Rep. Kurs TRÅDER

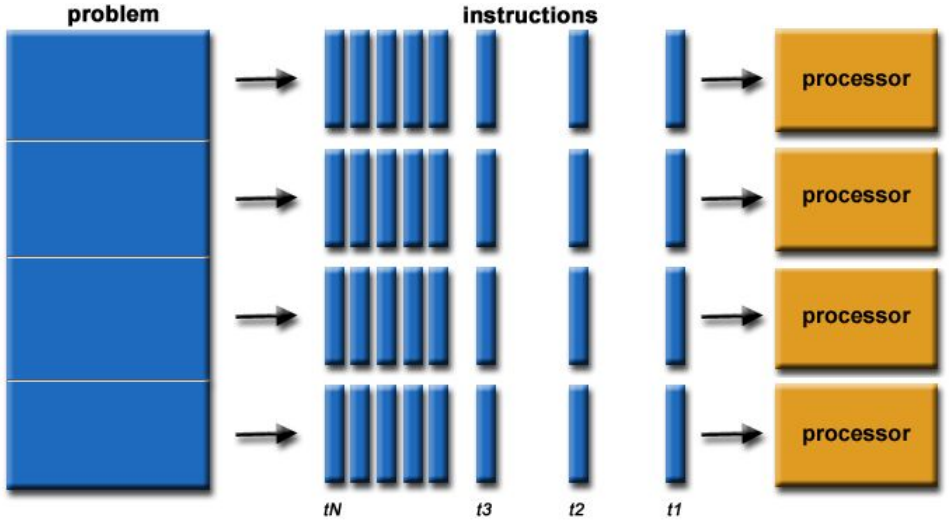
Foiler av: Marlen Jarholt (marleja@ifi.uio.no)

Plan for timen

1. Repetisjon av stoff
2. Jobbe med oppgaver
3. Felles gjennomgang av noen av oppgavene

Hva er et parallelt program og hvorfor bruker vi det

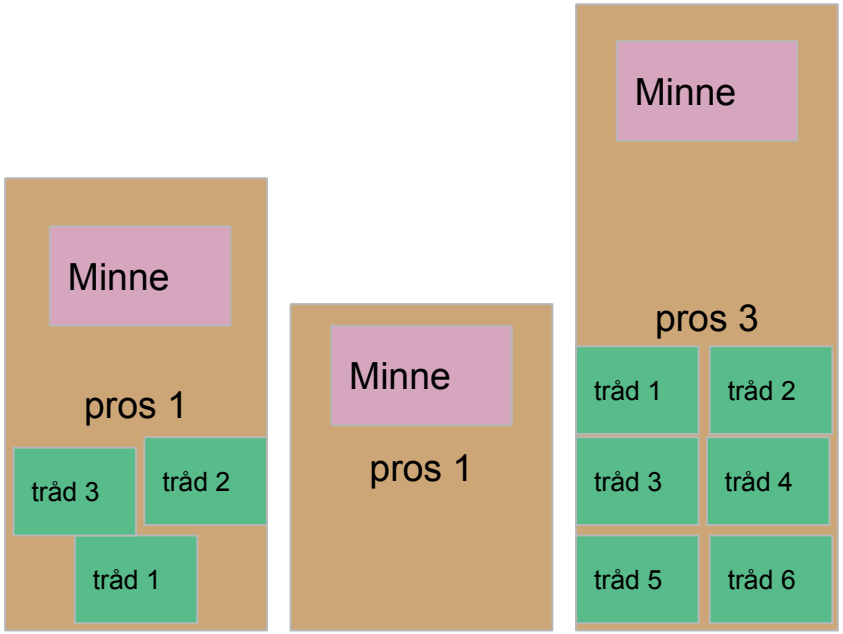
- CPU/ Kjerner
- Flere programmer for kjørt
- Maskinen har kapasiteten
- Dele opp problemet
- Raskere, men
  - Det tar tid å starte opp en tråd
  - Synkronisering



Bildekilde: https://computing.llnl.gov/tutorials/parallel_comp/

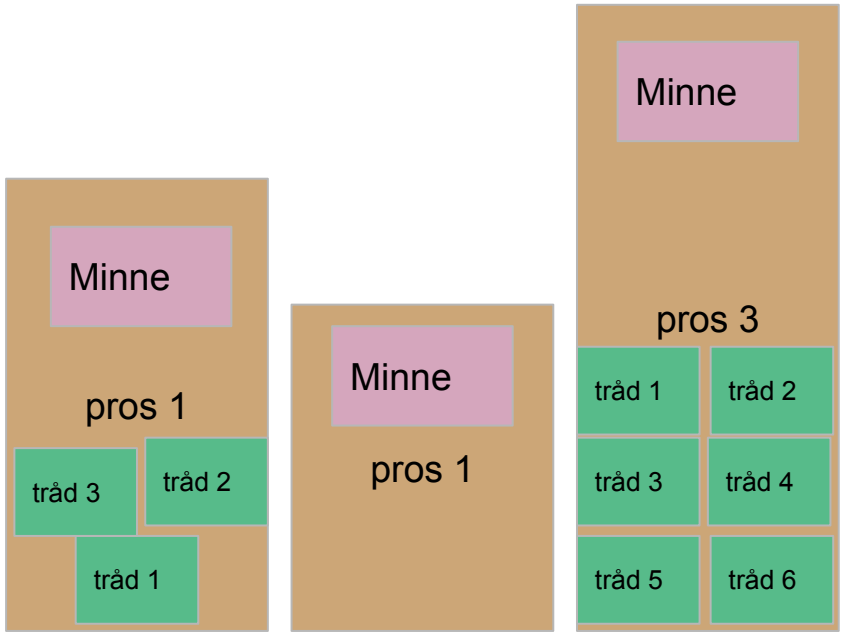
Hva er en tråd?

- En parallell eksekvering inne i en prosess
  - En prosess er utføringen av et program
- Deler minne til prosessen
- Tråder kan gi i ekte parallel
- “Små”-prosesser i en vanlig prosess
- En sekvens med instruksjoner



Delt data

- Data flere tråder deler
- Prosessens data
- Skriving
- Lesing
- Lesing og skriving
- Er du usikker lås



Hvordan lage en tråd ??

- Runnable steg 1
  - https://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html
  - Et interface
  - Vi må lage en klasse som implementerer dette interfacet
  - run()
    - Når du bruker .start() på en tråd vil objektets run metode bli kalt
- Thread steg 2
  - https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html#run()
  - En klasse
  - start()
  - Tar inn et runnable objekt i konstruktøren

Live-programmering (del 1)

Lag et program som starter opp 3 tråder. Trådene skal ta inn en id og printe denne id-en 9 ganger.

Locks

- For å bruke en Lock må man importer
  - java.util.concurrent.locks.Lock (Et grensesnitt, bl.a. to metoder lock() og unlock())
  - java.util.concurrent.locks.ReentrantLock (Lager de faktiske låseobjekten)
- Passer på at kun en tråd kan utføre koden om gangen
- Når man bruker Lock med lock() og unlock() er det viktig at man bruker en try-catch blokk. Nå må man også huske finally.
  - Finally er en kodesnutt som vil skje uansett. Når det kommer til Locks må man passe på å alltid unlock() i en finally blokk slik at hvis det skulle skje noe med en tråd kan de andre trådene få fortsette

```
46    public void metode(){
47      lock.lock()
48      try{
49        <kode her>
50      }catch(Exception e){
51        <hvis det er en exception og catche>
52      }finally{
53        lock.unlock() //vil alltid skje
54      }
```

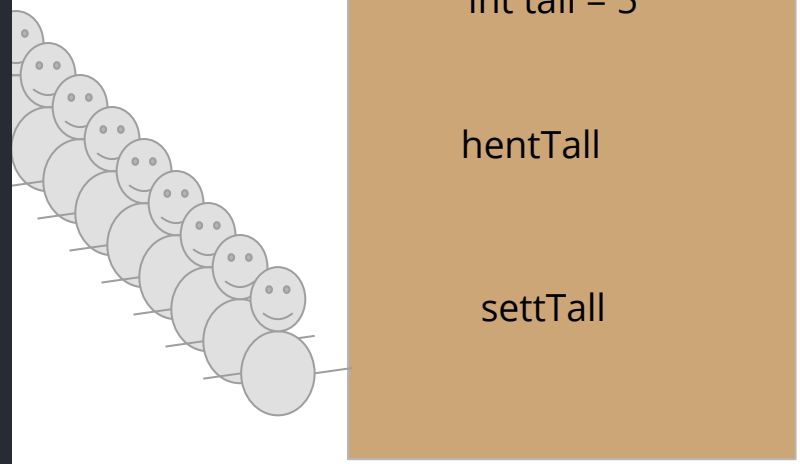
Monitor

- Et objekt som innkapsler den delta dataen
- Fungere som en type beskytter av felles data
- Monitor objektet har metoder som gjør at de andre klassen f.eks. kan endre data og hente ut data

PS: Den metoden blir brukt i mye i 1010 fordi det er en god objektorientert måte å gjøre synkronisering på

Bruk monitor design prinsippet

```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```

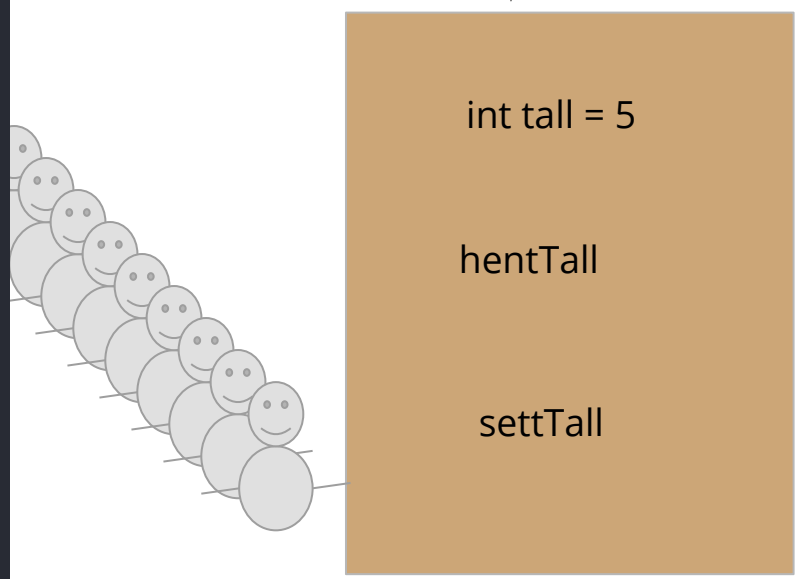


Monitor klasse

Bruk monitor design prinsippet

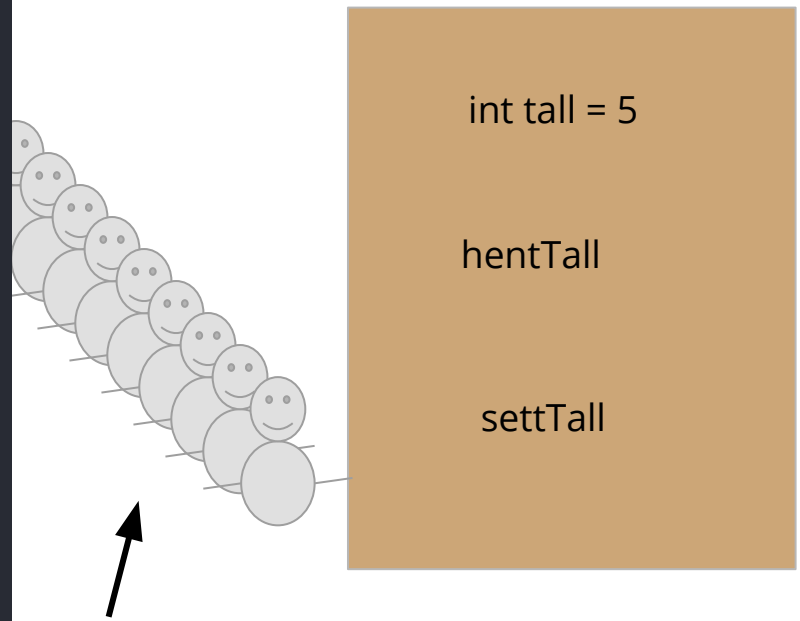
```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```

Forsøk på å illustrere monitor/phone boot/toalett



Bruk monitor design prinsippet

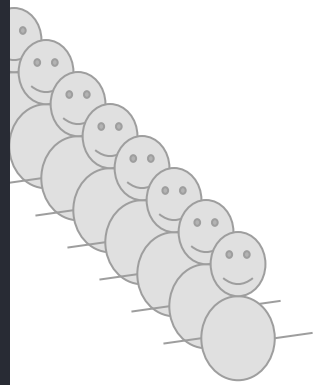
```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```

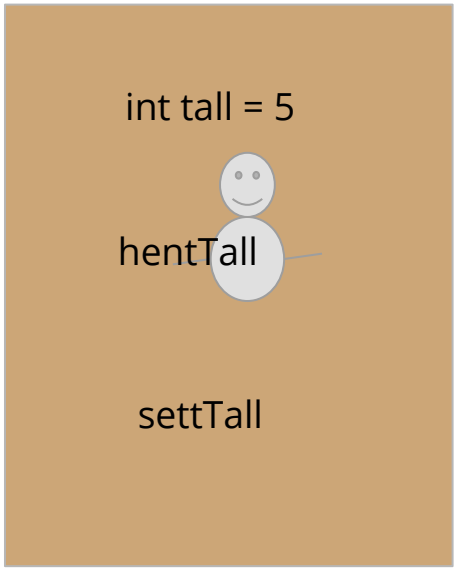


Alle trådene som aksesserer data i monitoren én og én

Bruk monitor design prinsippet

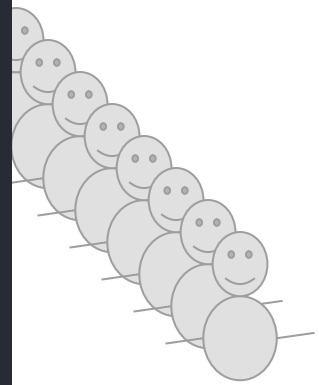
```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```

