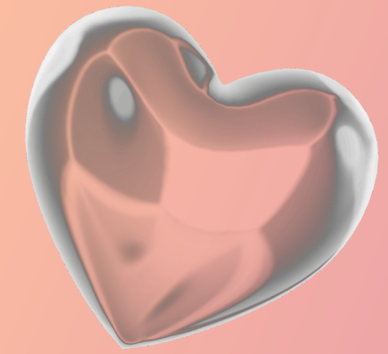
Beholdere

Repetisjon av uke 6 og 7

Sander Segbø Løvaas
sandersl@ifi.uio.no

Planen for i dag

Beholdere
Generiske klasser
Lenkeliste
Noder
Comparable
Iterable og iterator




Beholdere

Når vi koder bruker vi beholdere hele tiden.

Java har mange innbygde beholdere.

- Arrays, ArrayList, HashMap, PriorityQueue, HashSet, osv.

Det kan hende vi ønsker å designe beholdere selv

Relevant for algoritmer og datastrukturer (IN2010)

Array og ArrayList

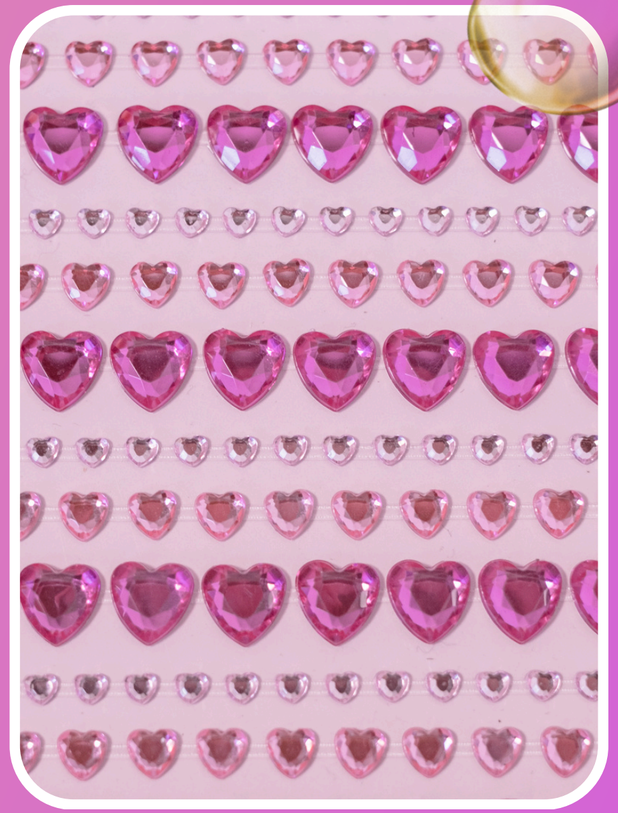
Array kan inneholde både primitive datayper og objekter, mens ArrayList kan bare inneholdeobjekter.

Array har begrenset funksjonalitet, ArrayList inneholder innebygde funksjoner.

ArrayList aksesserer elementer litt annerledes enn Arrays

- Array bruker man liste[0]
- ArrayList bruker man liste.get(0)

Alt info om ArrayList står i Java dokumentasjonen



Array og ArrayList

Actions on Array and ArrayList	Array	ArrayList
Declaration	String myList = new int[5];	ArrayList<String> myList = new ArrayList<String>();
Create new element	String a = new String("hello");	String a = new String("hello");
Add to list	myList[0] = a ;	myList.add(a);
Get size	myList.length ;	myList.size();
Get element from list	String s = myList[0];	Object o = myList.get(0);
Delete element	myList[0] = null;	myList.remove(0);

