

# REP. KURS TRÅDER

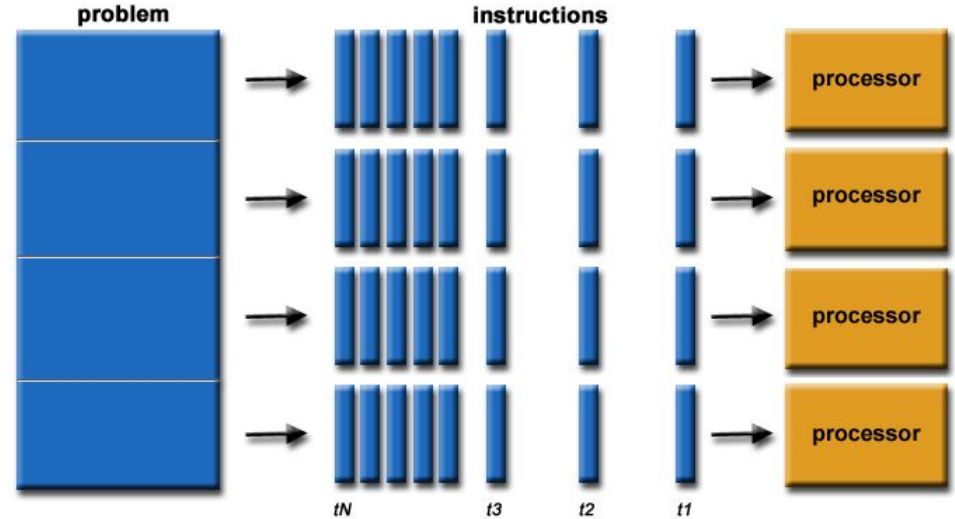
**Foiler av: Marlen Jarholt ([marleja@ifi.uio.no](mailto:marleja@ifi.uio.no))**

# 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

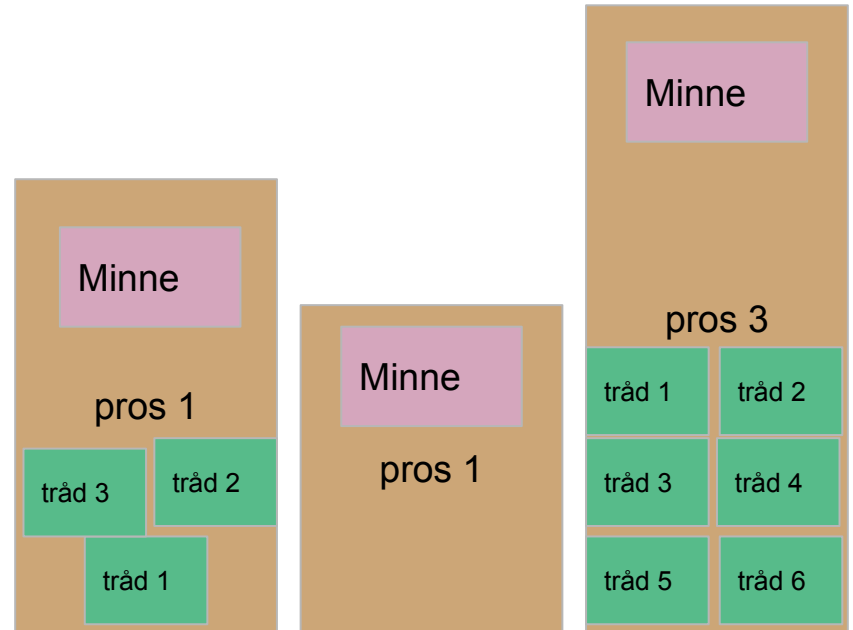
- CPU/ Kjerner
- Flere programmer for kjørt
- Maskinen har kapasiteten
- Dele opp problemet
- Raskere, men
  - Det tar tid å starte opp en tråd
  - Synkronisering



Bildekilde: [https://computing.llnl.gov/tutorials/parallel\\_comp/](https://computing.llnl.gov/tutorials/parallel_comp/)