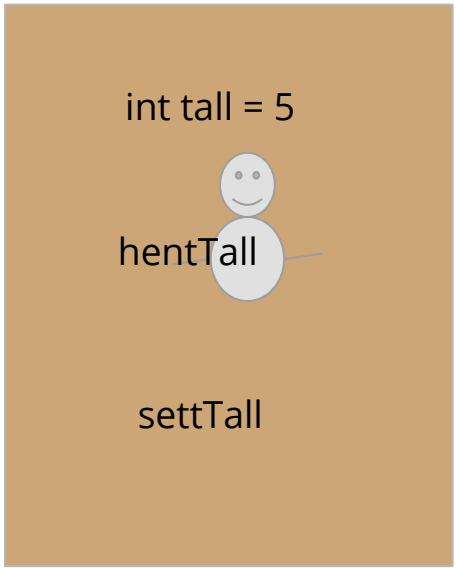




Bruk monitor design prinsippet

```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```

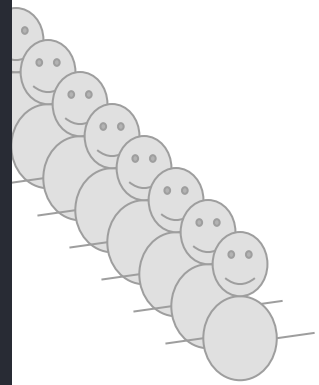


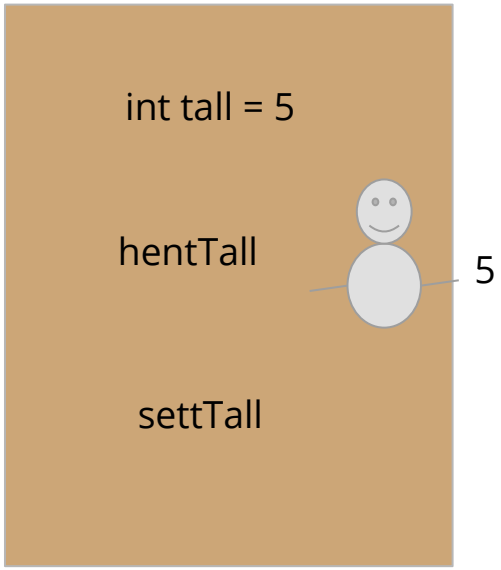


Låst fra noen tar lock() til de tar .unlock()

Bruk monitor design prinsippet

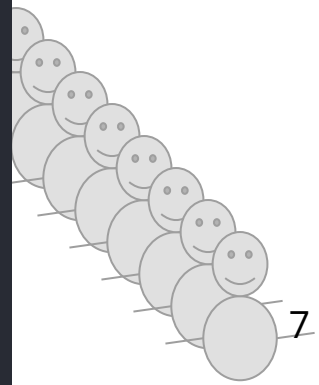
```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```





Bruk monitor design prinsippet

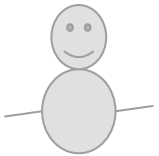
```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```



int tall = 5

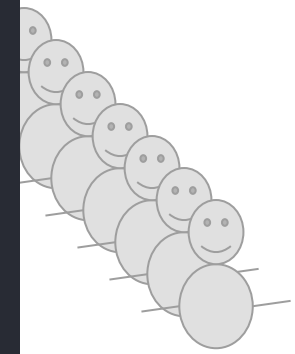
hentTall

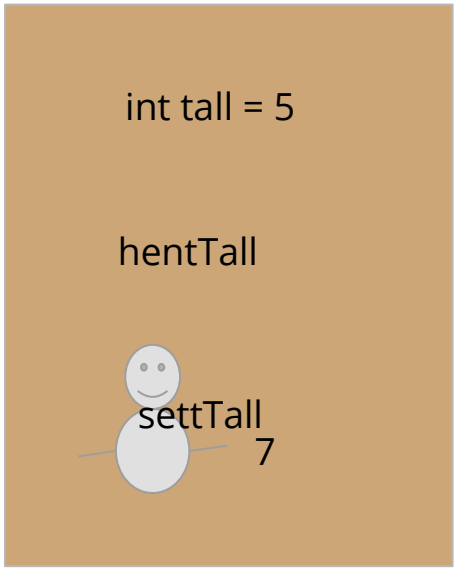
settTall



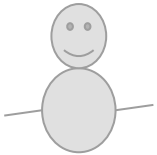
Bruk monitor design prinsippet

```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```



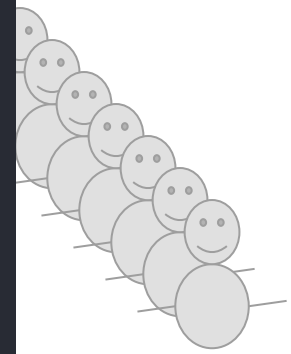


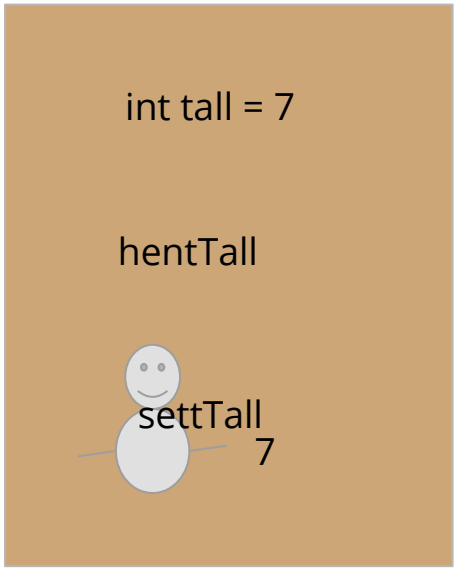
Låst fra noen tar lock() til de tar .unlock()

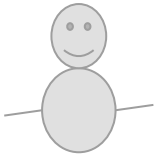


Bruk monitor design prinsippet

```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```

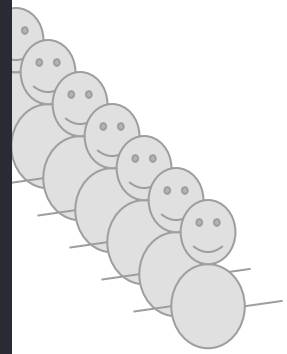






Bruk monitor design prinsippet

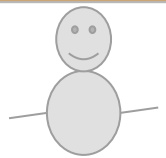
```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class tallBeholder{
  private Lock laas = new ReentrantLock();
  private int tall;

  public void settTall(int tall){
    laas.lock();
    try {
      this.tall = tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
  }
  public int hentTall(){
    laas.lock();
    try {
      return tall;
    } catch(Exception e) {
      System.out.println("Feil i monitor tallBeholder");
    } finally {
      laas.unlock()
    }
```

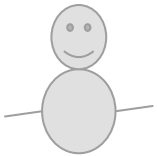


int tall = 7

hentTall

settTall

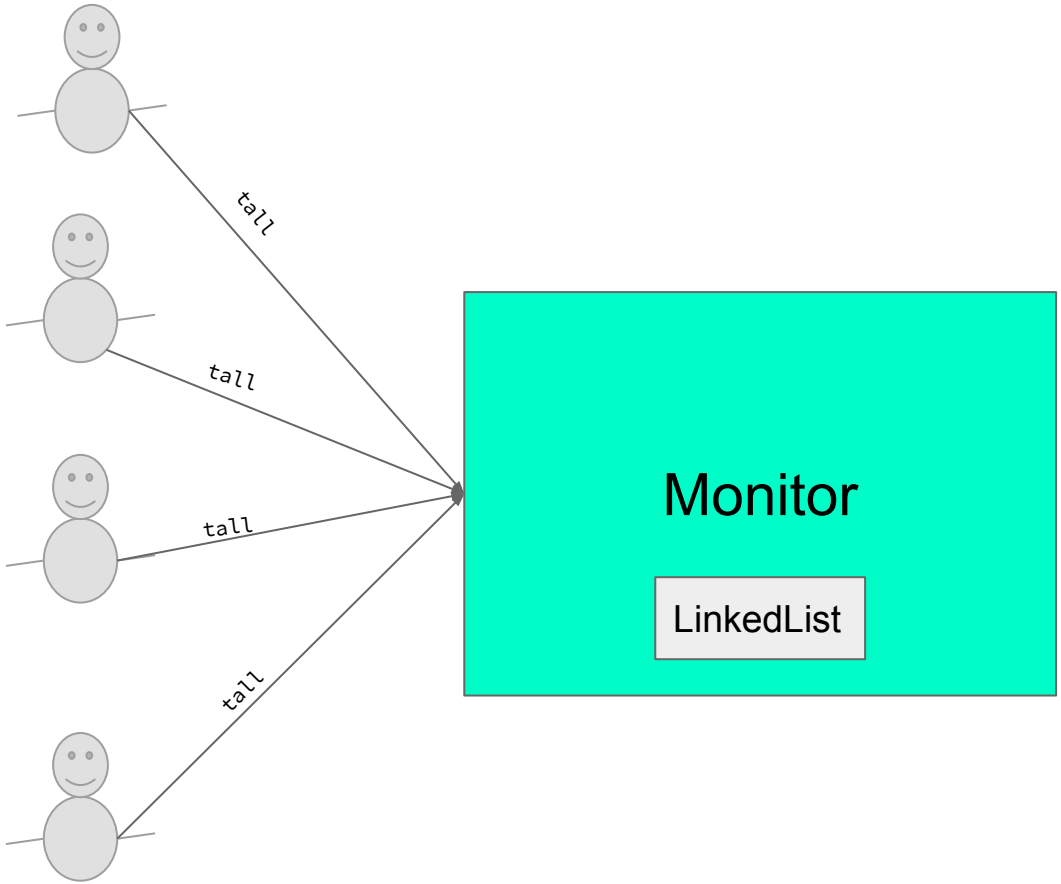




Live-programmering (del 2)

Fortsett fra del 1.

I stedet for å printe ut id sin skal trådene nå legge til id/tallet i en felles linkedlist. Print ut en beskjed hver gang en tråd legger til et nytt tall i lista


tall
tall
tall
tall
Monitor
LinkedList

Condition

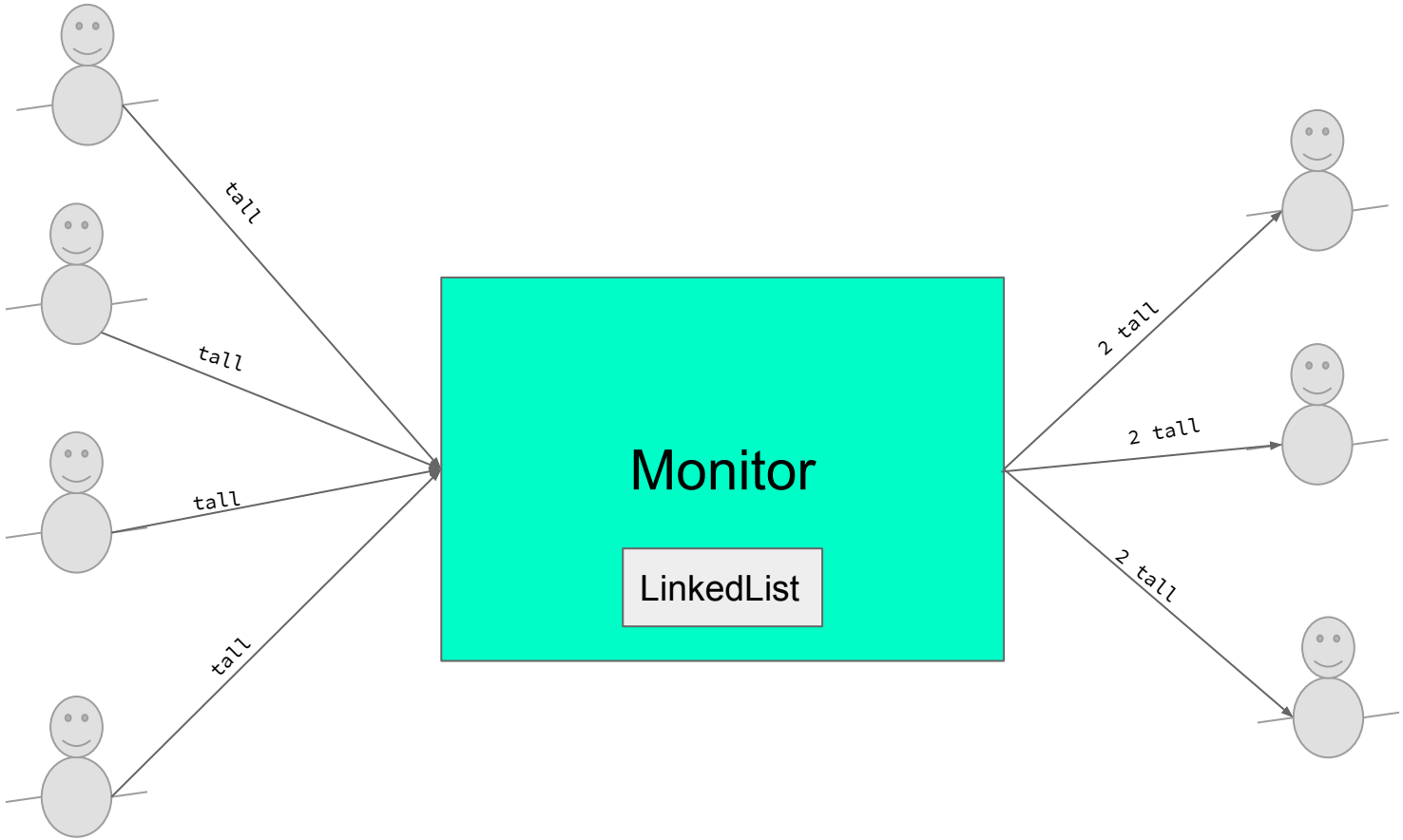
- En form for passiv venting
  - Vil bruke mindre ressurser enn aktiv venting
- await(): Får den tråden man er inne i nå til å vente til Condition gir signaliserer at det er greit å fortsette. (Eller at den blir interrupted)
- signal() : Signaliserer til en tråd om at den kan fortsett. (Som står å venter ved await())
- signalAll(): Signaliserer til alle ventene tråder

Live-programmering (del 3)

Fortsett fra del 3.

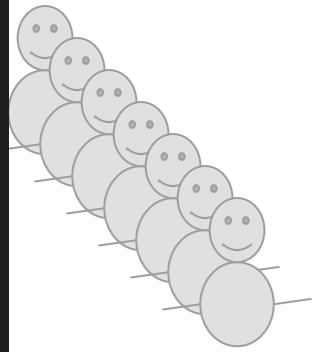
Lag en ny tråd-klasse, denne skal ta ut to tall fra monitoren og legge dem sammen før de printes ut.

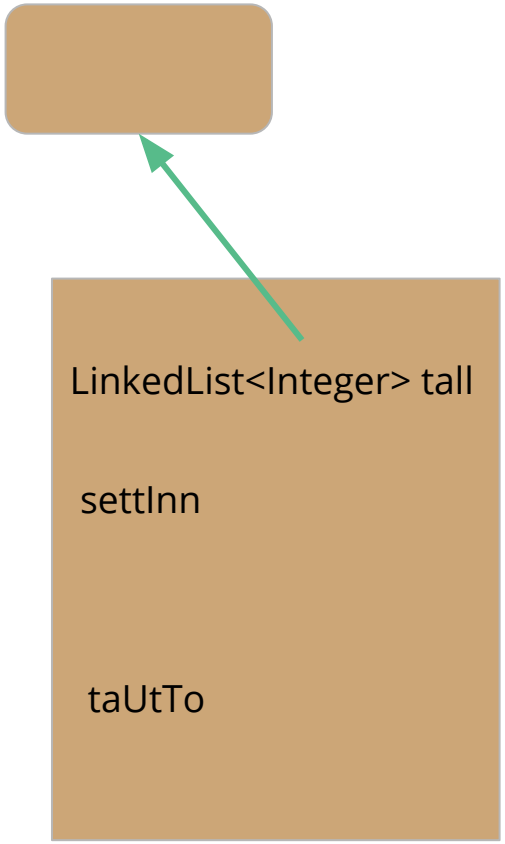
Hvis det ikke er to tall der må tråden vente, hvis det ikke vil komme flere tall og det ikke er flere tall i lista skal tråden terminere.


tall
tall
tall
tall
Monitor
LinkedList
2 tall
2 tall
2 tall

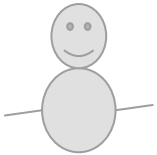
Bruk monitor design prinsippet