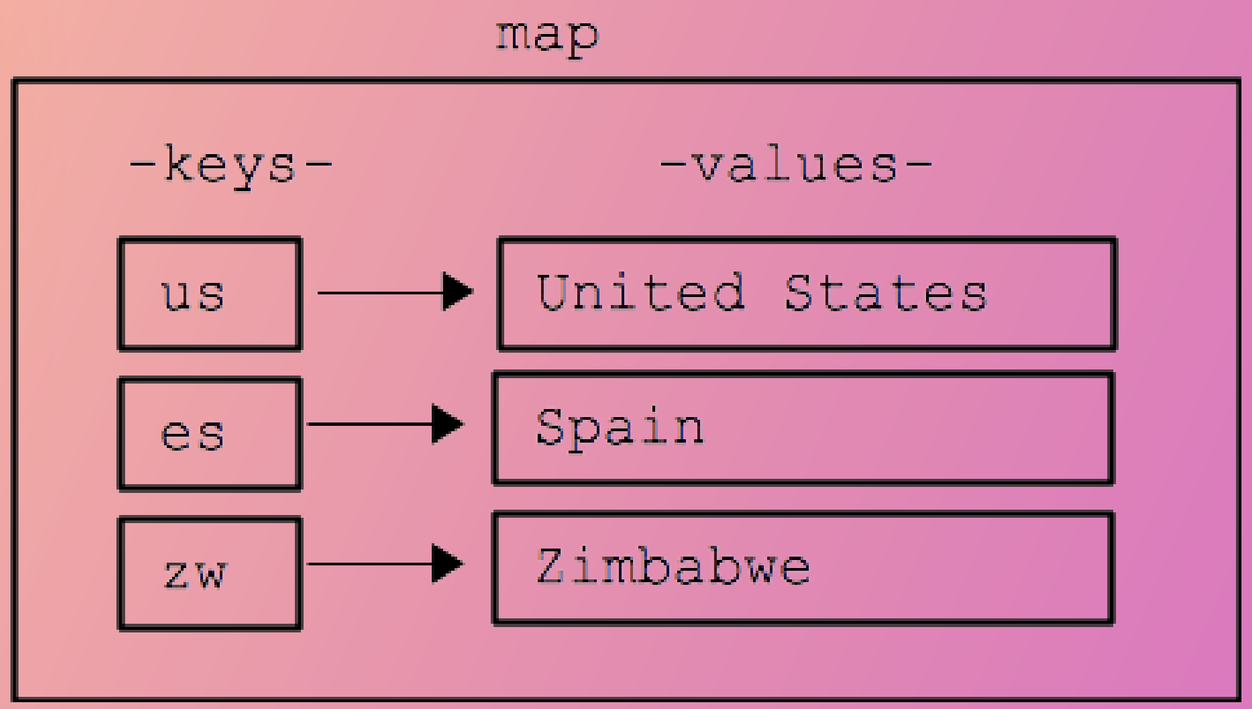
Check the given element is there or not	boolean isIn = false;
	
	for(String item : myList){
	If(a.equals(item){
	isIn = true;
	break;
	}
	}
		boolean isIn = myList.contains(a);

Array og ArrayList

```java
// Initiering
int[] intArray = new int[10];
String[] stringArray = {"Hello", "World"};

// Hente ut elementer
int firstElement = intArray[0]; // Hente første element
intArray[1] = 20; // Tilordne verdi til andre element

for (int i = 0; i < intArray.length; i++) {
    System.out.println(intArray[i]);
}

// Iterering over arrays
for (String str : stringArray) {
    System.out.println(str);
}
```

```java
// ArrayList
// Initiering
ArrayList<String> list = new ArrayList<>();

// Legge tilelementer
list.add("Hello");
list.add("World");

// henting og setting av elementer
String firstElement = list.get(index:0);
list.set(1, "Java");

// Fjering av elementer
list.remove(i);

//Iterering over ArrayLists
for (int i = 0; i < list.size(); i++) {
System.out.println(list.get(i));
}

for (String str : list) {
    System.out.println(str);
}
```

HashMaps


map
-keys-
-values-
us
United States
es
Spain
zw
Zimbabwe


us
es
zw

HashMaps

```java
// HashMaps
//Initiering
HashMap<String, Integer> map = new HashMap<>();

// Legge tilog oppdatere
map.put("One", 1);
map.put("Two", 2);

// Hente ut elementer
int value = map.get(key:"One");

// Fjerne elementer
map.remove("Two");

// Iterering over HashMap
for (String key : map.keySet()) {
System.out.println("Key: " + key + " Value: " + map.get(key));
}

for (Map.Entry<String, Integer> entry : map.entrySet()) {
    System.out.println("Key: " + entry.getKey() + " Value: " + entry.getValue());
}
```

Generiske klasser

Når vi lager beholdere i Java ønsker vi at den skal kunne holde på objekter av hvilken som helst klasse

Upraktisk hvis en beholder bare kan holde på en type

Vi må da bruke det som kalles en generisk klasse

Altså en klasse som tar inn en annen klasse som “parameter” til klassen.