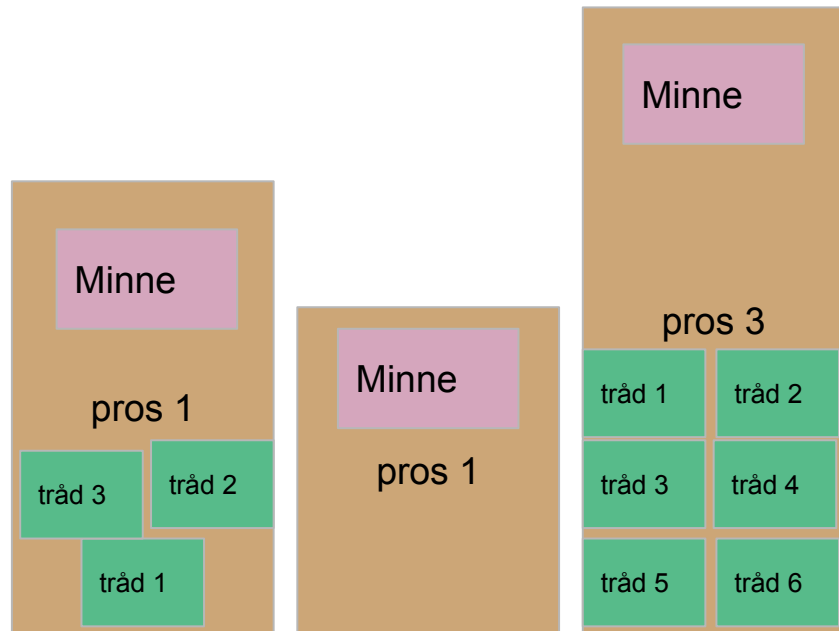
# HVA ER EN TRÅD?

- 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
- Skrivning -> Lås
- Lesing -> Trengs ikke å låse
- Lesing og skrivning -> 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\(\)](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 importere
  - `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 mellom `lock()` og `unlock()` omgangen
- 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         }
55     }
```



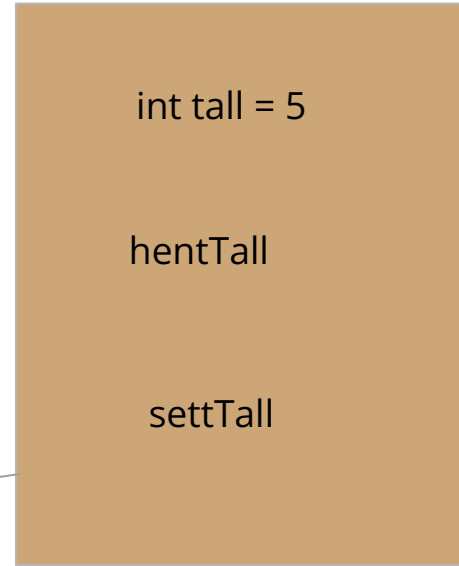
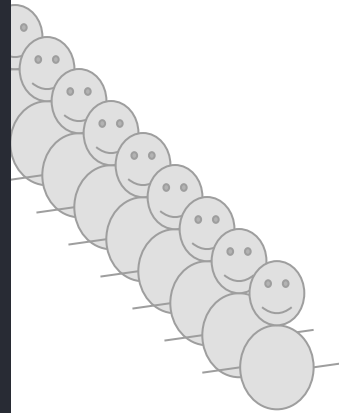
# MONITOR

- Et objekt som innkapsler den delte dataen
- Fungere som en type beskytter av felles data
- Monitor objektet har metoder som gjør at de andre objekter (ofte tråd objekter) 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

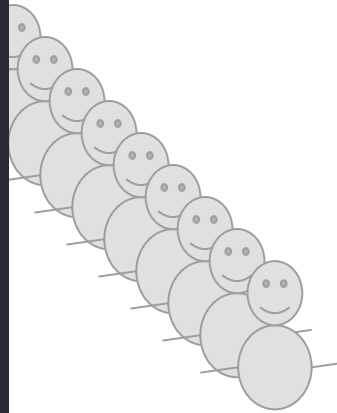
```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3 -
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7     -
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```



