# Velkommen!

...

Johanna
johannph på mattermost
johannph@uio.no på mail!

# Kort: Praktisk informasjon

- Undervisningstilbud
  - https://www.uio.no/studier/emner/matnat/ifi/IN1010/v21/undervisningstilbud/
  - Jeg har konkrete spørsmål/problemer med min kode -> Labtime!
  - Jeg vil ha mer liveprogrammering -> Plenumstime!
  - Jeg vil jobbe med andre (og kanskje en kjapp recap av forelesning) -> Gruppetime!
  - Jeg vil ha en recap av de vanskeligste konseptene fra forelesning -> Repetisjonsgruppe!
- Oblig 3, frist Mandag 1. mars kl 23.59
- Skriv alt dere lurer på i chatten enten til everybody eller bare til meg 😊
- Grupper oblig 4 er ute, håper dere har fått en gruppe, eller laget en selv!

# Mentimeter neste uke: repetisjon

https://www.menti.com/wpzy45bsao

# Repetisjon forrige uke

# Forrige uke

Beholdere

Lenkeliste

Generiske klasser

# Beholder

Et objekt som representerer en gruppe objekter av samme type.

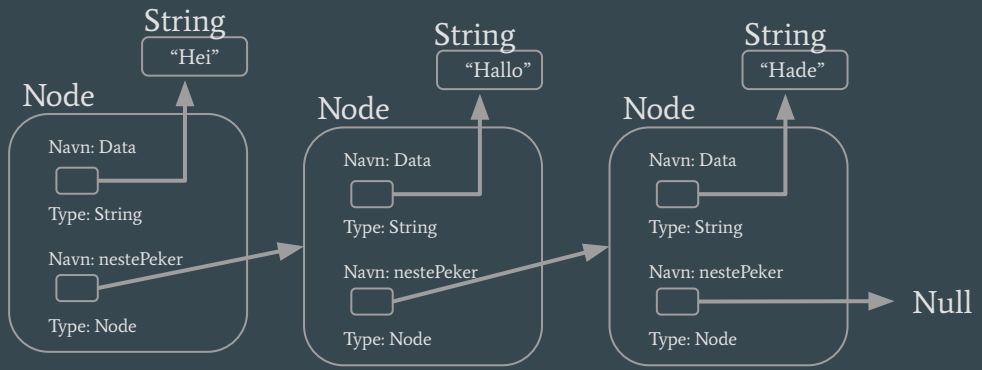
Ofte metoder for å legge til, hente ut og finne størrelse.

Eksempler på beholdere: array(har ingen metoder), ArrayList, HashMap, Lenkeliste

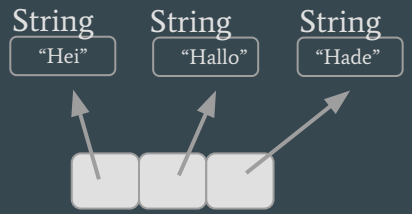
```
34
35      NavnKlasseEllerInterface[] navnVariabel = new NavnKlasseEllerInterface[lengdeArray];
36      ArrayList<NavnKlasseEllerInterface> navnVariabel = new ArrayList();
37      HashMap<NavnKlasseEllerInterfaceNokkel, NavnKlasseEllerInterfaceVerdi> navnVariabel = new HashMap();
38
```