```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



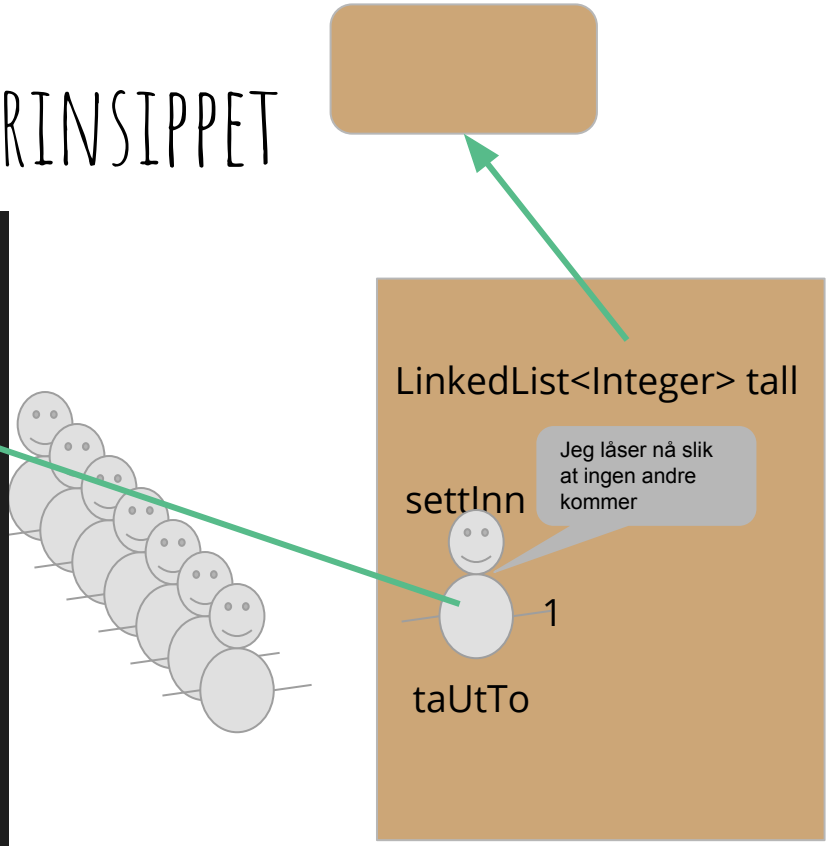


Full kode ligger ut på rep.kurs siden



Bruk monitor design prinsippet

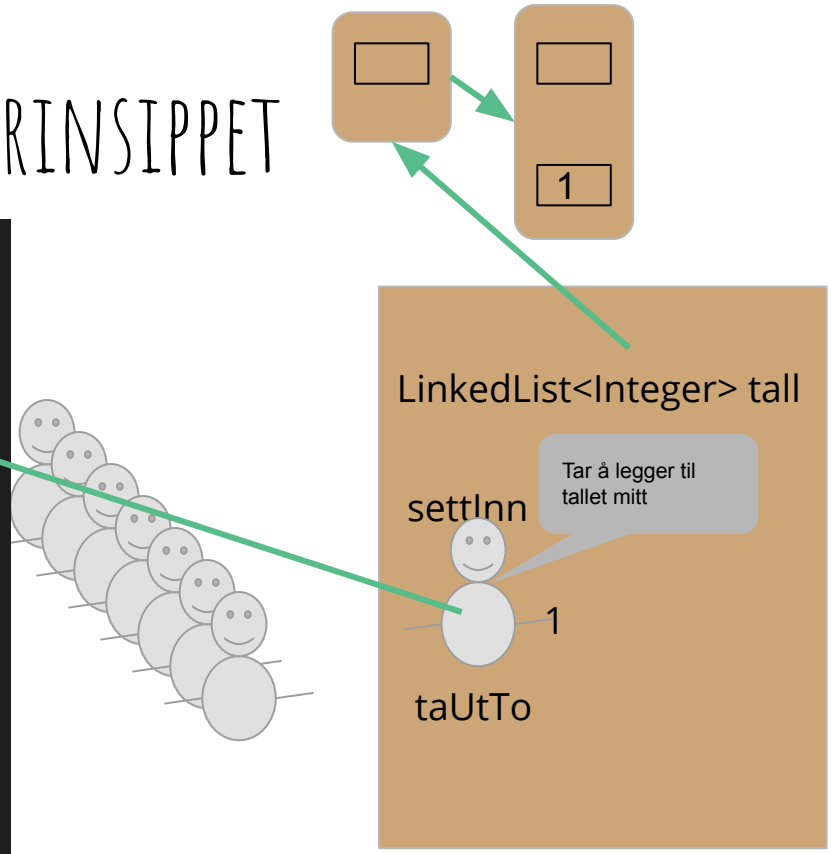
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

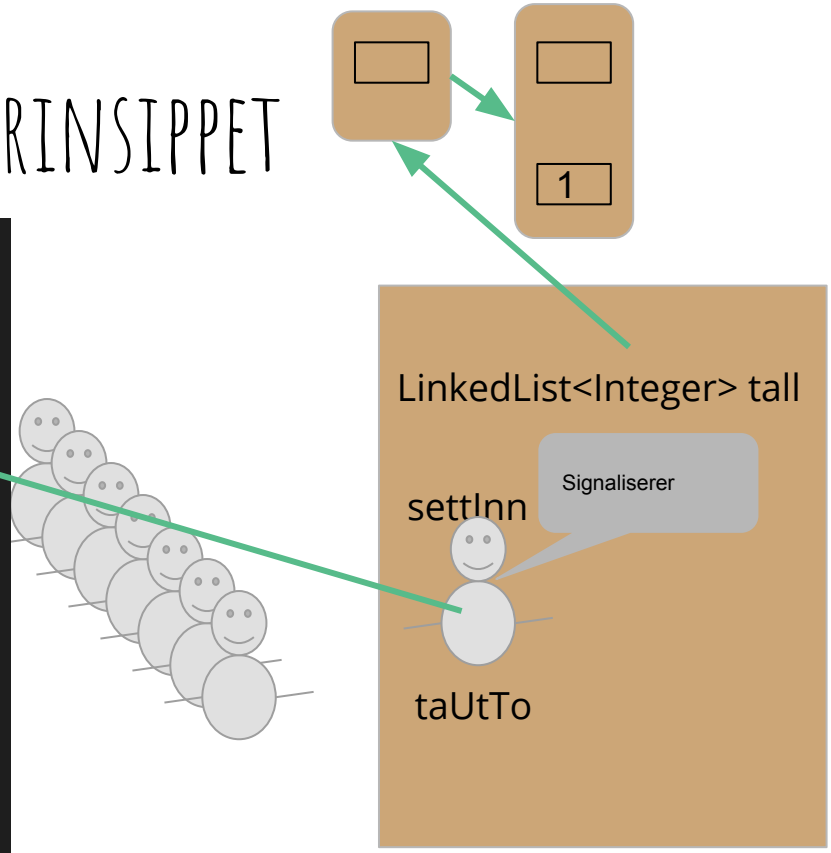
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

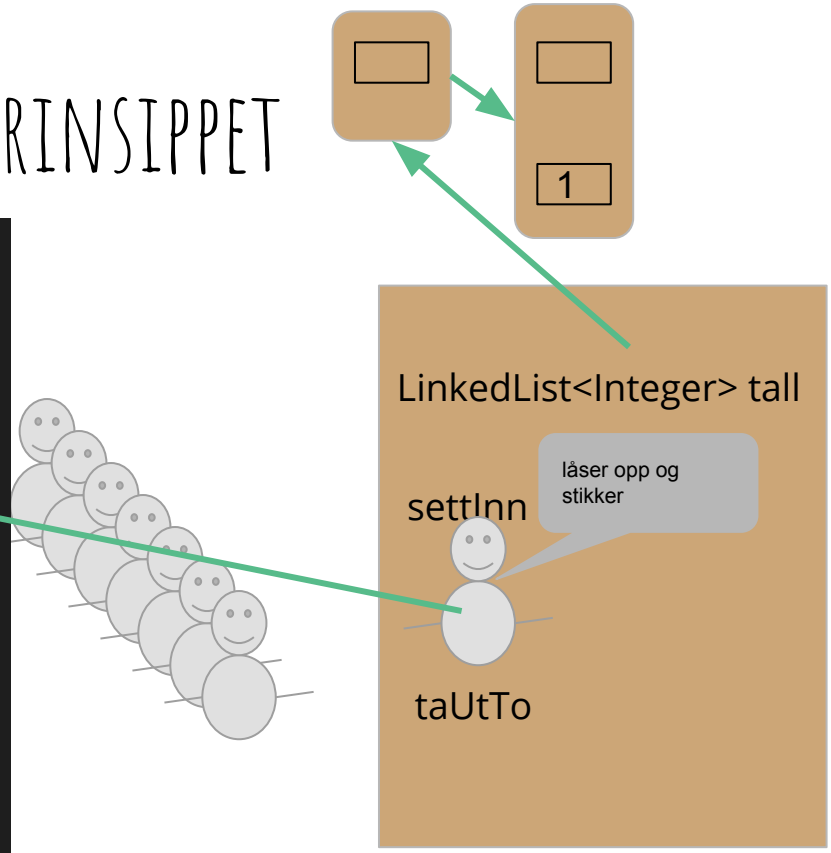
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

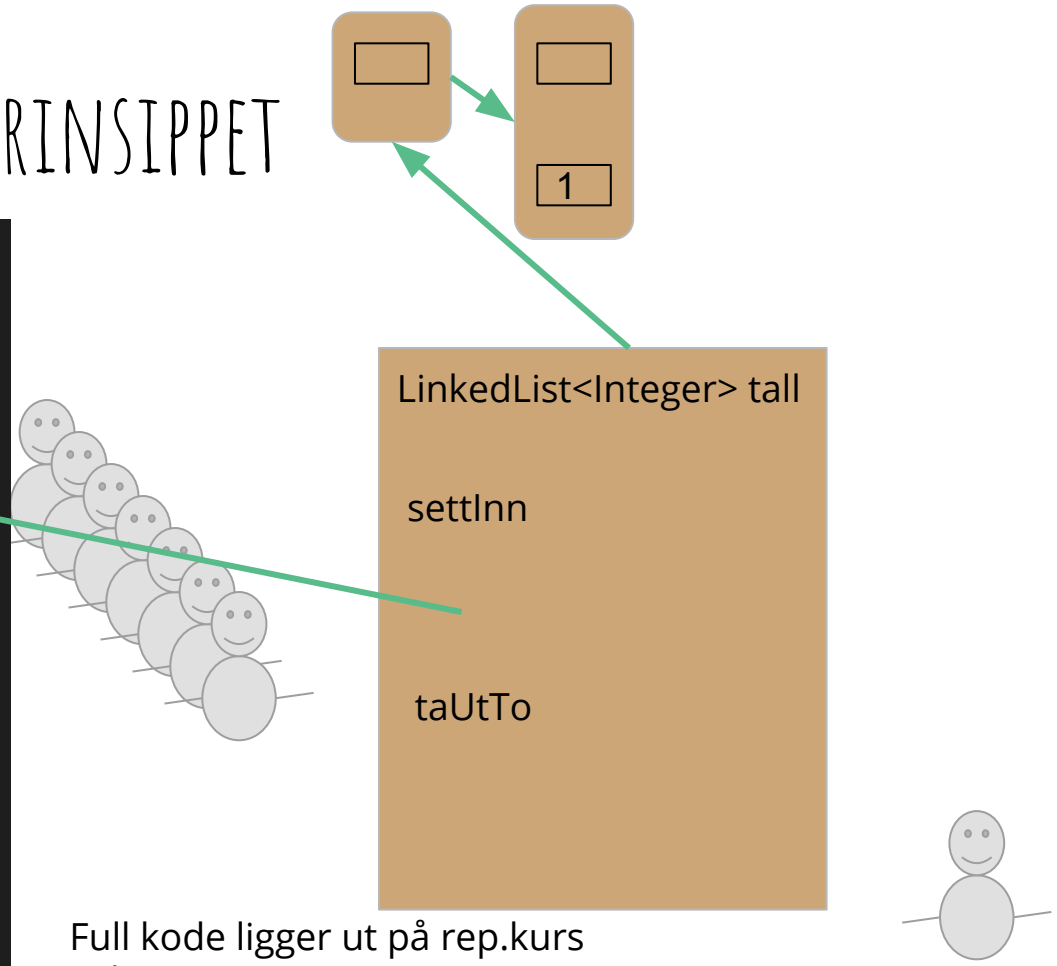
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