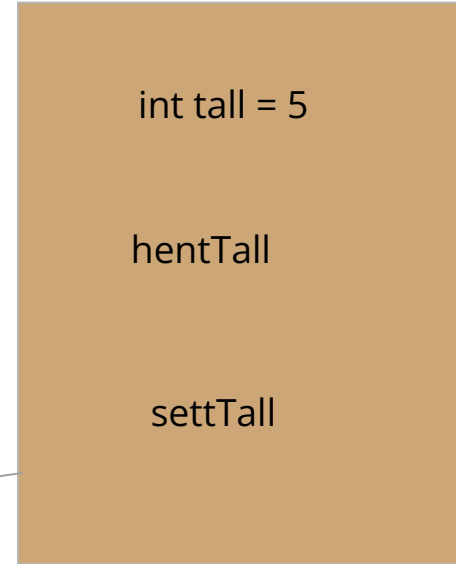
← Monitor klasse

# BRUK MONITOR DESIGN PRINSIPPET

```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3 -
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7     -
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```

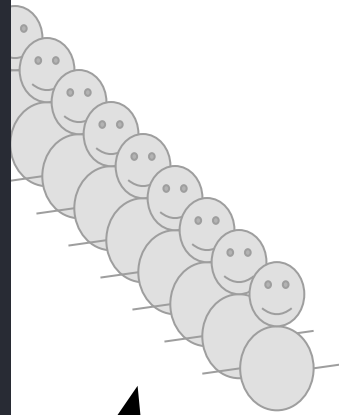


Forsøk på å illustrere monitor



# BRUK MONITOR DESIGN PRINSIPPET

```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3 -
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7     -
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```



int tall = 5

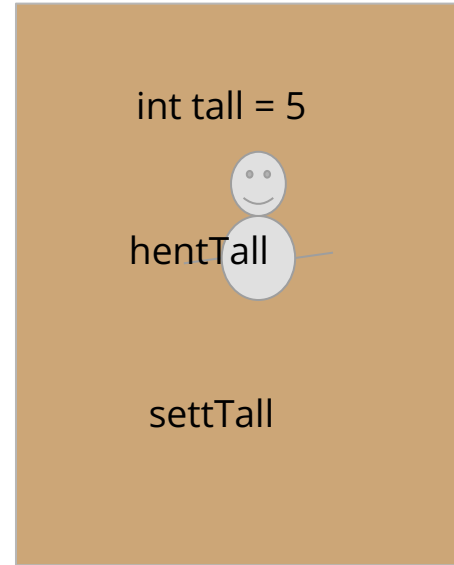
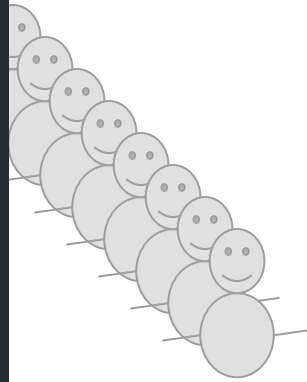
hentTall

settTall

Alle trådene som aksesserer data i monitoren én og én

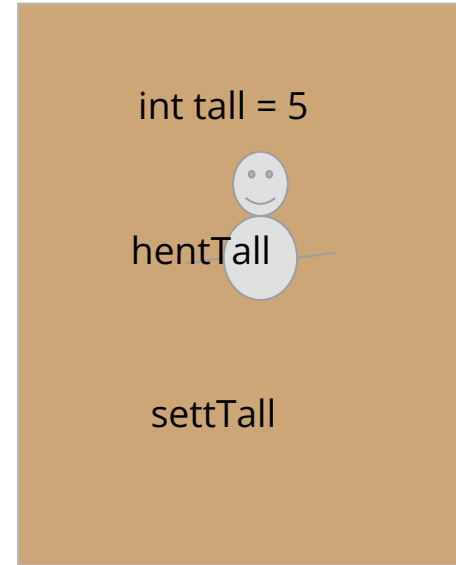
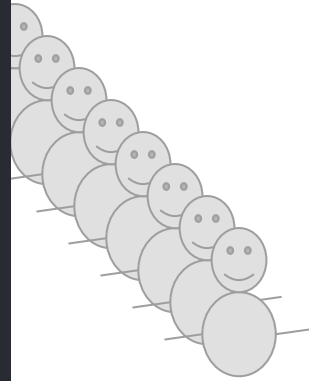
# BRUK MONITOR DESIGN PRINSIPPET

```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18
19    public int hentTall(){-
20        laas.lock();-
21        try {-
22            return tall;-
23        } catch(Exception e) {-
24            System.out.println("Feil i monitor tallBeholder");-
25        } finally {-
26            laas.unlock();-
27        }-
28    }-
29 }
```