# Lenkeliste

Vi skal se på en spesiell type beholder: lenkeliste

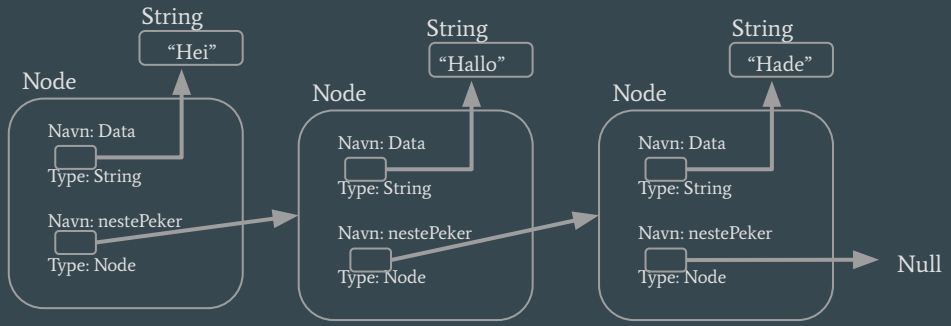


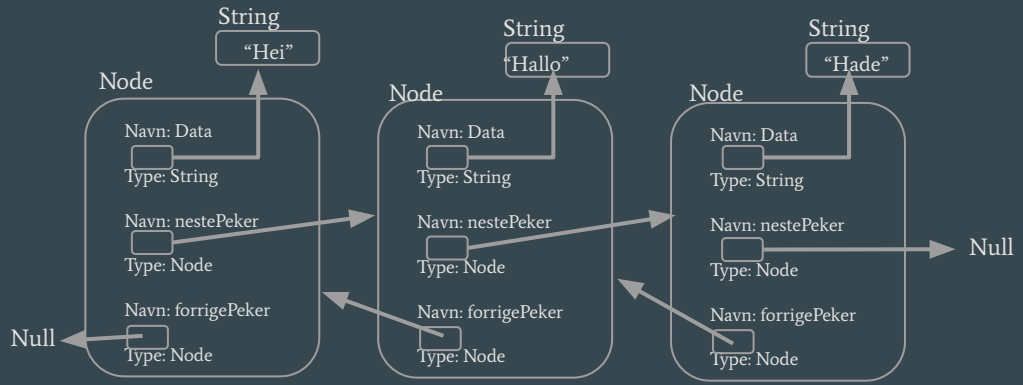

Lenkeliste




ArrayList eller Array f.eks.

# Enkeltlenket vs dobbeltlenket lenkeliste



# Generisk klasse Lenkeliste

Generisk lenkeliste, der Node er en indre klasse i Lenkeliste, da slipper vi å sende inn Noder men kun verdien.

```java
class Lenkeliste<T>{

    private Node start;

    private class Node{
        T data;
        Node neste;
    }

    public void leggTil(T nyData){
        //legget til ny data
    }

    public void slettData(T slettData){
        //sletter data
    }

    public T hentData(T data){
        //henter node med gitt data
    }
}
```

# Slette objekter

Når vi sletter objekter så må vi bare slette alle pekere til det objektet.

Objekter er litt som heliumballonger, har vi ikke festet en tråd i dem flyr de av gårde og forsvinner(eller blir søppel da).

# Hvordan skal jeg implementere lenkeliste??!

TEGN TEGN TEGN!

Gå gjennom koden din og tegn alt som skjer!

# Repetisjon denne uken

# I dag

Stabel

Kø

Litt mer om lenkelister

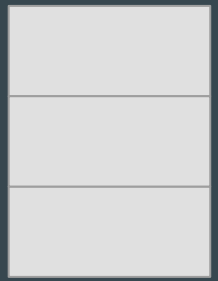
Comparable og compareTo

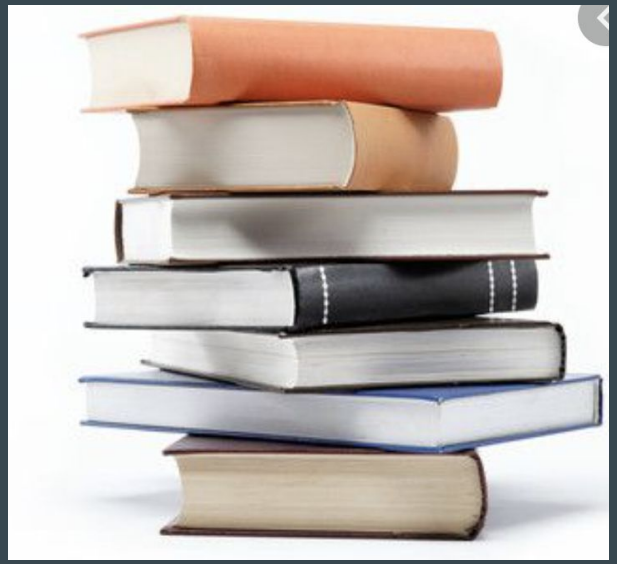
Katt som lenkeliste: Iterable og iterator