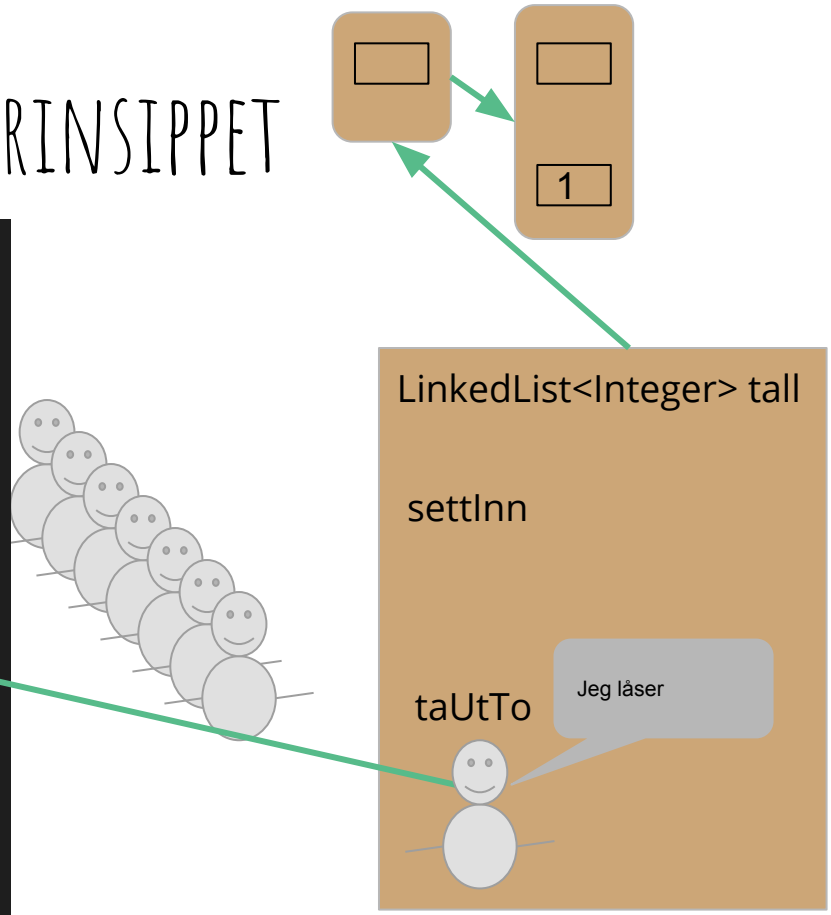
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

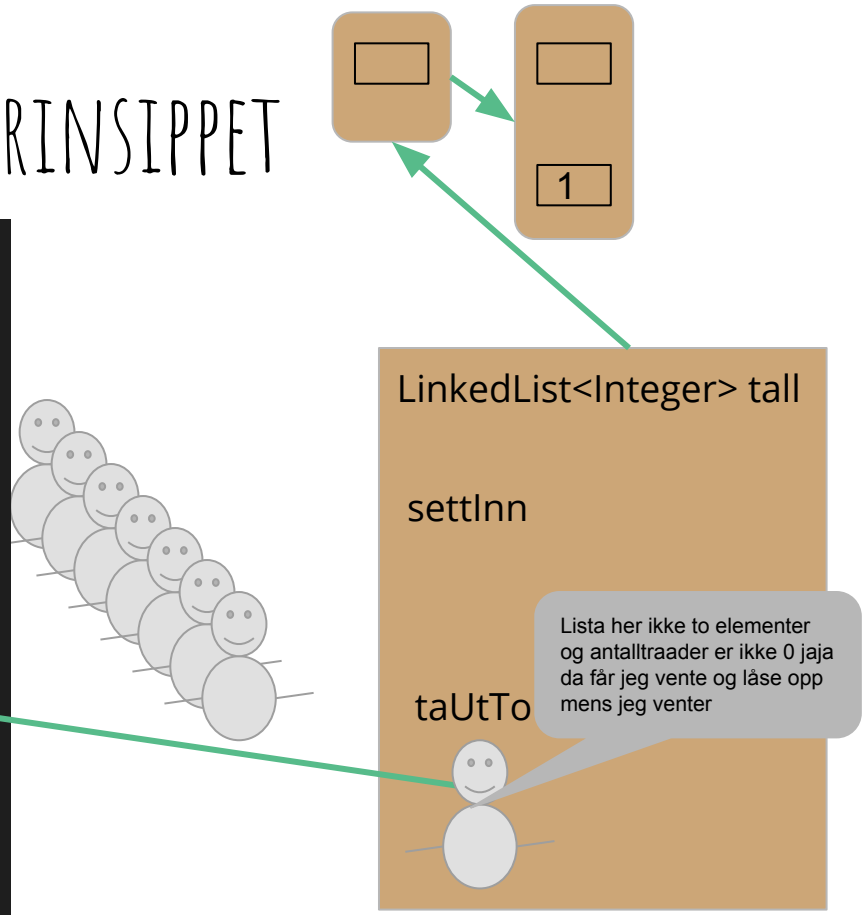
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

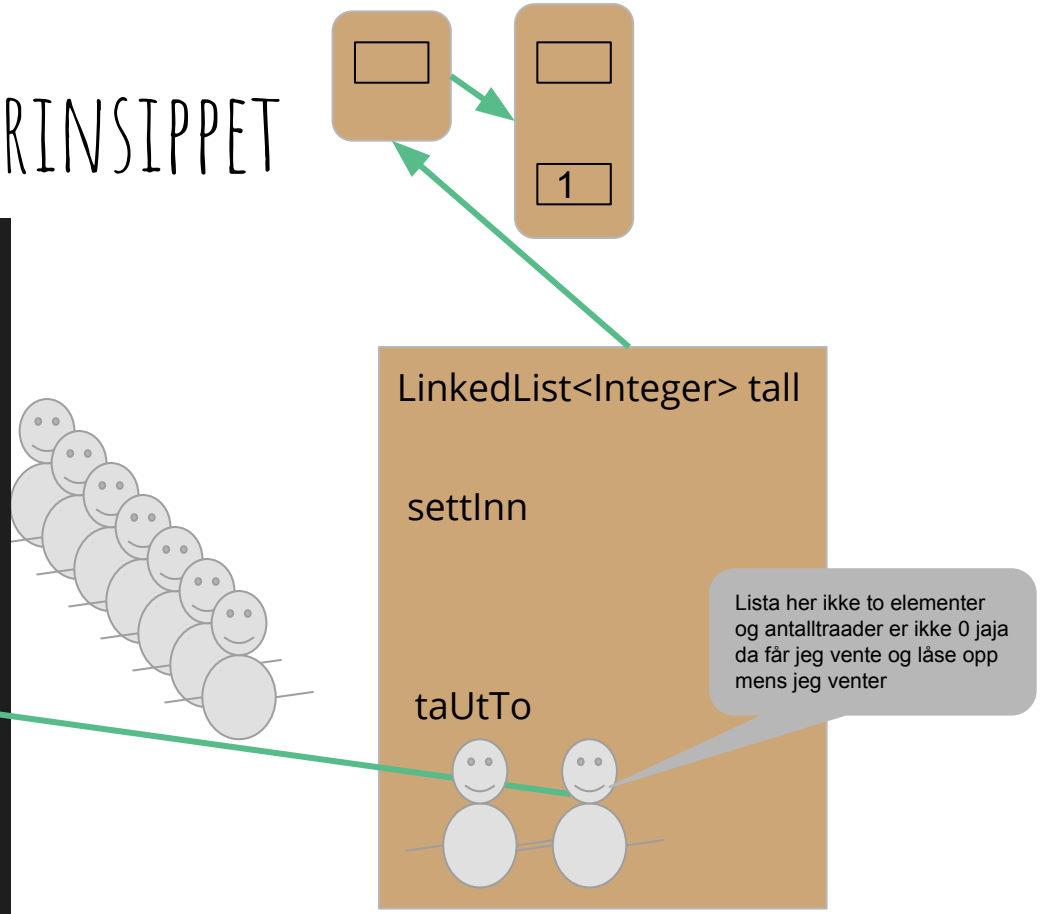
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

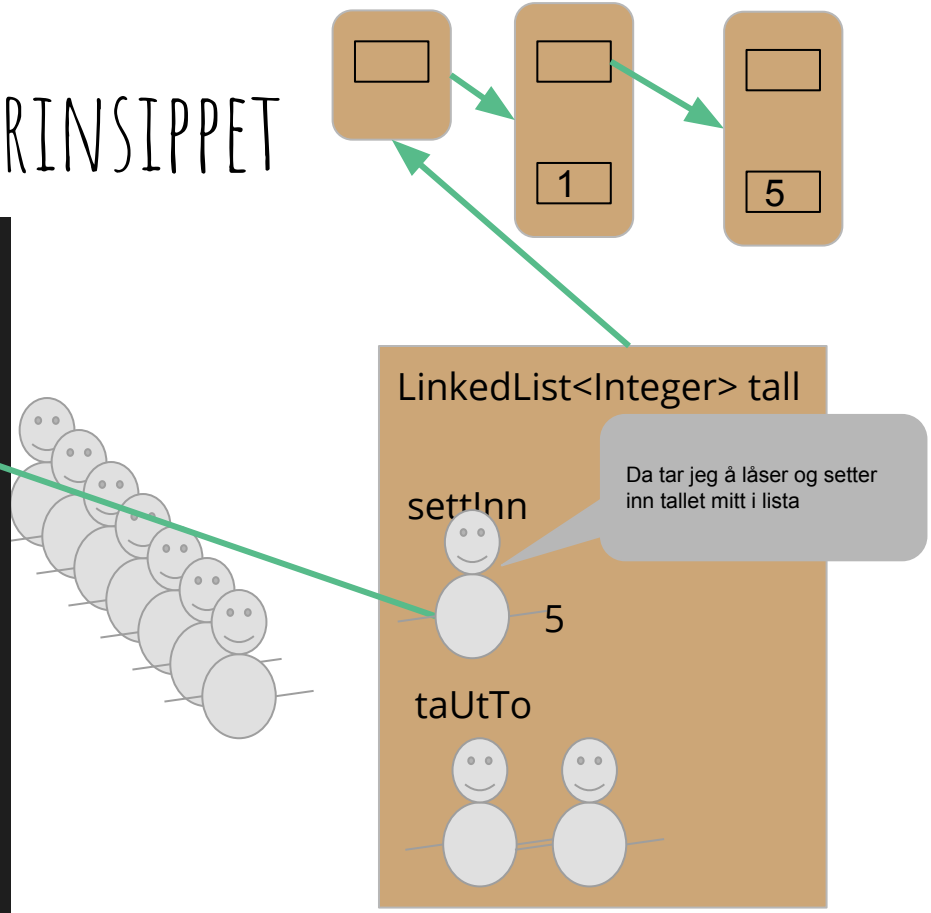
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

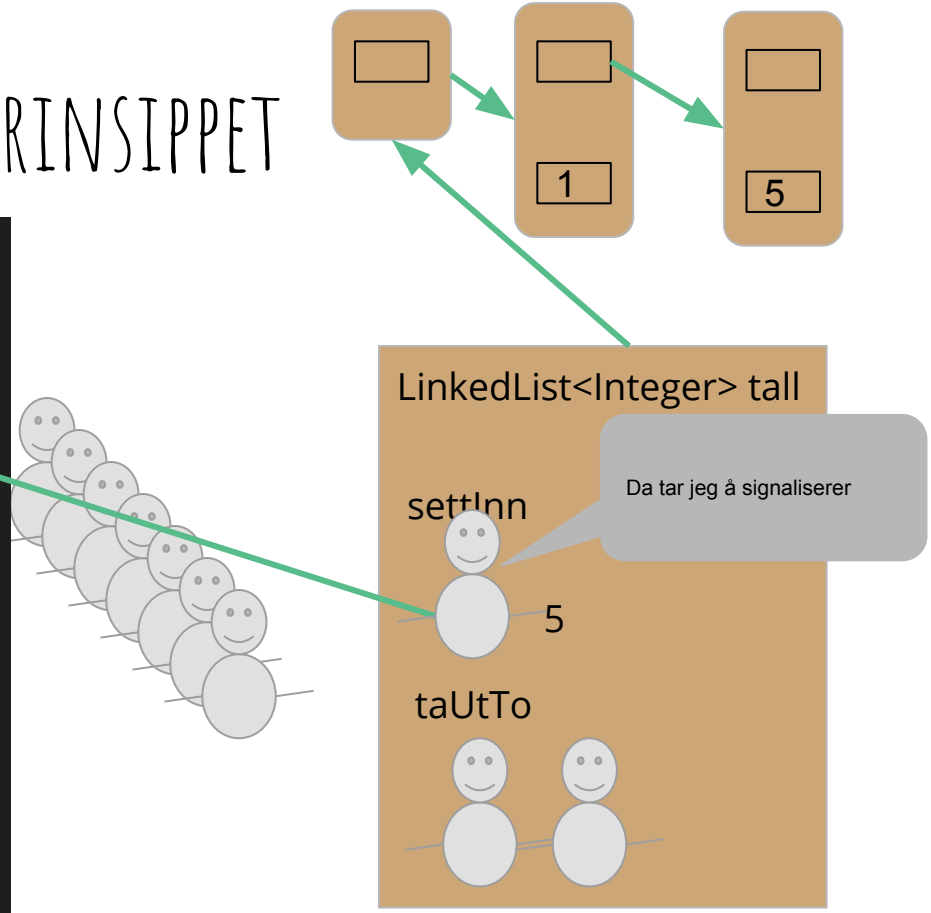
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