# BRUK MONITOR DESIGN PRINSIPPET

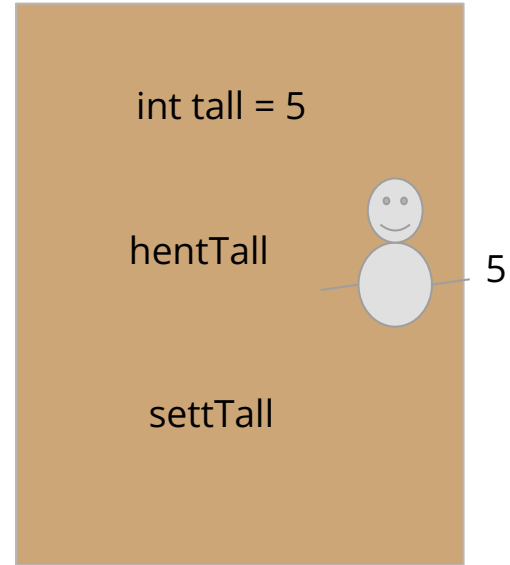
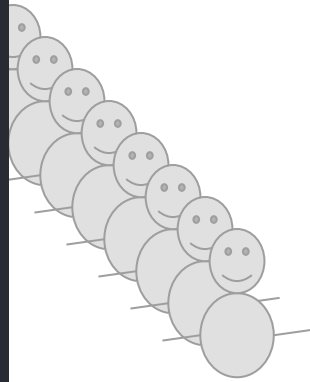
```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```



Låst fra noen kaller på lock() til de tar kaller på unlock()

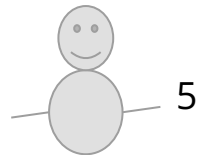
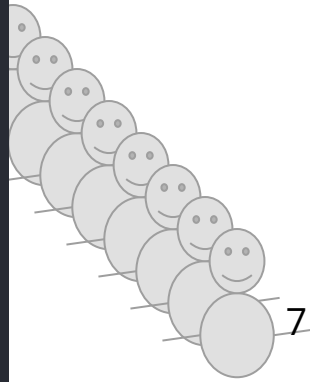
# BRUK MONITOR DESIGN PRINSIPPET

```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```



# BRUK MONITOR DESIGN PRINSIPPET

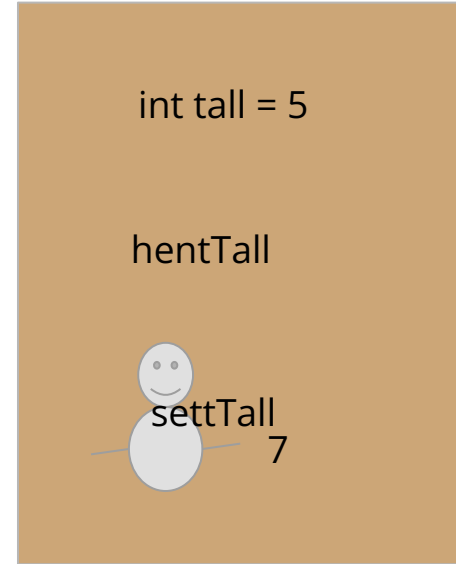
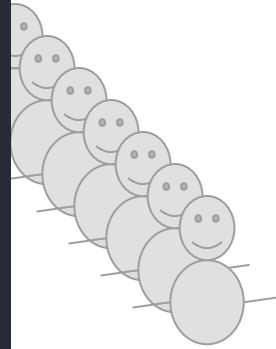
```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3 -
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7     -
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```



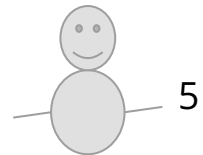


# BRUK MONITOR DESIGN PRINSIPPET

```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```

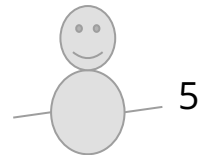
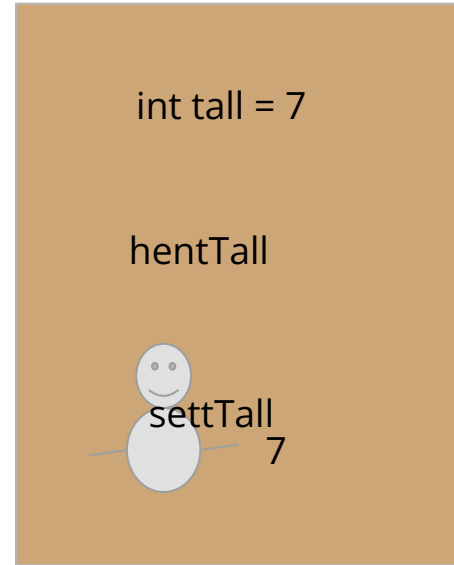
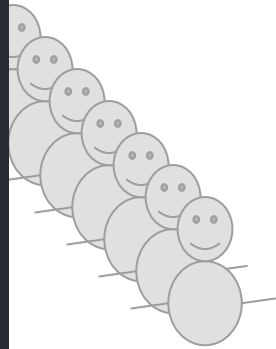


Låst fra noen kaller på lock() til de tar kaller på unlock()