Klasseparameter /typeparameter

Oppgis i signaturen til klassen
class Beholder<T> {...}

En plassholder for hva beholderen skal ta inn
T står for type, men vi kan bruke andre som:

E Element

K Key

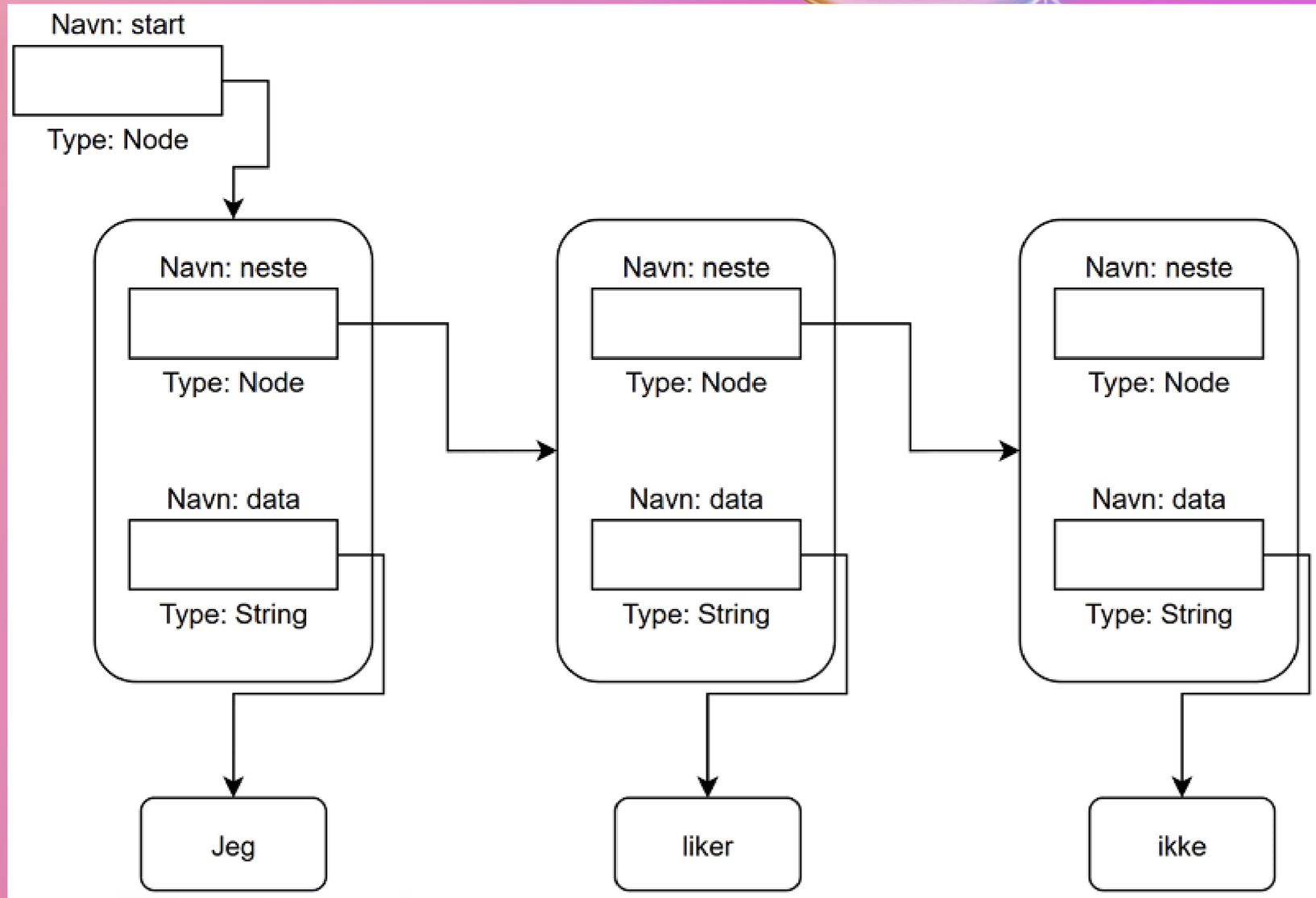
V Value

N Number

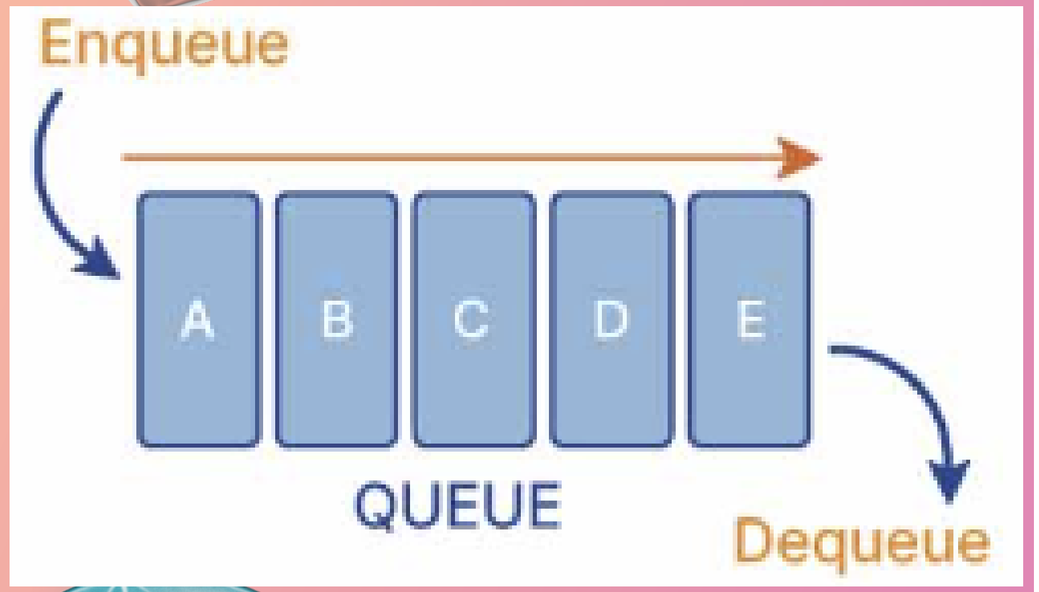
