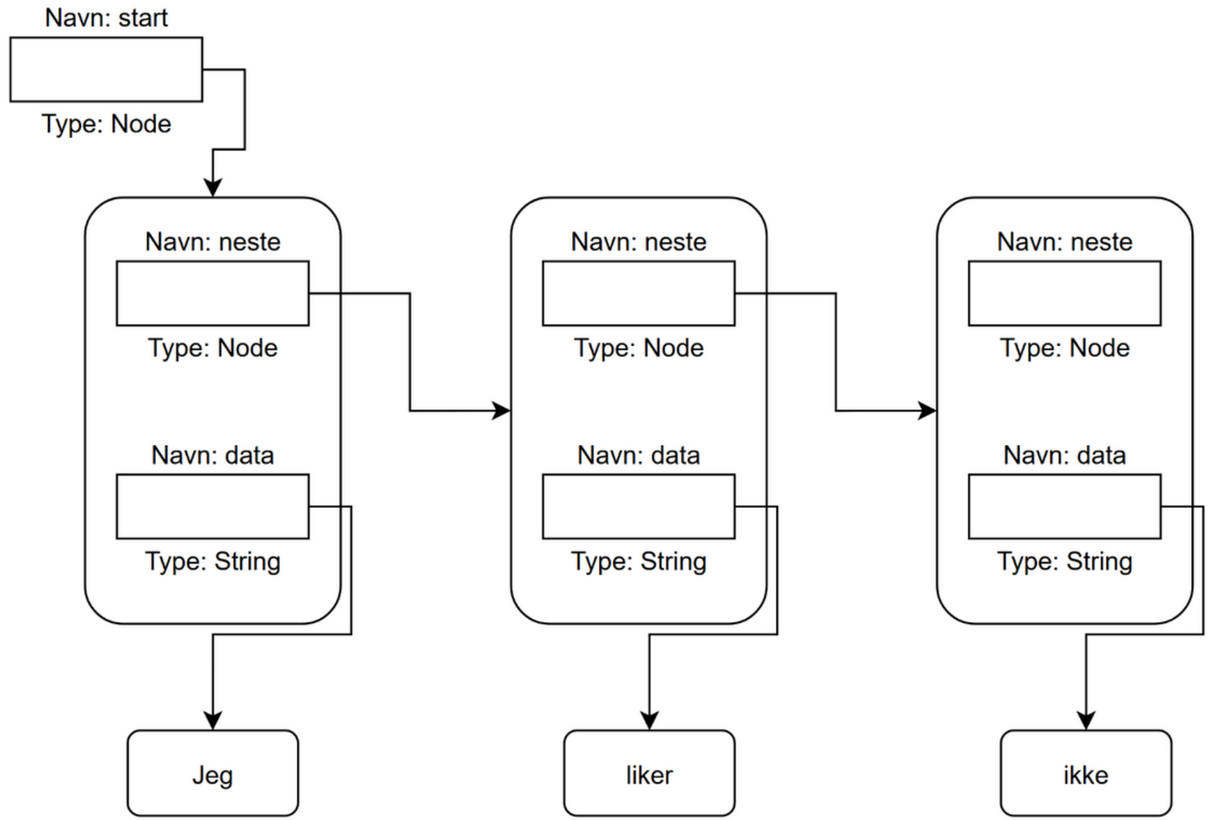
Lenkeliste

Enkelt sagt er en lenkeliste en samling av noder der hver node peker til neste node i listen.