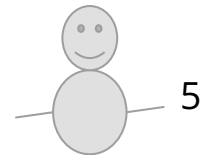
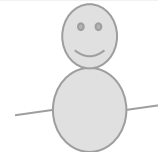
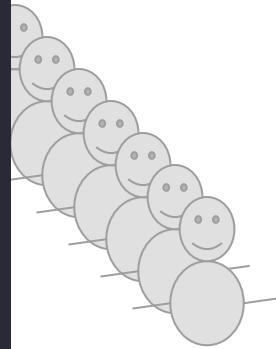
# BRUK MONITOR DESIGN PRINSIPPET

```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3 -
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7     -
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```



# BRUK MONITOR DESIGN PRINSIPPET

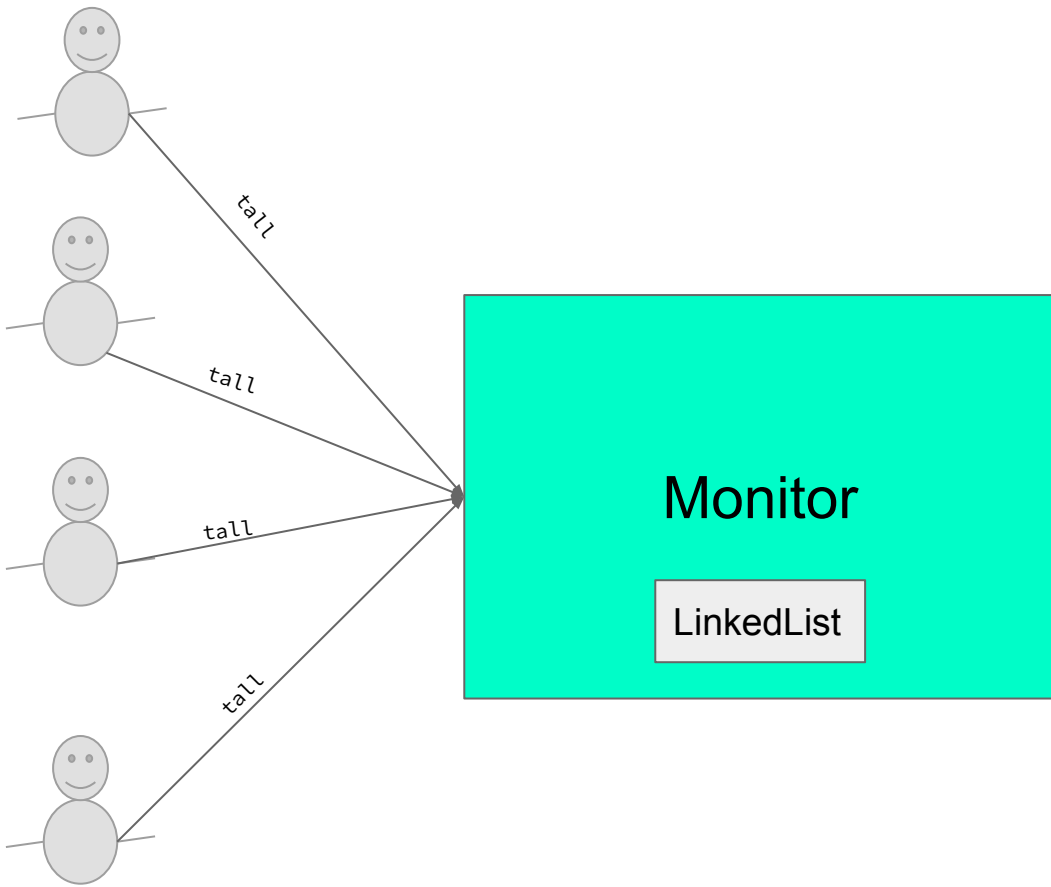
```
1 import java.util.concurrent.locks.Lock;-
2 import java.util.concurrent.locks.ReentrantLock;-
3 -
4 class tallBeholder{-
5     private Lock laas = new ReentrantLock();-
6     private int tall;-
7     -
8     public void settTall(int tall){-
9         laas.lock();-
10        try {-
11            this.tall = tall;-
12        } catch(Exception e) {-
13            System.out.println("Feil i monitor tallBeholder");-
14        } finally {-
15            laas.unlock();-
16        }-
17    }-
18    public int hentTall(){-
19        laas.lock();-
20        try {-
21            return tall;-
22        } catch(Exception e) {-
23            System.out.println("Feil i monitor tallBeholder");-
24        } finally {-
25            laas.unlock();-
26        }-
27    }-
28 }
```



# 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



# CONDITION

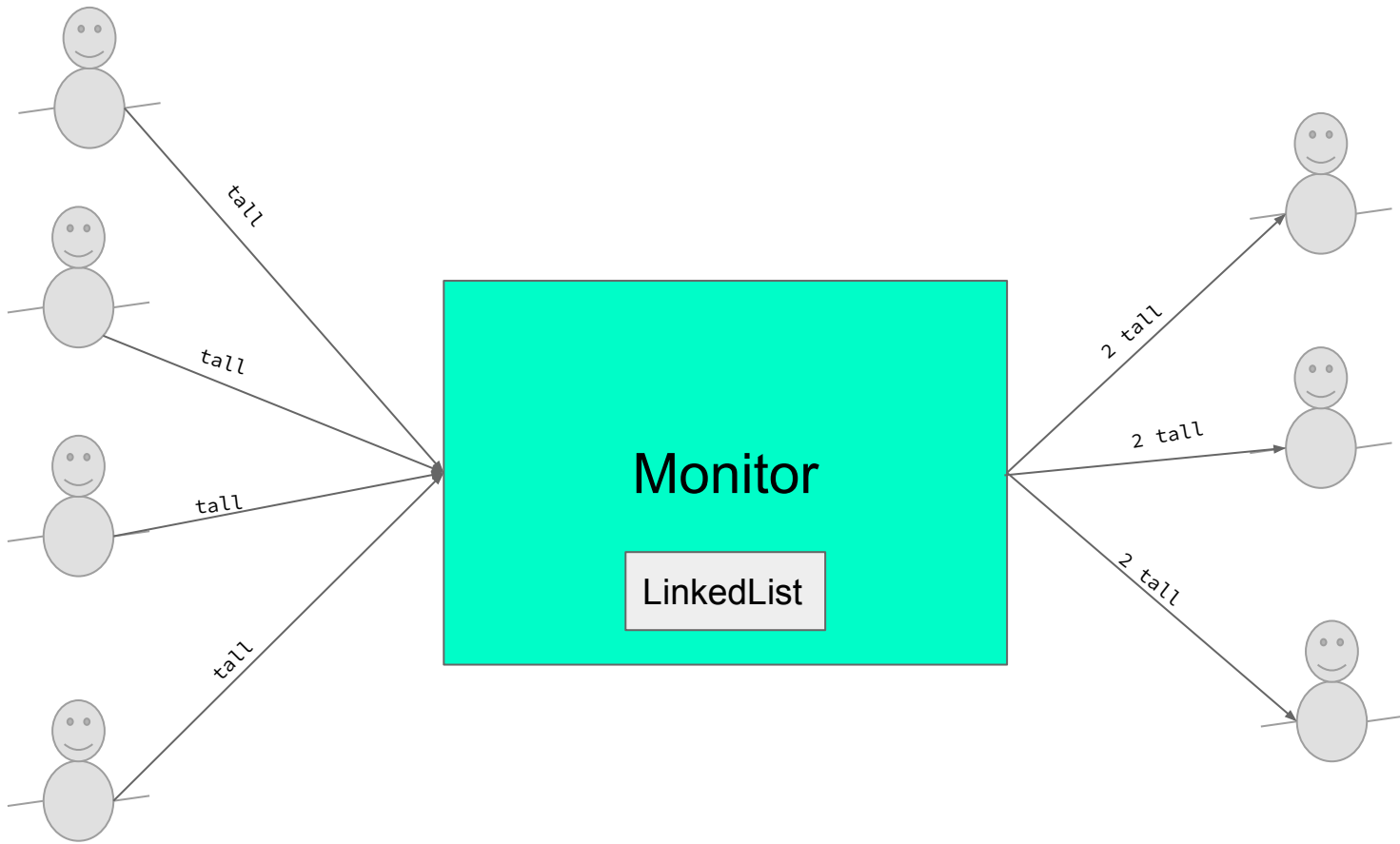
- En form for passiv venting
  - Vil bruke mindre ressurser enn aktiv venting
- Brukes når trådene må vente på en tilstand til dataen for å kunne bruke den.
- `await()`: Får den tråden man er inne i nå til å vente til Condition gir signaliserer at det er greit å fortsette. (Eller at den blir interrupted)
- `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.

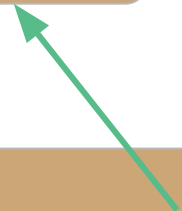
Lag en ny tråd-klasse, denne skal ta ut to tall fra monitoren og legge dem sammen før de printes ut.

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.



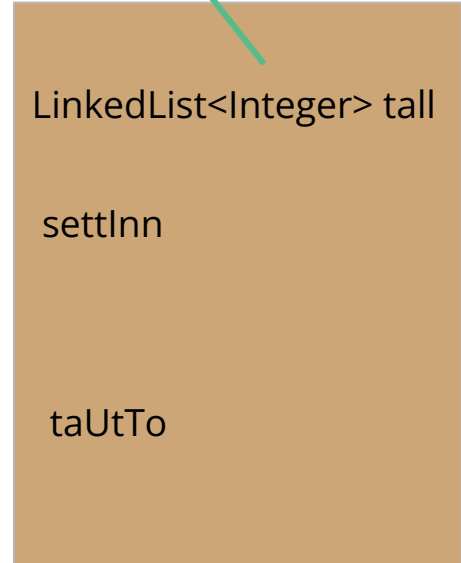
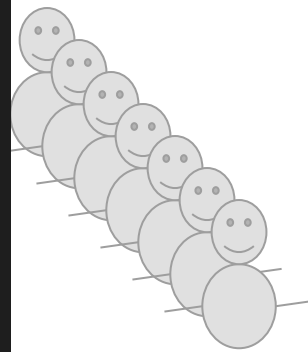


# BRUK MONITOR DESIGN PRINSIPPET

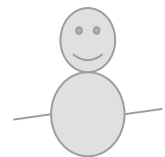