# Stabel/Stack: LIFO (Last In First Out)

Det siste vi legger inn er det første vi tar ut:

push:

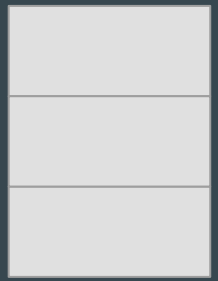


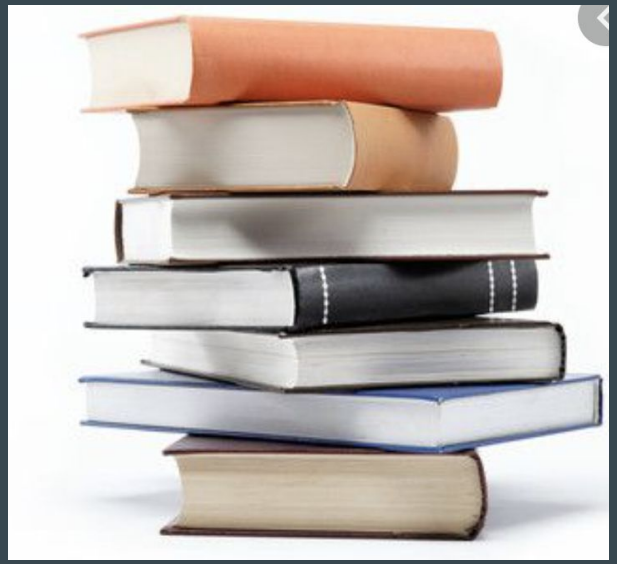


# Stabel/Stack: LIFO (Last In First Out)

Det siste vi legger inn er det første vi tar ut:

push:

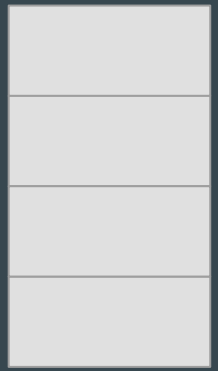


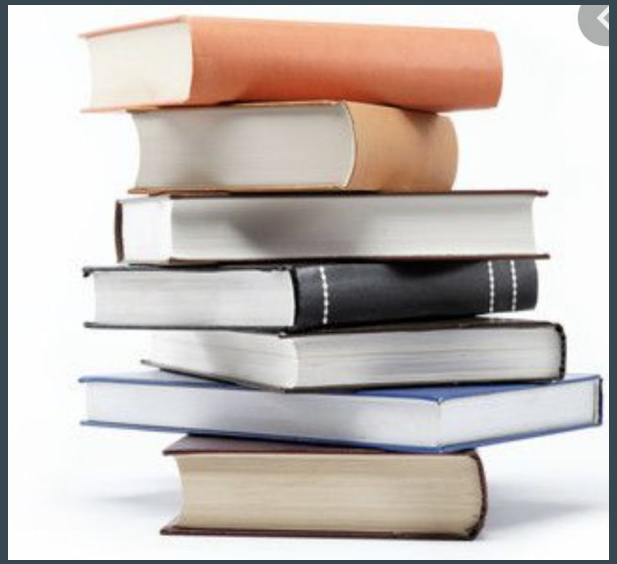


# Stabel/Stack: LIFO (Last In First Out)

Det siste vi legger inn er det første vi tar ut:

push:



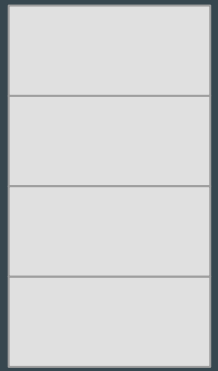


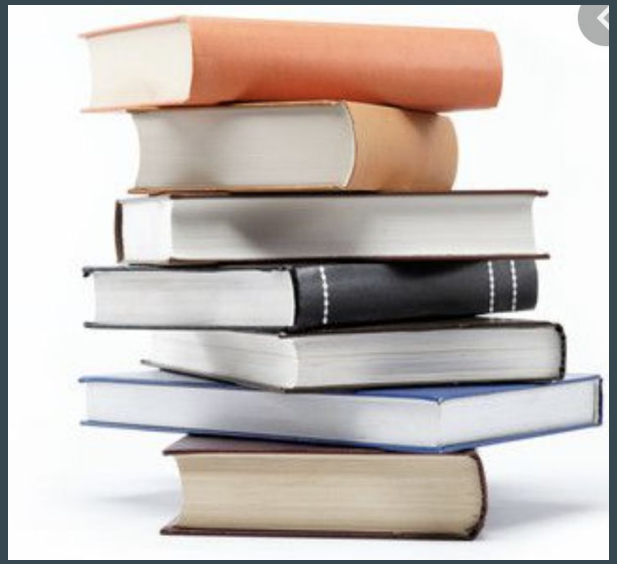
# Stabel/Stack: LIFO (Last In First Out)

Det siste vi legger inn er det første vi tar ut:

push:

pop:



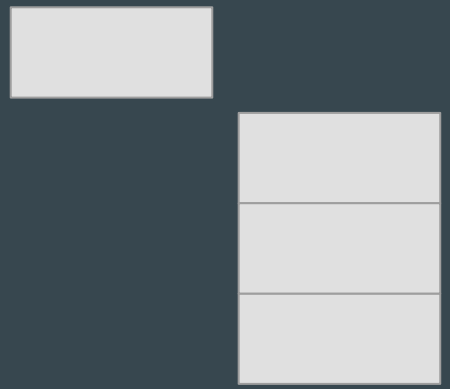


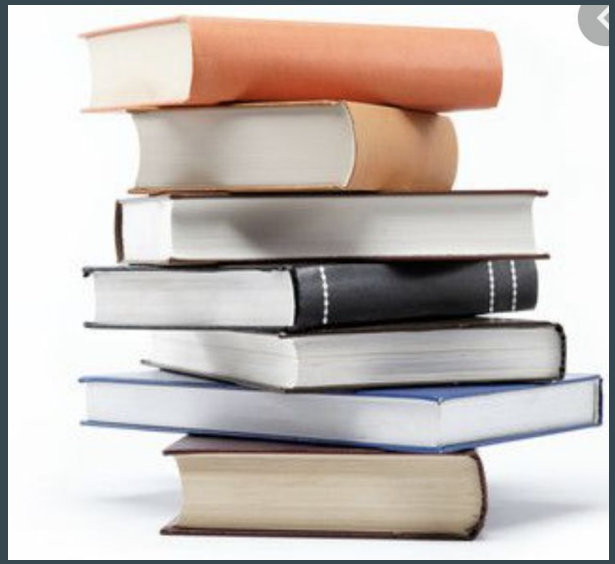
# Stabel/Stack: LIFO (Last In First Out)

Det siste vi legger inn er det første vi tar ut:

push:

pop:



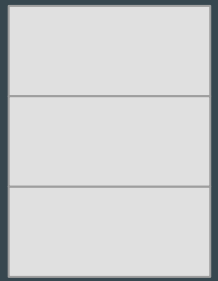


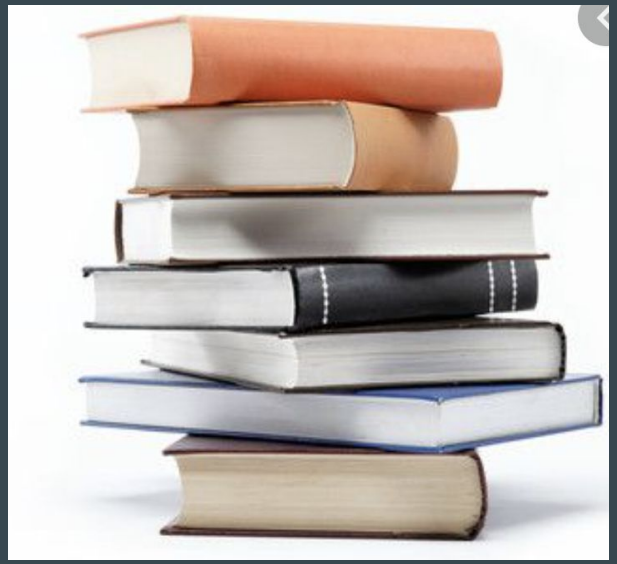
# Stabel/Stack: LIFO (Last In First Out)