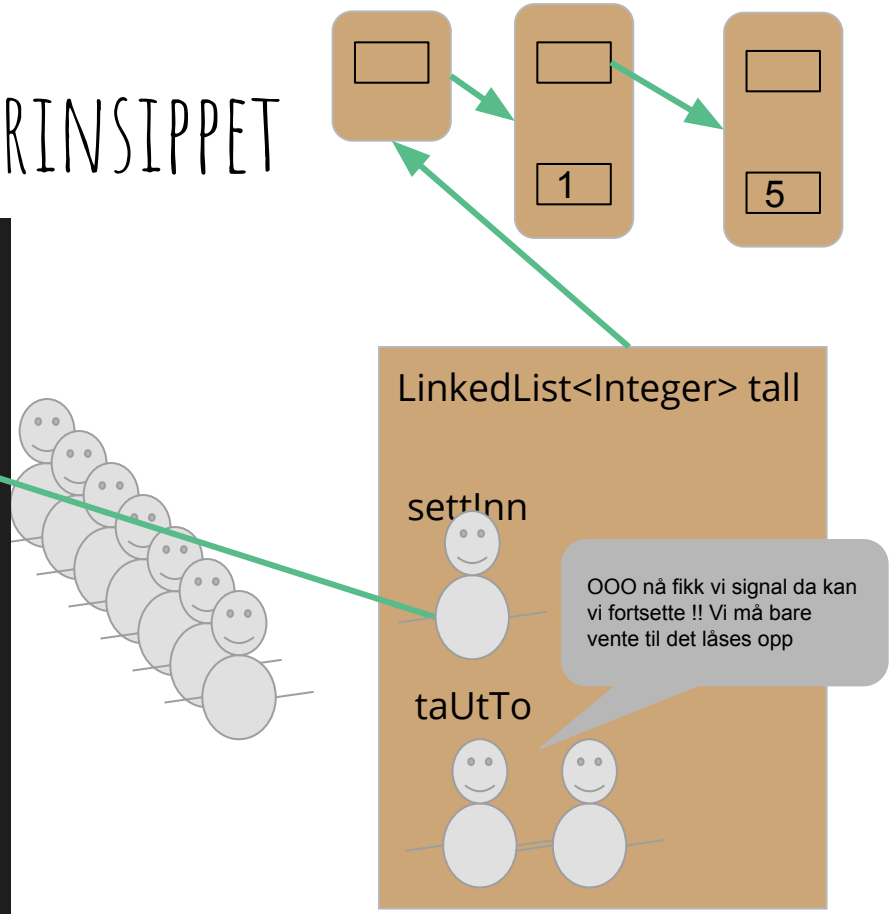
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

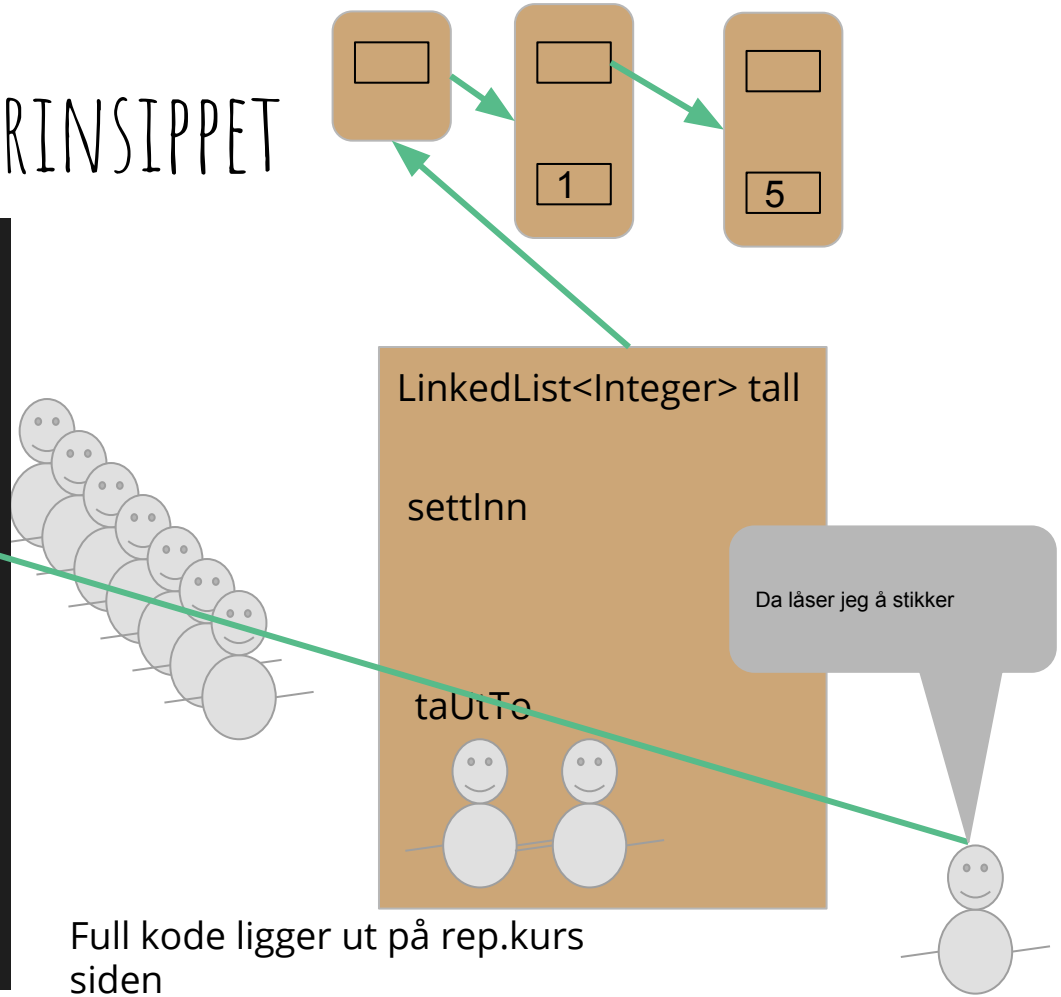
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

Bruk monitor design prinsippet

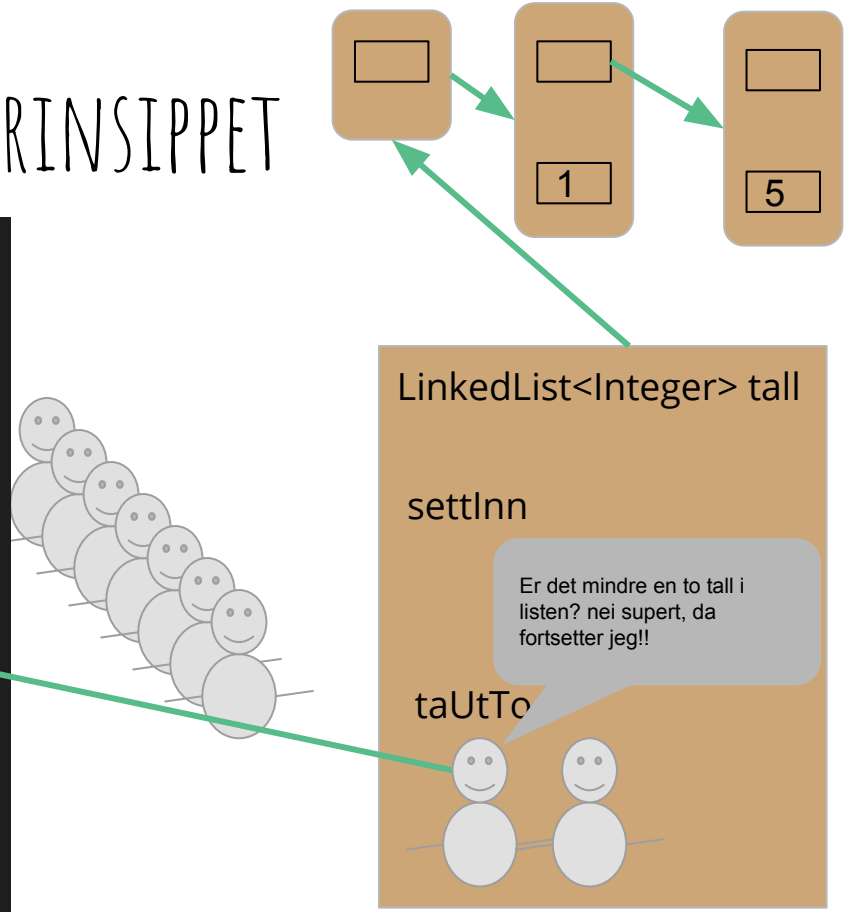
```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Full kode ligger ut på rep.kurs siden

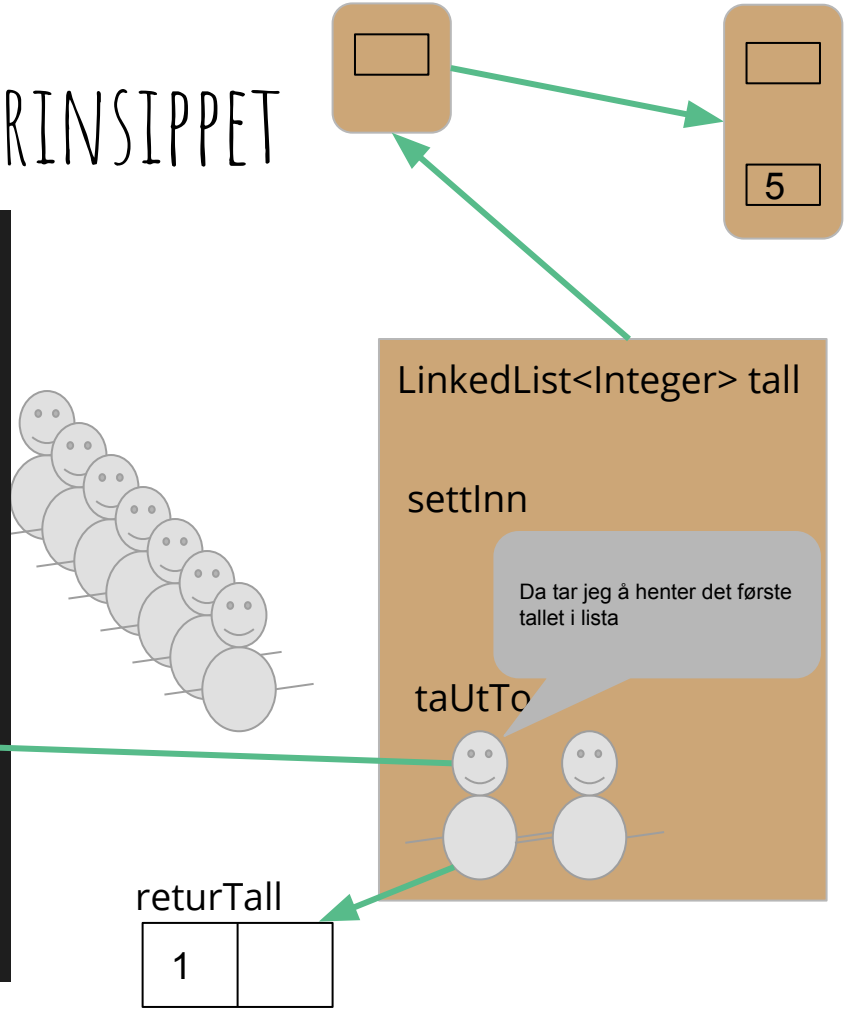
Bruk monitor design prinsippet

```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



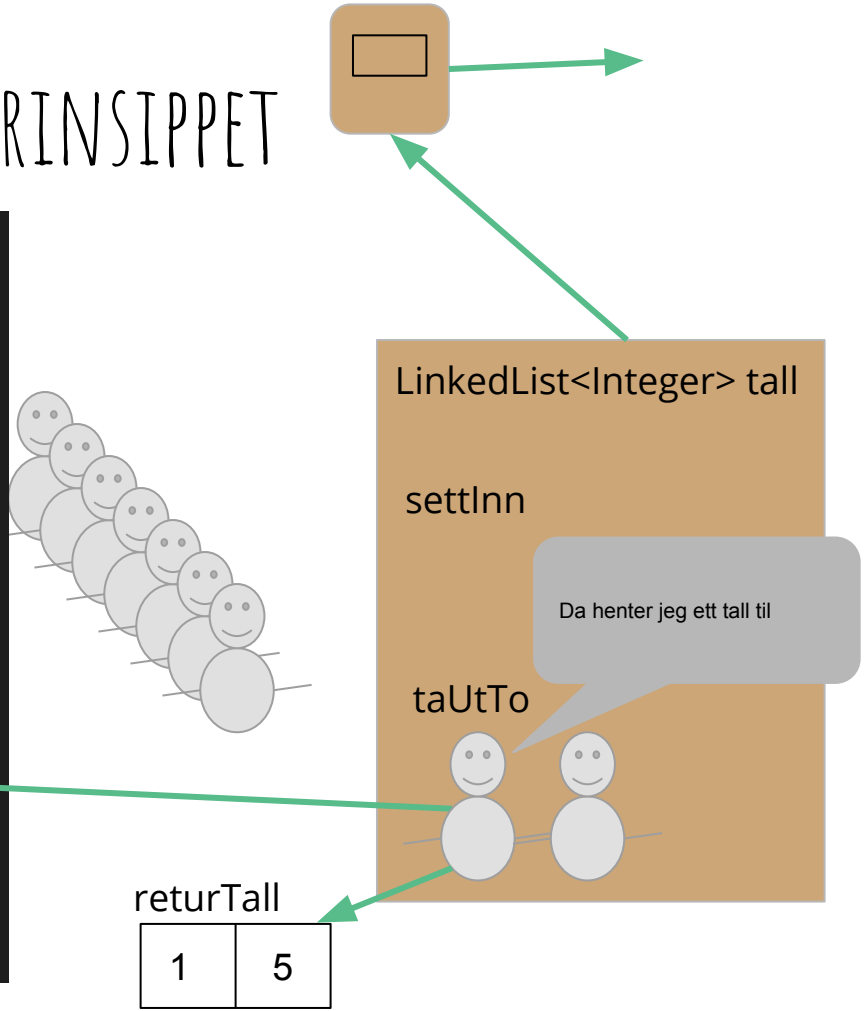
Bruk monitor design prinsippet

```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



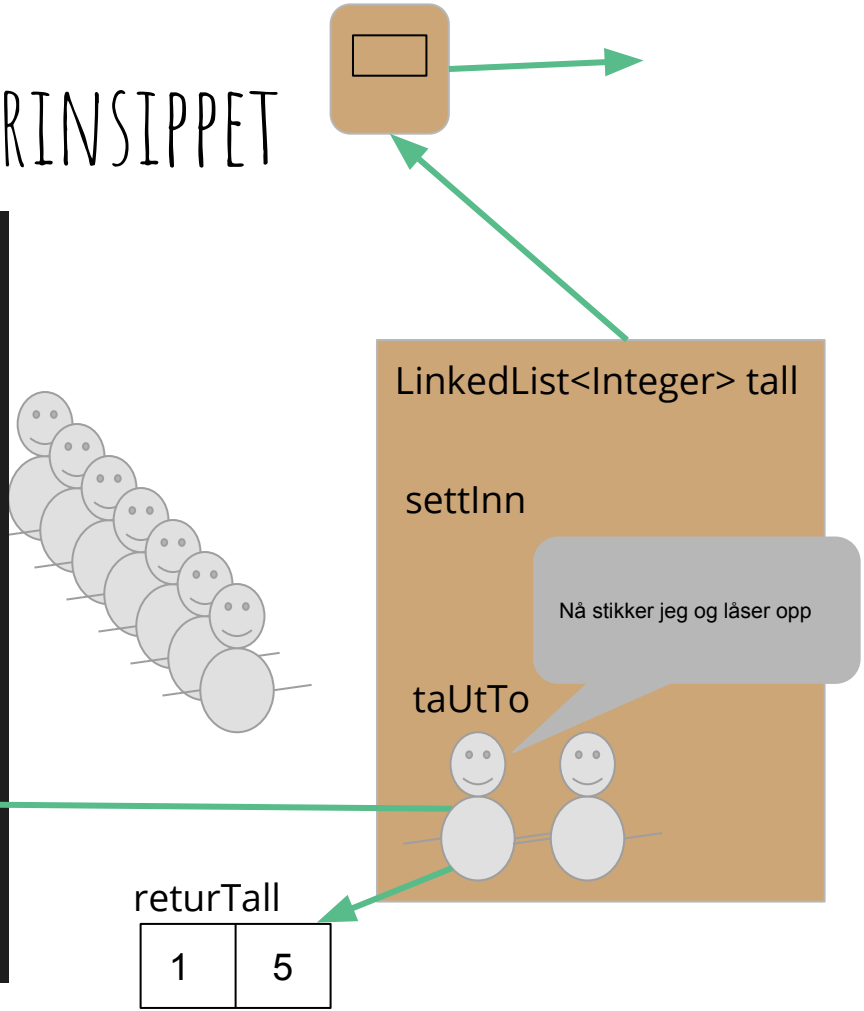
Bruk monitor design prinsippet