```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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();
    }
}
```



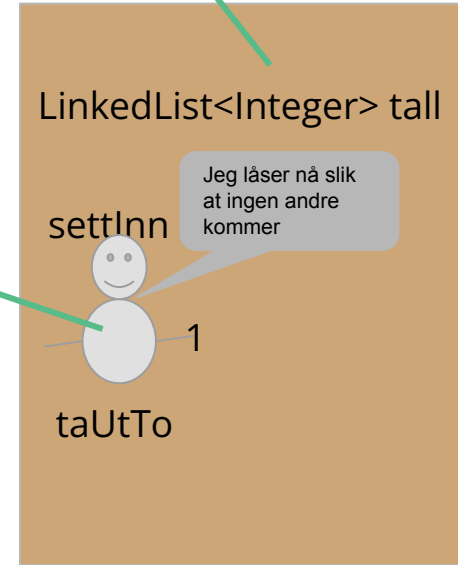
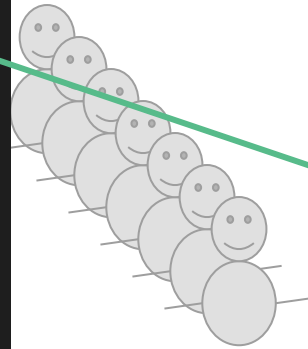
Full kode ligger ut på rep.kurs siden



# BRUK MONITOR DESIGN PRINSIPPET

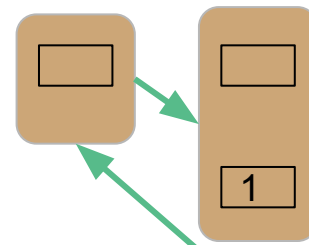
```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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(InterruptedExceotion e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