Det siste vi legger inn er det første vi tar ut:

push:

pop:





# Kø/queue: FIFO (First In First Out)

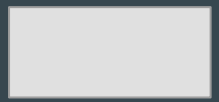
Det første vi legger inn er det første vi tar ut:

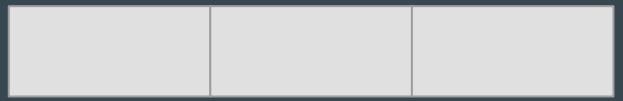
add:

# Kø/queue: FIFO (First In First Out)

Det første vi legger inn er det første vi tar ut:

add:





# Kø/queue: FIFO (First In First Out)

Det første vi legger inn er det første vi tar ut:

add:

# Kø/queue: FIFO (First In First Out)

Det første vi legger inn er det første vi tar ut:

add:

remove:

# Kø/queue: FIFO (First In First Out)

Det første vi legger inn er det første vi tar ut:

add:

remove:

# Kø/queue: FIFO (First In First Out)

Det første vi legger inn er det første vi tar ut:

add:

remove:

# Interfacet Comparable og Metoden CompareTo

Hvis en klasse implements Comparable betyr det at vi kan gjøre sånne operasjoner:

objekt1.compareTo(objekt2)
objekt1.compareTo(objekt2)
objekt1.compareTo(objekt2)

f.eks. klassen Integer og String implementerer comparable:

“a”.compareTo(“b”)  “a”.compareTo(“b”)  “a”.compareTo(“b”)

# Implementere compareTo

Livekoding med dyr sammenlignet på alder.

Hva burde man sammenligne på?
- Man må se hva som er hensiktsmessig i sitt program eller for sin klasse
- For katter kunne det vært navn, alder, id eller noe helt annet

Se vedlagt kode: Dyr.java og TestDyr.java

# Comparable og compareTo()

```java
import java.util.Arrays;

class TestDyr{
    public static void main(String[] args){
        Dyr dyr1 = new Dyr("Karl", 3);
        Dyr dyr2 = new Dyr("Bob", 1);
        Dyr dyr3 = new Dyr("Fia", 2);

        Dyr[] alleDyr = {dyr1, dyr2, dyr3};
        System.out.println(Arrays.toString(alleDyr));
        Arrays.sort(alleDyr);
        System.out.println(Arrays.toString(alleDyr));
        System.out.println(dyr1.compareTo(dyr2) > 0);

```

```
jonbon@jons-macbook-pro uke7 % java TestDyr
[Karl, Bob, Fia]
[Bob, Fia, Karl]
true
```

```java
class Dyr implements Comparable<Dyr>{
    protected String navn;
    protected int alder;

    public Dyr(String navn, int alder){
        this.navn = navn;
        this.alder = alder;
    }
    public int hentAlder(){
        return alder;
    }
    public String toString(){
        return navn;
    }
    @Override
    public int compareTo(Dyr dyr){
        if (alder == dyr.hentAlder()){
            return 0;
        }
        else if (alder < dyr.hentAlder()){
            return -1;
        }
        return 1;
    }
```

# Interface Iterable og interfacet Iterator

Disse finnes i java biblioteket!! Vi skal ikke skrive dem!