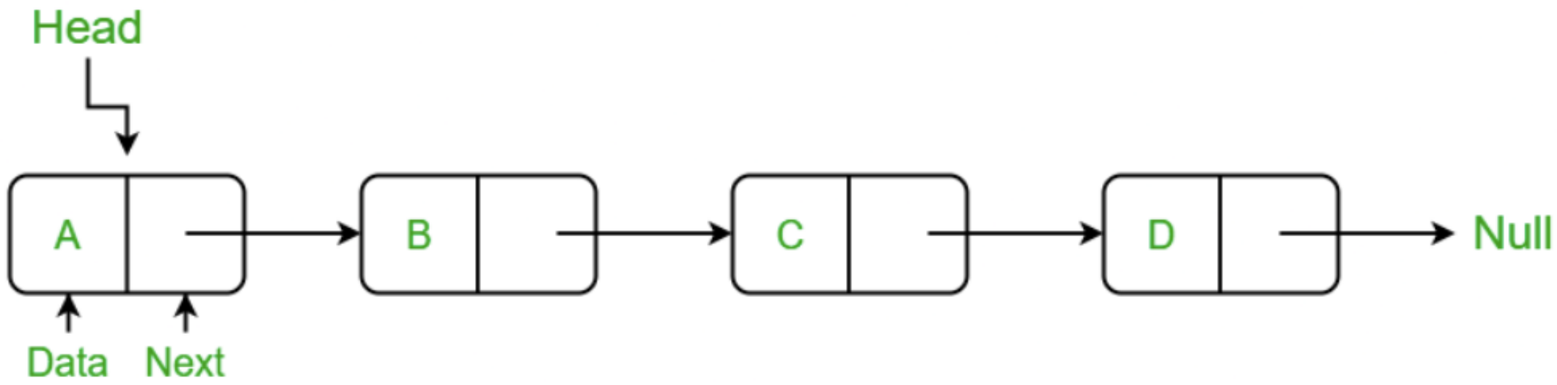
Lenkelister håndterer nodene for oss

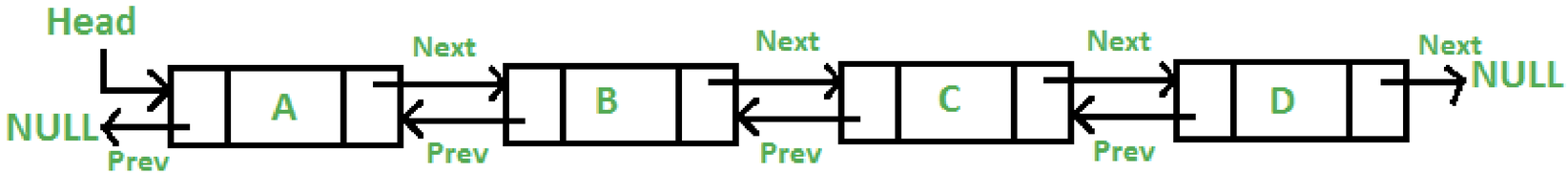
Vi implementerer selv hvordan vi henter ut og setter inn elementer
(og eventuelt andre metoder)

Hvorfor lærer vi oss lenkelister?



Enkeltlenket og Dobbeltlenket





Oppgave

Fordeler og ulemper med dobbeltlenket over enkeltlenket?

Listevarianter

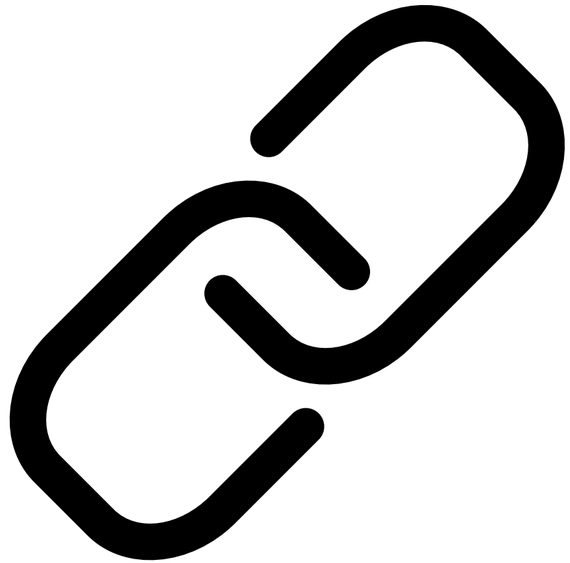
Nå har vi muligheten å lage forskjellige lister

Vanlige varianter:

FIFO - first in, first out
  -en typisk kø

LIFO / Stack - last in, first out
  -en stabel

Prioritetskø
  -elementene blir sortert basert på noe, og det med høyest prioritert hentes ut først



Oppgave

Hva må man kunne gjøre for å lage en prioritetskø?

Hint: elementer i en prioritetskø sorteres

Sammenligne objekter

For å kunne implementere en prioritetskø er vi nødt til å kunne sammenligne objekter.

Lett å sammenligne ting Java vet om, for eksempel int og String.

Når vi bruker == sammenligner vi referansene til objekter, ikke objektenes faktiske innhold.
-derfor har String equals() metoden



Comparable

Comparable er et interface

Inneholder en metode: compareTo()

```
public int compareTo(Object obj);
```

forste*.compareTo(*andre) skal returnere:
0 dersom *forste* og *andre* er **like**
positivt tall dersom *forste* er **større** enn *andre*
negativt tall dersom *forste* er **mindre** enn *andre*

*Du bestemmer selv hva det vil si om to objekter er like, mindre enn, eller større enn hverandre

Comparable

Når vi lager egne beholdere kan vi “tvinge” elementene i beholderen til å *måtte* implementere interfacet Comparable

```
public class prioritetsKø<T extends Comparable<T>>{
    T first;

    public void settInn(T newElement){

        // ...

        if(newElement.compareTo(first) > 0){
            // ...
        }

        // ...
    }
}
```

Comparable

men??
class prioritetsKoe<T extends Comparable<T>>?
Er ikke Comparable<T> et interface?
Betyr det da at T også er nødt til å være et interface?

Nei!

“extends” her er en placeholder for *enten* extends *eller* implements

Java gidder ikke å lage et eget kodeord som kommer i tillegg til disse to, og bruker bare “extends”

```
public class prioritetsKoe<T extends Comparable<T>>{
    T first;

    public void settInn(T newElement){

        // ...

        if(newElement.compareTo(first) > 0){
            // ...
        }

        // ...
    }
}
```

Oppgave

Hvis vi har en Person klasse, hva ville compareTo sammenlignet?

Iterable og Iterator

Vi kan også designe beholderen våres slik at det går an å iterere gjennom den med en **for-each-**løkke

for(E element : beholder) {...}

Iterable: et interface som representerer en samling av elementer som kan bli iterert over

Iterator: et interface som gir metoder for å traversere gjennom elementer av en **iterable**



Oppgaver