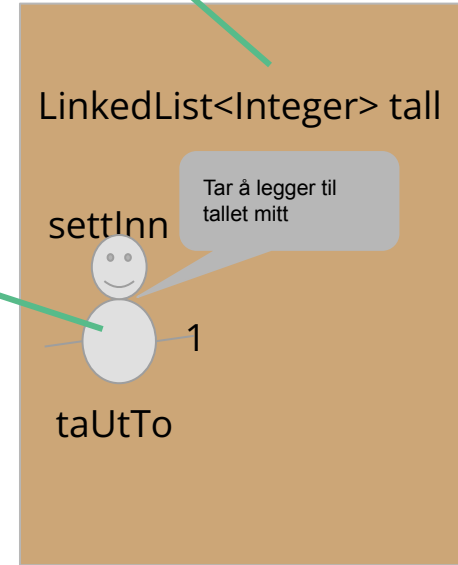
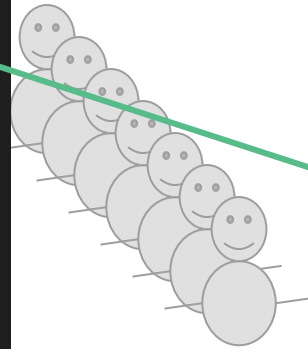
Full kode ligger ut på rep.kurs siden