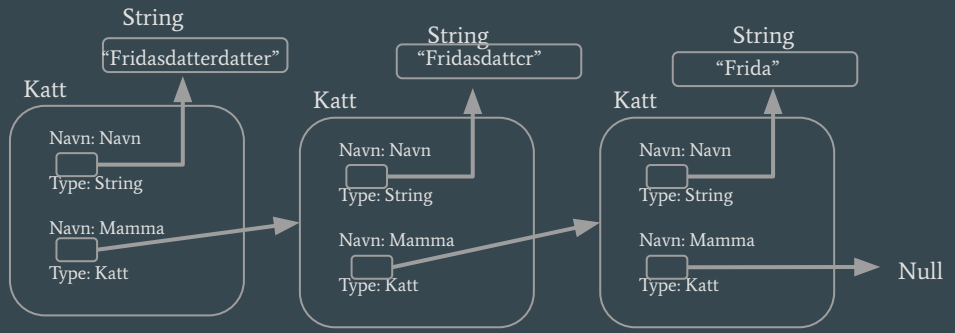
```
public interface Iterator<T>{
  boolean hasNext();
  T next();
}
public interface Iterable{
  Iterator iterator();
}
```

Hvis vi ønsker å lage en beholder der man f.eks. kan bruke en for-each-loop må beholderen vår implement interface Iterable altså implementere metoden iterator(). Metoden skal returnere et objekt av en klasse som implements interfacet Iterator. Den klassen må vi også skrive!

# Implementere Iterator-klasse og iterator()-metode

Vi skal lage en lenkeliste av klassen katt!

Hver katt får en mamma, men for noen katter har vi ikke informasjon om deres mamma, så da er den bare null. Hvis vi vet mammaen i flere generasjoner så får vi en lenkeliste! Katt skal implementere interfacet Iterable (ha en iterator() metode)




```java
public interface Iterator<T>{
  boolean hasNext();
  T next();
}
public interface Iterable{
  Iterator iterator();
}
```

# Hovedprogrammet

```
class TestKattLenkeliste{
    public static void main(String[] args) {

        // Opretter tre katter
        // public Katt(String navn, int alder, String favorittmat, Katt mamma){
        Katt dyr1 = new Katt("Frida", 15, "sushi", null);
        Katt dyr2 = new Katt("Fridasdatter", 10, "Torsk", dyr1);
        Katt dyr3 = new Katt("Fridasdatterdatter", 2, "Koie", dyr2);

        // Lager KattLenkeliste
        // public KattLenkeliste(Katt start)
        // argumentet til konstruktør er den første katten i lekelisten
        KattLenkeliste katteSlekt = new KattLenkeliste(dyr3);

        // Kan nå iterere gjennom lenkelisten
        for (Katt katt : katteSlekt){
            System.out.println(katt);
```

# Katt

```java
class Katt extends Dyr{
  protected String favorittmat;
  protected Katt mamma;

  public Katt(String navn, int alder, String favorittmat, Katt mamma){
    super(navn, alder);
    this.favorittmat = favorittmat;
    this.mamma = mamma;
  }
  public String hentFavorittMat(){
    return favorittmat;
  }
  public String hentNavn(){
    return navn + " favorittmaten min er " + favorittmat;
  }
  public Katt hentMamma(){
    return mamma;
  }
```

# Kattlenkeliste implements Iterable og KattIterator implements Iterator

```java
class KattLenkeliste implements Iterable<Katt>{
    private Katt start;

    public KattLenkeliste(Katt start){
        this.start = start;
    }
    public KattIterator iterator(){
        return new KattIterator(start);
    }
}
```

```java
import java.util.Iterator;

class KattIterator implements Iterator<Katt>{
    private Katt kattPeker;

    public KattIterator(Katt start){
        kattPeker = start;
    }
    public Katt next(){
        Katt returKatt = kattPeker;
        kattPeker = kattPeker.hentMamma();
        return returKatt;
    }
    public boolean hasNext(){
        return kattPeker != null;
    }

}
```

# Mentimeter neste uke

https://www.menti.com/wpzy45bsao

# Jobbe selv

Jobb med hva dere vil og rekk opp hånda hvis dere trenger hjelp med noe/har spørsmål så møtes vi i breakoutroom! 😊