Lenkeliste

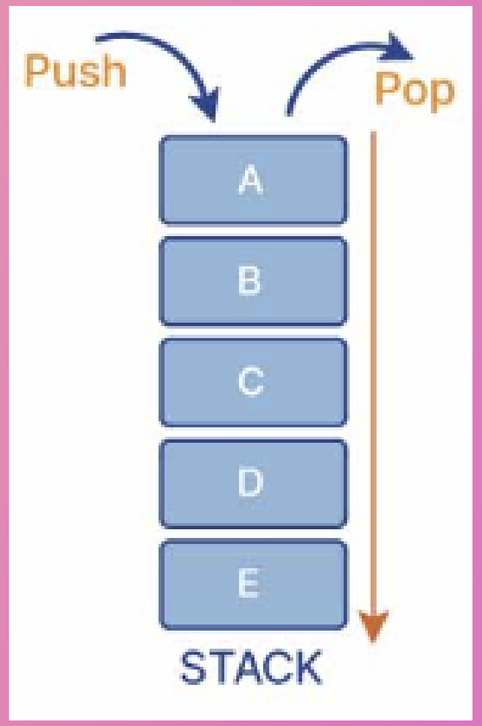
For å slippe å hekte masse noder i main, lager vi et eget objekt som håndterer nodene

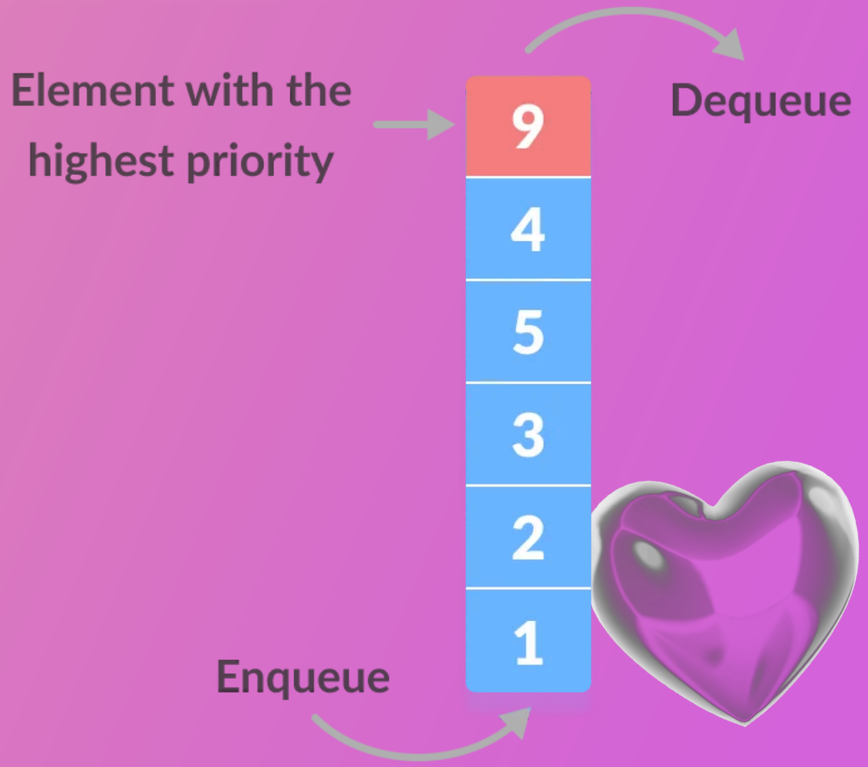
Tilbyr de metodene vi behøver for å hente ut og sette inn
- andre metoder som size()