```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



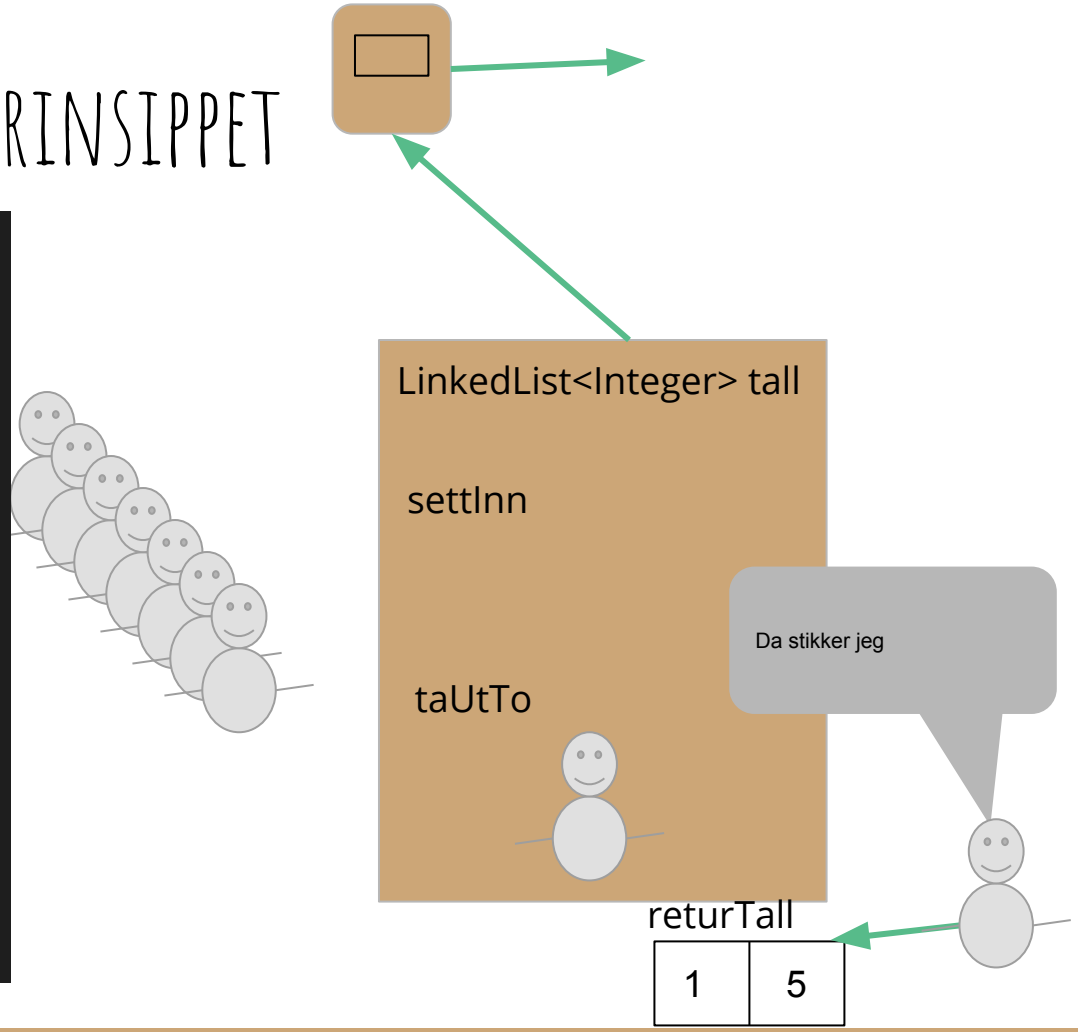
Bruk monitor design prinsippet

```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



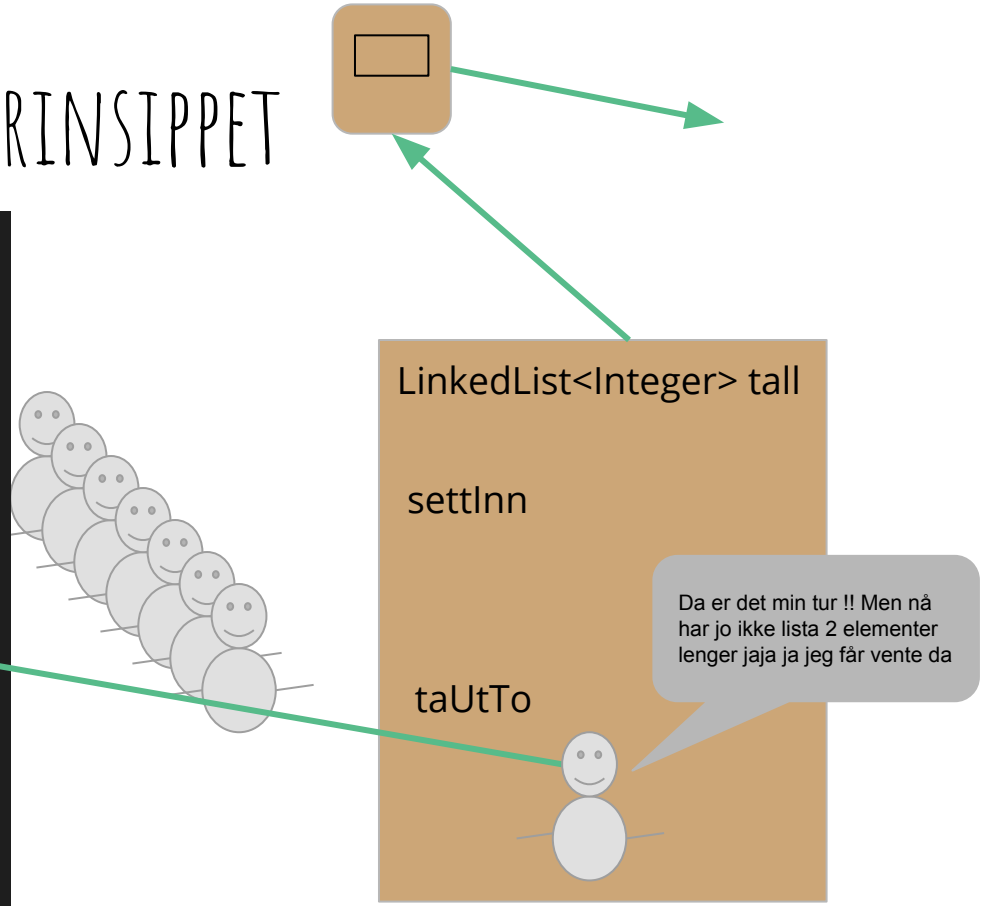
Bruk monitor design prinsippet

```java
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



Bruk monitor design prinsippet

```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));
        return returTall;
    }catch(InterruptedException e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



CountDownLatch

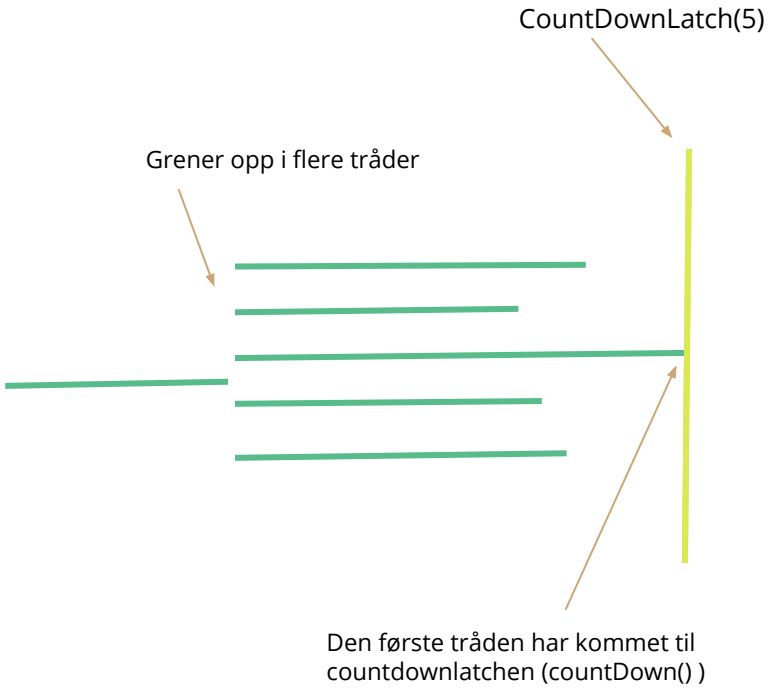
- Konstruktøren tar inn antall tråder den skal vente på
- Nyttige metoder:
  - await(), Tråden blir stoppet her og står å venter til countern har telt seg ned til 0. Denne metoden kan kaste unntak! (Derfor må man bruke try-catch)
  - countDown(): Teller ned counteren en gang.
  - getCount(): returnerer countern (sagt med andre ord hvor langt den har kommet i nedtellingen)

DOKUMMENTASJON: https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/CountDownLatch.html

```java
class EksempelCountDwon {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CountDownLatch cdl = new CountDownLatch(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CountDownTraad(cdl)).start();
        }
        System.out.println("Hovedtraad venter");
        try {
            cdl.await();
        } catch (InterruptedException e) {
            System.out.println("Ble forstyrret");
        }
        System.out.println("Hovedtraad ferdig");
    }

}

class CountDownTraad implements Runnable{

    CountDownLatch cdl;

    public CountDownTraad(CountDownLatch cdl){
        this.cdl = cdl;
    }

    @Override
    public void run(){
        System.out.println("CountDown");
        cdl.countDown();
    }
}
```

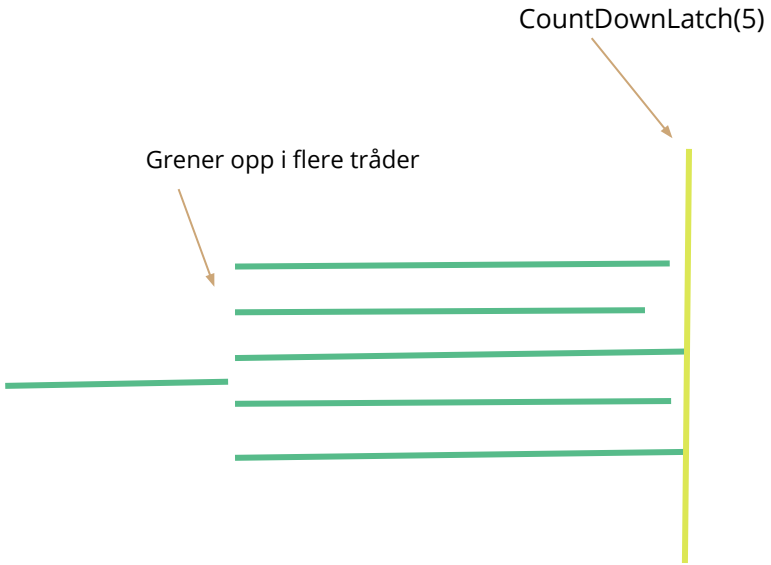

```java
class EksempelCountDwon {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CountDownLatch cdl = new CountDownLatch(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CountDownTraad(cdl)).start();
        }
        System.out.println("Hovedtraad venter");
        try {
            cdl.await();
        } catch (InterruptedException e) {
            System.out.println("Ble forstyrret");
        }
        System.out.println("Hovedtraad ferdig");
    }