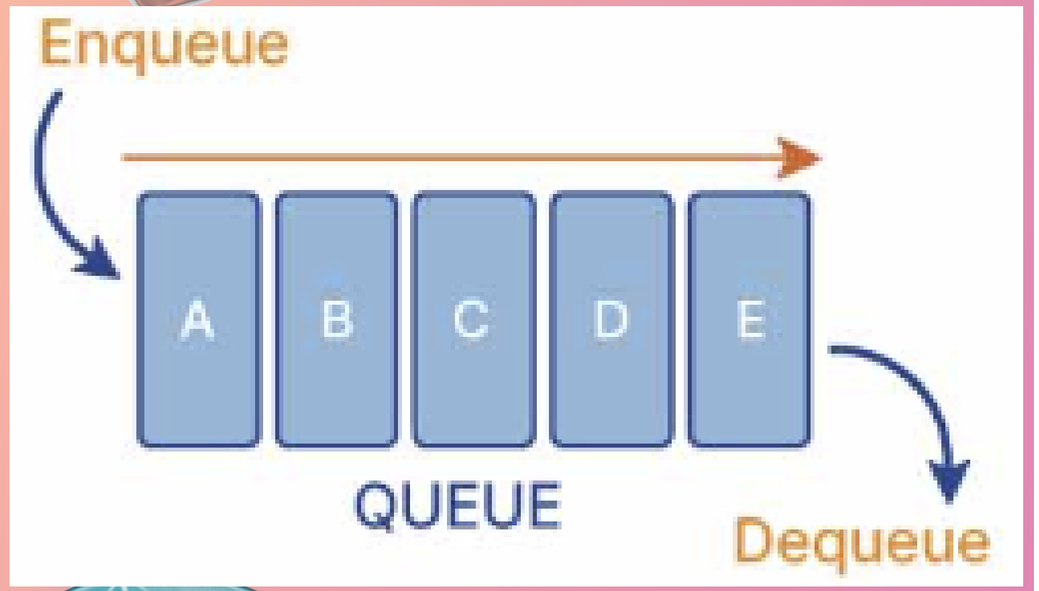
Listevarianter



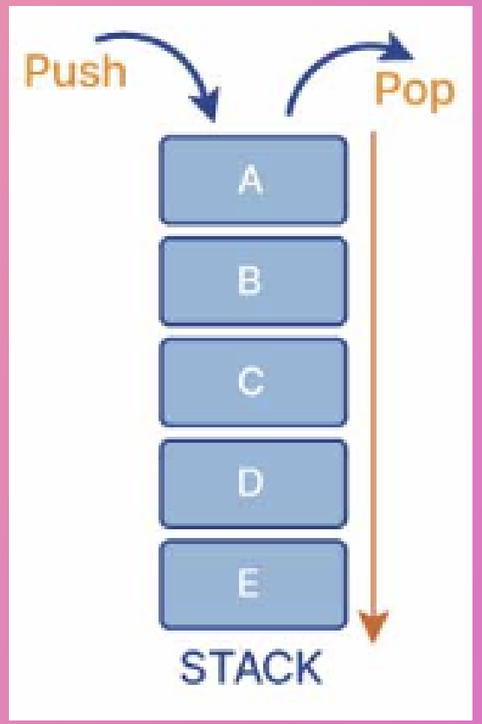




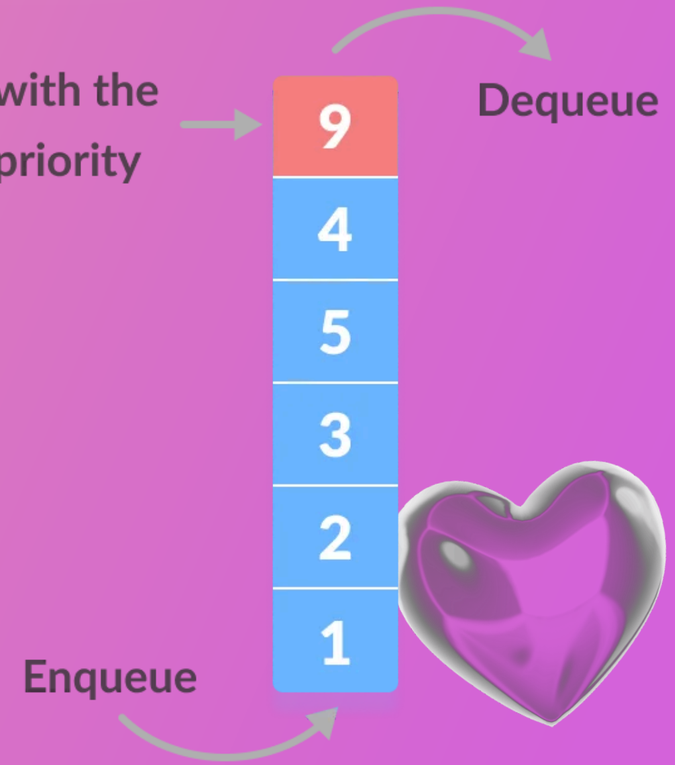
Listevarianter



FIFO - First in, first out



LIFO/Stack/ stabel - last in, first out



Prioritetskø - Elementene blir sortert basert på noe,og det med høyest prioritet hentes ut først

Noder

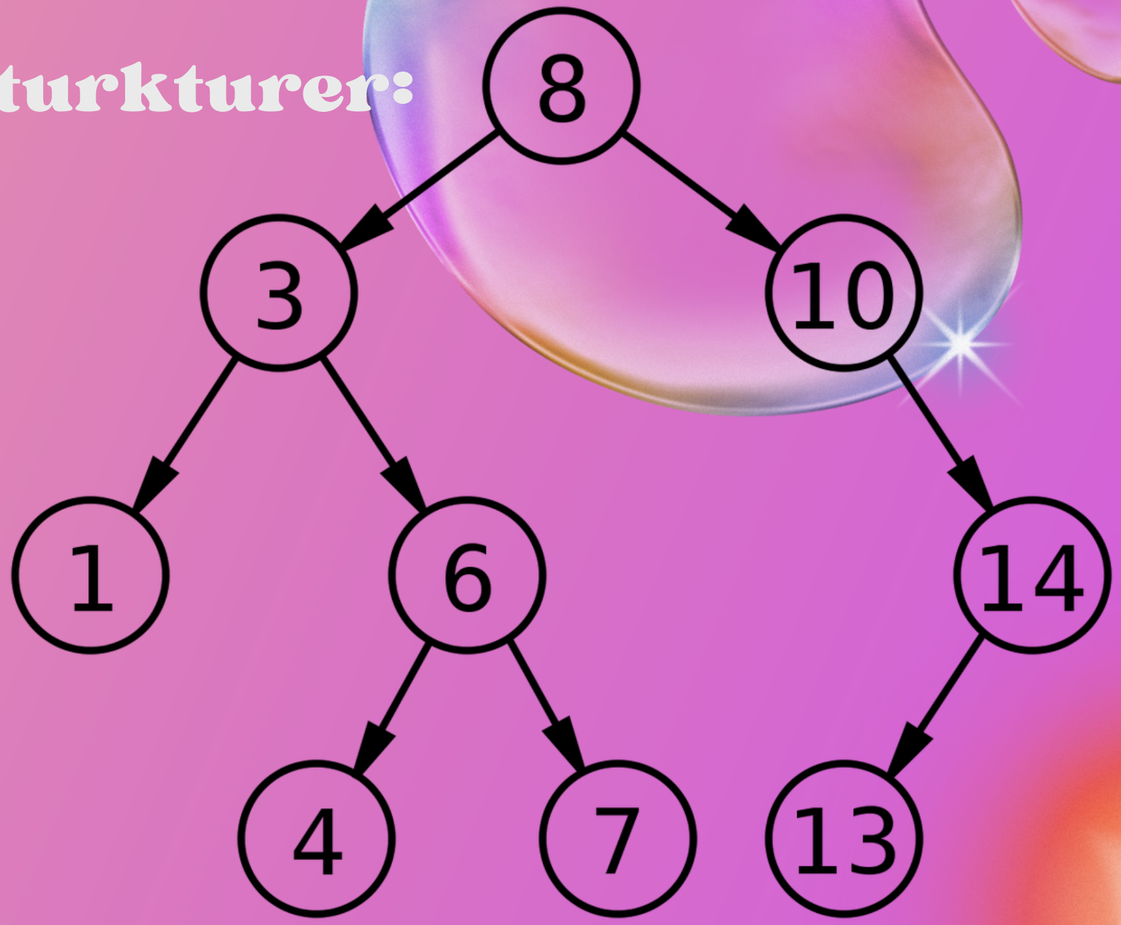
Hva er en node og hvorfor bruker vi de?