}
```

```java
class CountDownTraad implements Runnable{

    CountDownLatch cdl;

    public CountDownTraad(CountDownLatch cdl){
        this.cdl = cdl;
    }

    @Override
    public void run(){
        System.out.println("CountDown");
        cdl.countDown();
    }
}
```



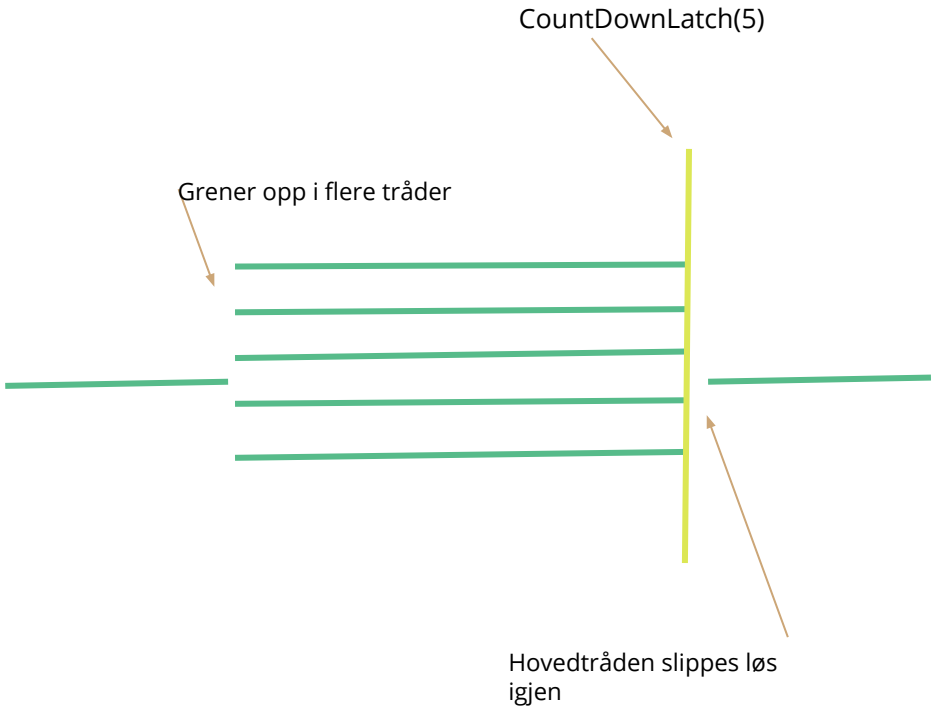

```java
class EksempelCountDwon {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CountDownLatch cdl = new CountDownLatch(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CountDownTraad(cdl)).start();
        }
        System.out.println("Hovedtraad venter");
        try {
            cdl.await();
        } catch (InterruptedException e) {
            System.out.println("Ble forstyrret");
        }
        System.out.println("Hovedtraad ferdig");
    }

}
```

```java
class CountDownTraad implements Runnable{

    CountDownLatch cdl;

    public CountDownTraad(CountDownLatch cdl){
        this.cdl = cdl;
    }

    @Override
    public void run(){
        System.out.println("CountDown");
        cdl.countDown();
    }
}
```




```java
class EksempelCountDwon {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CountDownLatch cdl = new CountDownLatch(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CountDownTraad(cdl)).start();
        }
        System.out.println("Hovedtraad venter");
        try {
            cdl.await();
        } catch (InterruptedException e) {
            System.out.println("Ble forstyrret");
        }
        System.out.println("Hovedtraad ferdig");
    }

}
```

```java
class CountDownTraad implements Runnable{

    CountDownLatch cdl;

    public CountDownTraad(CountDownLatch cdl){
        this.cdl = cdl;
    }

    @Override
    public void run(){
        System.out.println("CountDown");
        cdl.countDown();
    }
}
```




```java
class EksempelCountDwon {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CountDownLatch cdl = new CountDownLatch(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CountDownTraad(cdl)).start();
        }
        System.out.println("Hovedtraad venter");
        try {
            cdl.await();
        } catch (InterruptedException e) {
            System.out.println("Ble forstyrret");
        }
        System.out.println("Hovedtraad ferdig");
    }

}
```

```java
class CountDownTraad implements Runnable{

    CountDownLatch cdl;

    public CountDownTraad(CountDownLatch cdl){
        this.cdl = cdl;
    }

    @Override
    public void run(){
        System.out.println("CountDown");
        cdl.countDown();
    }
}
```

Skriv i chatten til Marlen(chat)

En forventet utskift i terminalen etter at koden er kjørt.

PS: her er det flere riktige svar.

```java
class EksempelCountDwon {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CountDownLatch cdl = new CountDownLatch(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CountDownTraad(cdl)).start();
        }
        System.out.println("Hovedtraad venter");
        try {
            cdl.await();
        } catch (InterruptedException e) {
            System.out.println("Ble forstyrret");
        }
        System.out.println("Hovedtraad ferdig");
    }

}

class CountDownTraad implements Runnable{

    CountDownLatch cdl;

    public CountDownTraad(CountDownLatch cdl){
        this.cdl = cdl;
    }

    @Override
    public void run(){
        System.out.println("CountDown");
        cdl.countDown();
    }
}
```

CyclicBarrier

- Konstruktøren tar inn antall tråder som skal synkroniseres på et tidspunkt
- Metoder:
  - await(): Venter til alle trådene kommer til denne barrieren. Kan kaste unntak(Husk try-catch)

DOKUMENTASJON: https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/CyclicBarrier.html

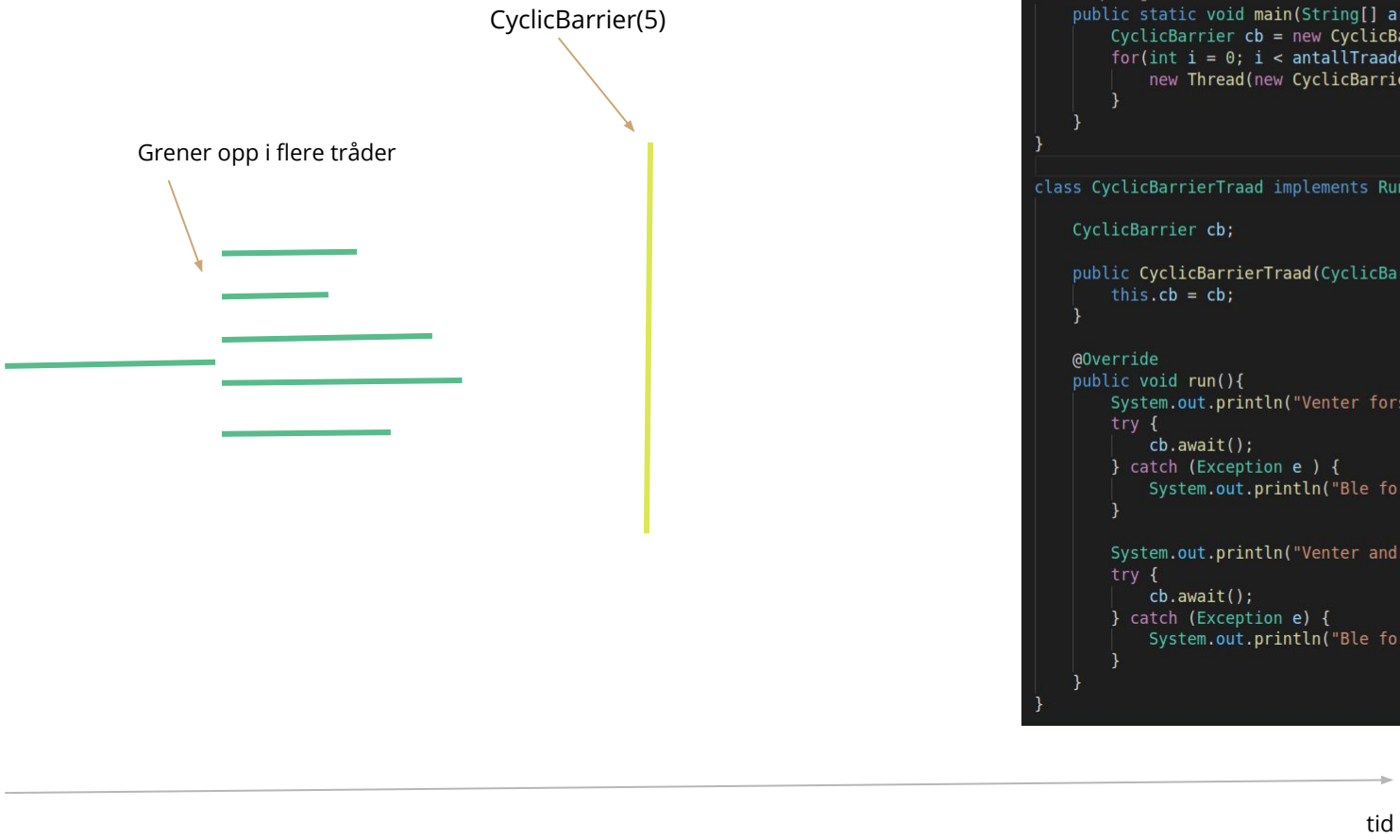
```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}

class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```

Grener opp i flere tråder

```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

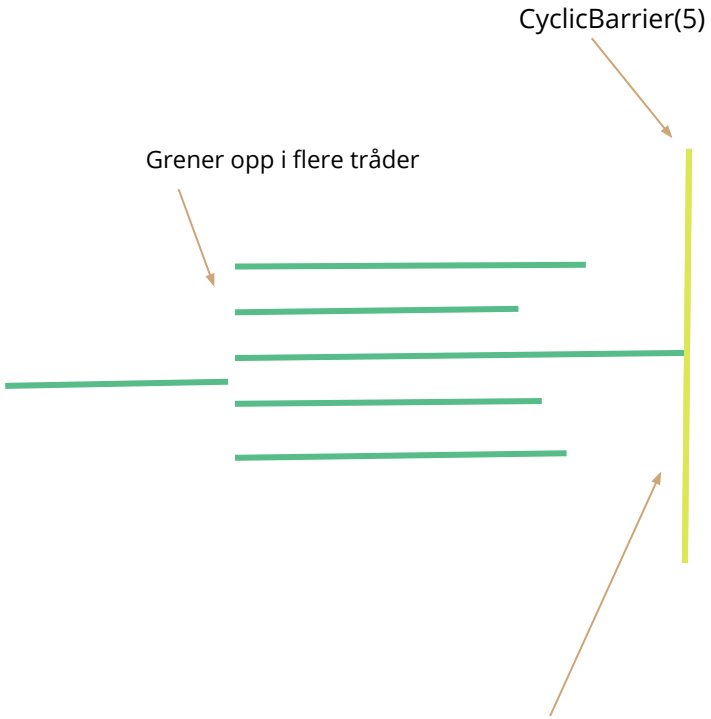
```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```

tid



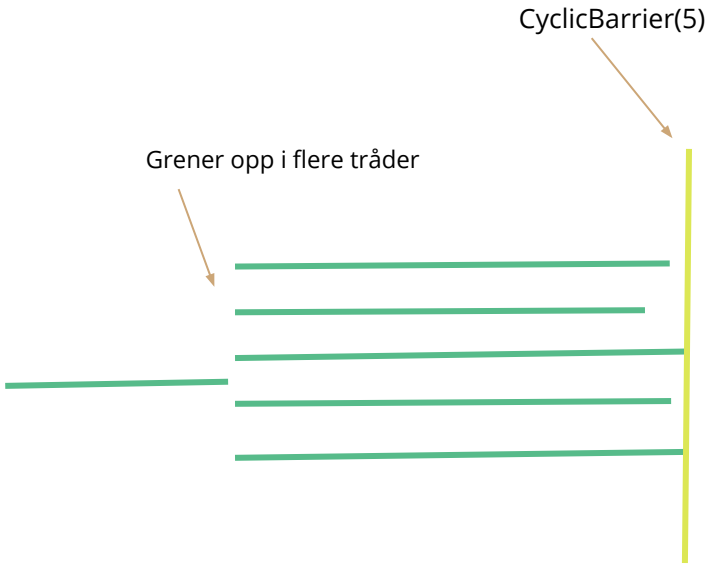

```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```



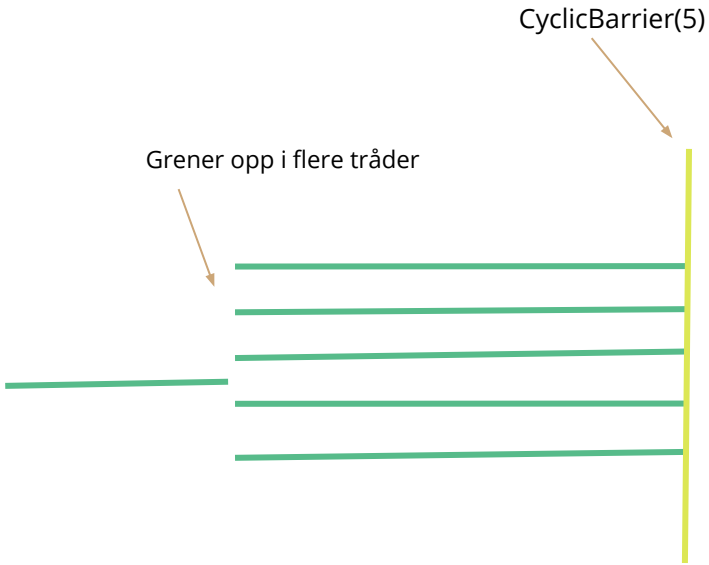

```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```



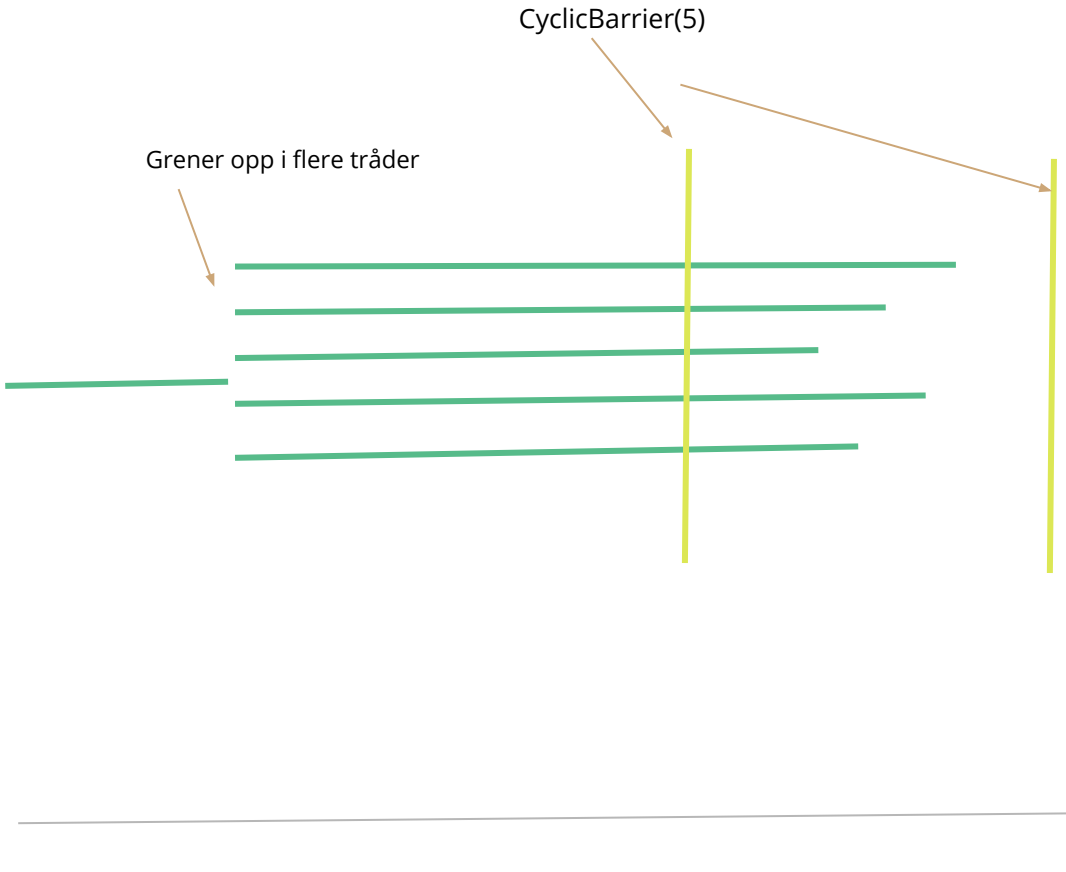

```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```



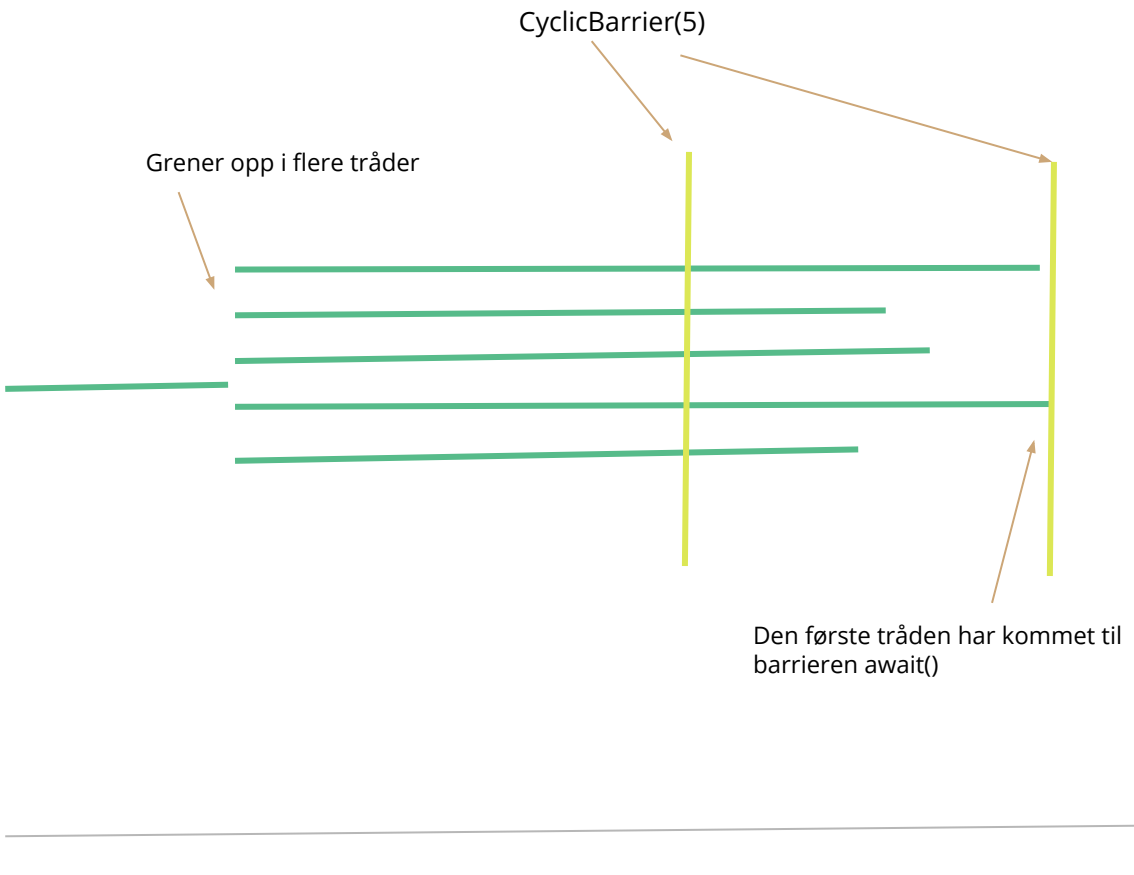

```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```

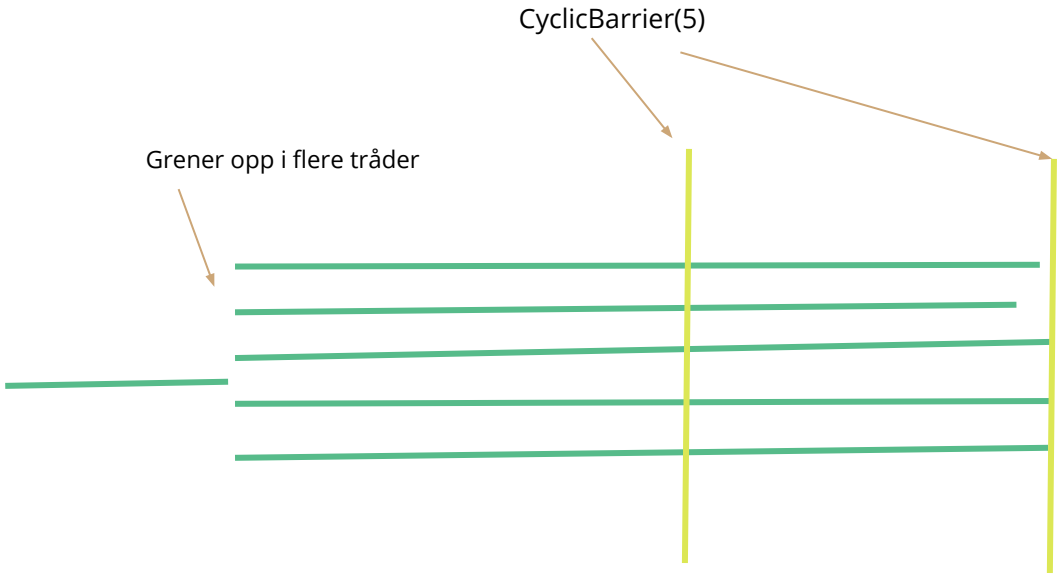



```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```

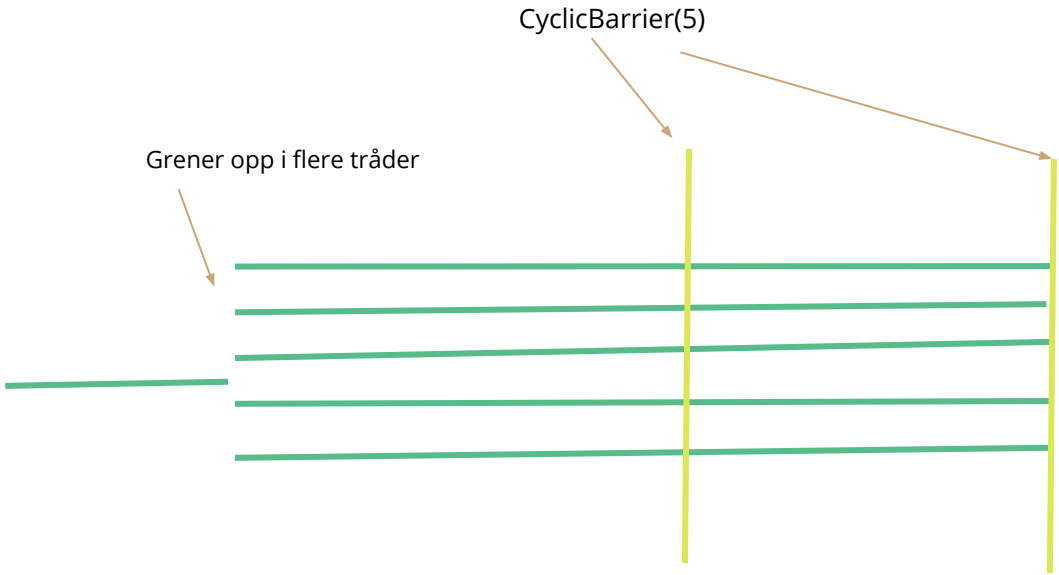



```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}

class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```




```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```




```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```

Skriv i chatten til Marlen(chat)

Forventet utskift i terminalen etter at koden er kjørt.

PS: Her er det bare ett riktig svar

```java
class EksempelCyclic {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CyclicBarrierTraad(cb)).start();
        }
    }
}
```

```java
class CyclicBarrierTraad implements Runnable{

    CyclicBarrier cb;

    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        System.out.println("Venter andre gang");
        try {
            cb.await();
        } catch (Exception e) {
            System.out.println("Ble forstyrret");
        }
    }
}
```

Join

Join er en metode du kan kalle join på en tråd.

Da vil tråden du er i vente til den tråden som kalte på join metoden er terminert (ferdig)

```java
class EksempelJoin {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {

        ArrayList<Thread> traader = new ArrayList<>(antallTraader);
        for(int i = 0; i < antallTraader; i ++){
            Thread traad = new Thread(new JoinTraad());
            traader.add(traad);
            traad.start();
        }

        for(Thread traad : traader){
            try {
                traad.join();
            } catch (InterruptedException e) {
                System.out.println("Interupt feil");
            }
        }
        System.out.println("Hovedtraad ferdig");
    }

}
class JoinTraad implements Runnable{

    @Override
    public void run(){
        System.out.println("Kjorer run metoden");
    }
}
```

Join

Send marlen(chat) i chatten hva som vil printes ut her

PS: her er det kun et riktig svar

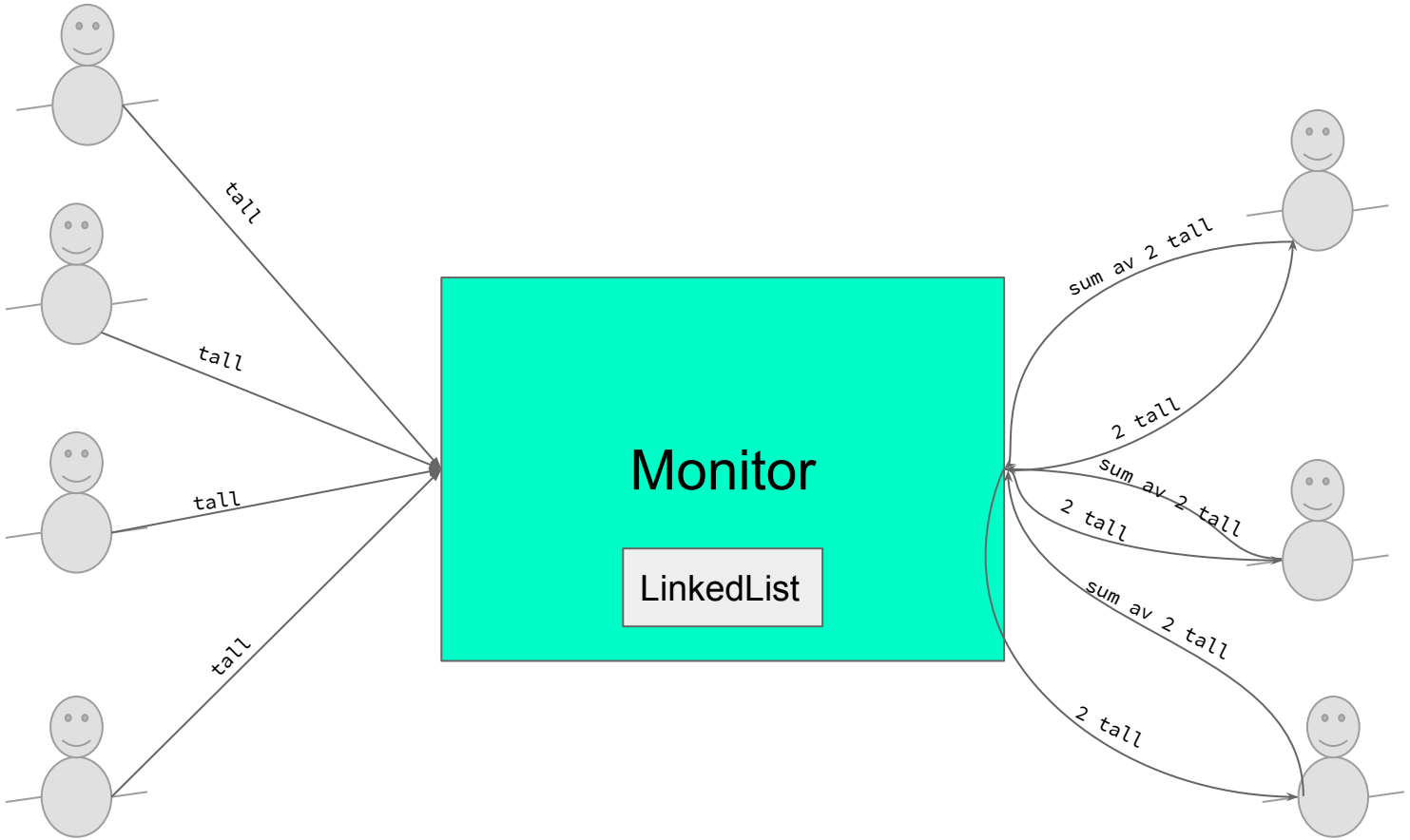
```java
class EksempelJoin {

    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {

        ArrayList<Thread> traader = new ArrayList<>(antallTraader);
        for(int i = 0; i < antallTraader; i ++){
            Thread traad = new Thread(new JoinTraad());
            traader.add(traad);
            traad.start();
        }

        for(Thread traad : traader){
            try {
                traad.join();
            } catch (InterruptedException e) {
                System.out.println("Interupt feil");
            }
        }
        System.out.println("Hovedtraad ferdig");
    }

}

class JoinTraad implements Runnable{

    @Override
    public void run(){
        System.out.println("Kjorer run metoden");
    }
}
```

Live-programmering (del 4)

Fortsett fra del 4.

Du skal endre programmet ditt slik at tråden som legger sammen to ikke printer dem ut men legger resultatet tilbake i linkedlisten.

Lag en metode for å hente ut det siste tallet. Slik at hovedprogrammet kan printe det ut når vi er ferdig :))


tall
tall
tall
tall
Monitor
LinkedList
sum av 2 tall
2 tall
sum av 2 tall
2 tall
sum av 2 tall
2 tall

Noen spørsmål ??