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

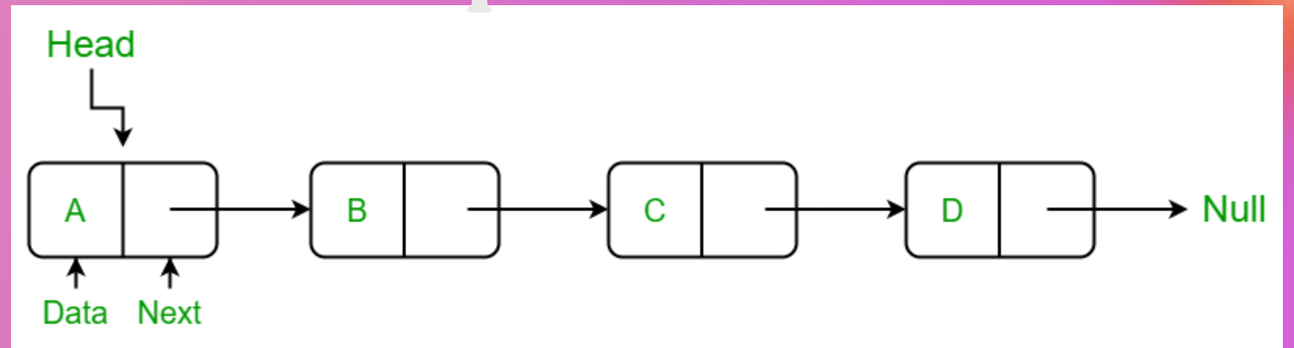
Inneholder vanligvis data og en eller flere pekere som peker på andre noder.

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

Datasturkturer:



Binært søketre (ikke pensum i IN1010)



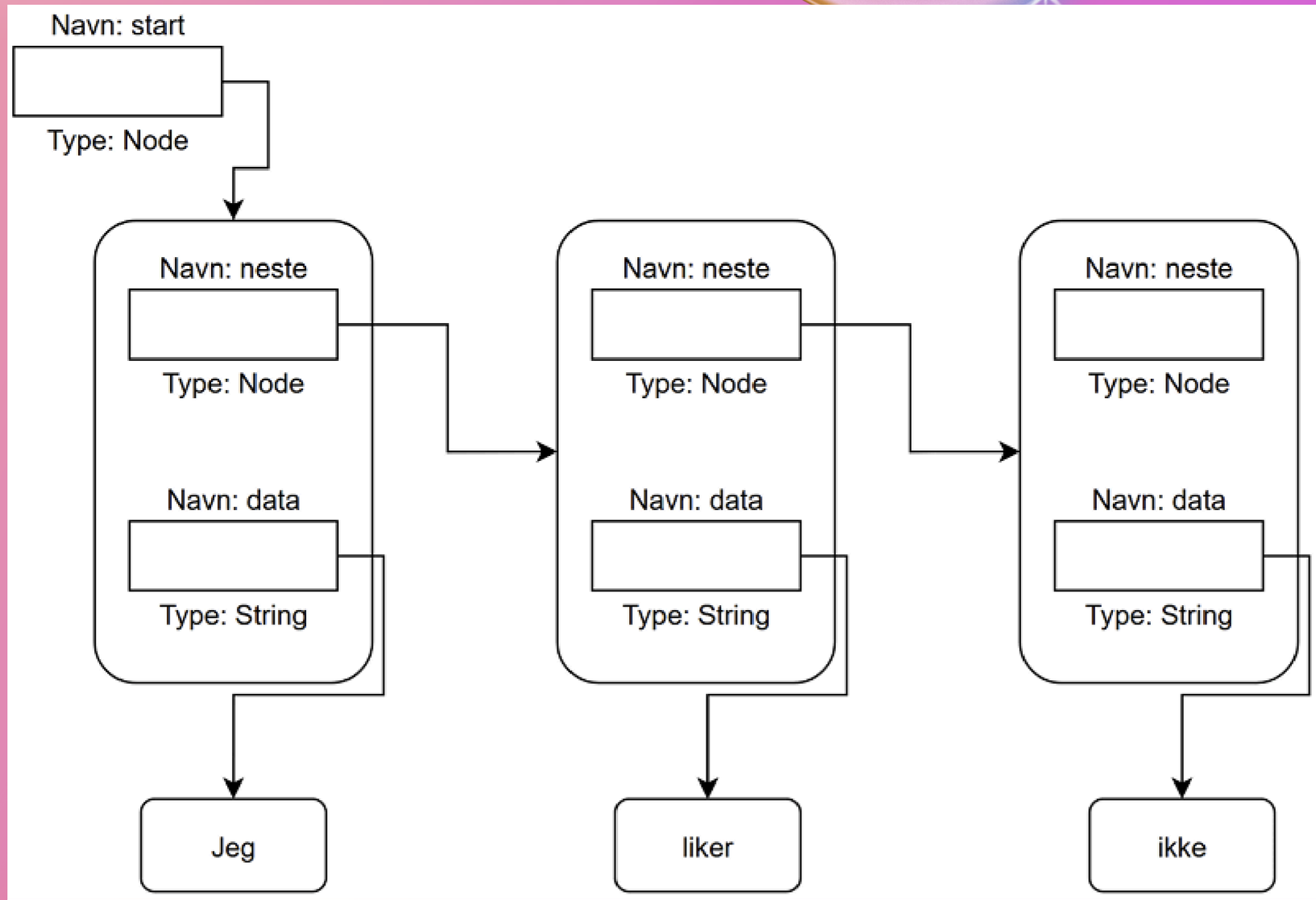
Lenkeliste

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

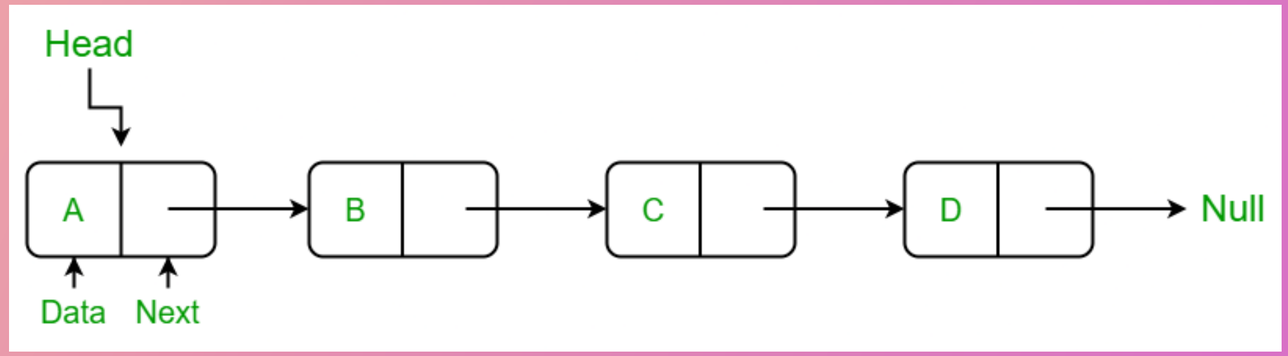
Lenkelister håndterer nodene for oss.

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

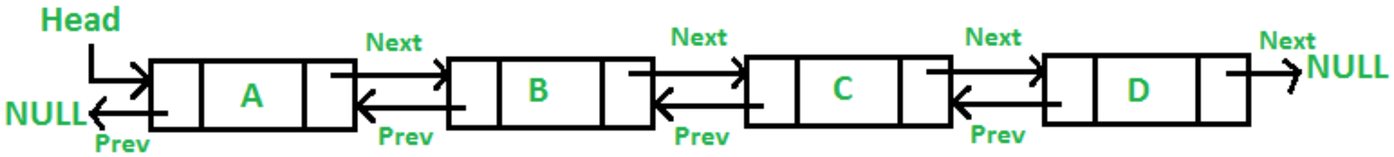
Hvorfor lærer vi oss lenkelister?



Enkeltlenket



Dobbeltlenket



Kodeeksempel

Comparable

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

Lett å sammenligne ting Java vet om, f.eks. int og string.

Når vi bruker == sammenligner vi refereransene til objekter, ikke objektenes faktiske innhold.

- Av den grunn har String equals()-metoden.

Comparable

Comparable er et interface

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

Inneholder en metode: compareTo()

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 Comparable

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

    public void settInn(T newElement){

        // ...

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

        // ...
    }
}
```

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**.

```
interface Liste<T> extends Iterable<T>{
    int size();
    void add(T x);
    void set(int pos, T x);
    T get(int pos);
    T remove(int pos);
}
```

```
public class Arrayliste<T> implements Liste<T>{
   // instansvariabler
    class ListeIterator implements Iterator<T> {
        private int pos = 0;
        @Override
        public T next() {
            pos++;
            return get(pos-1);
        }
        @Override
        public boolean hasNext() {
            return pos < size();
        }
    }
    // Andre metoder
```

LAB

TRIX-oppgaver tagget med 06-uke og 07-uke

Gruppe 8, ukesoppgaver uke 5 og uke 6
(andre grupper også)

Eksamensøving