# BRUK MONITOR DESIGN PRINSIPPET



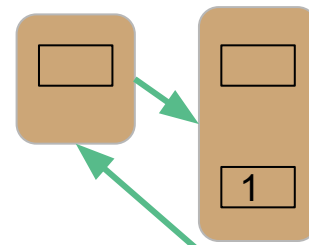
```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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(InterruptedExceotion e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



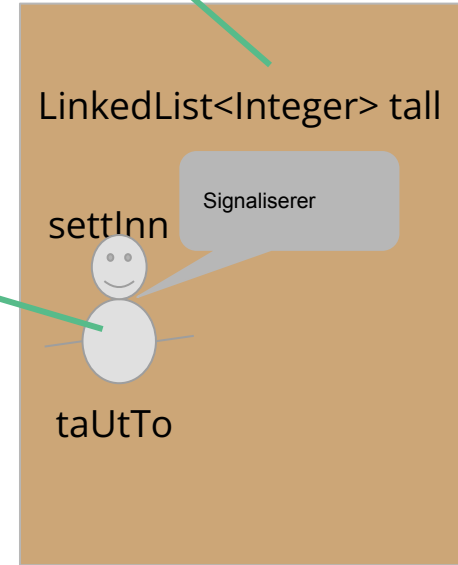
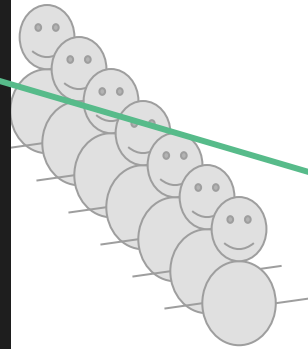
Full kode ligger ut på rep.kurs siden

# BRUK MONITOR DESIGN PRINSIPPET



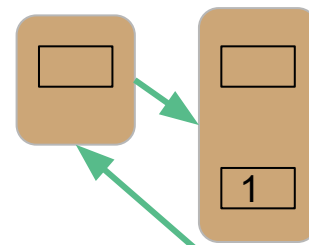
```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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(InterruptedExceotion e){
        return null;
    }finally{
        laas.unlock();
    }
}
```



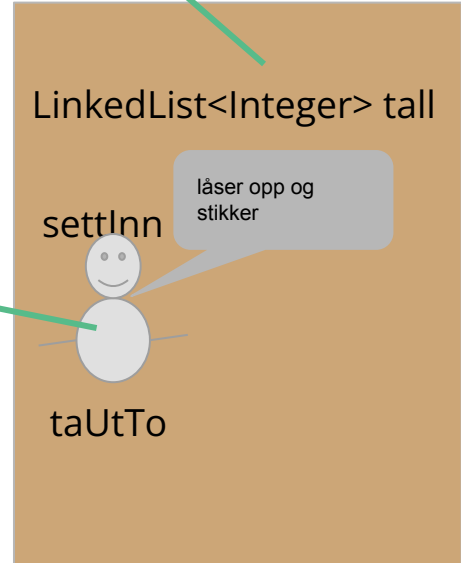
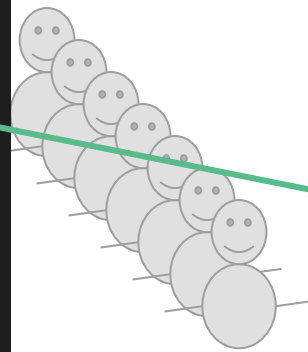
Full kode ligger ut på rep.kurs siden

# BRUK MONITOR DESIGN PRINSIPPET



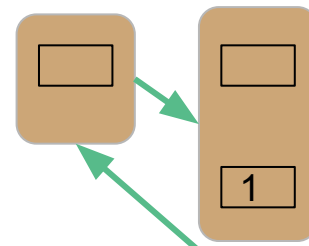
```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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();
    }
}
```



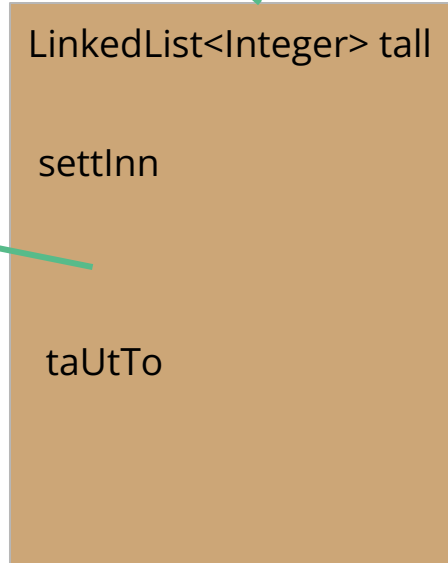
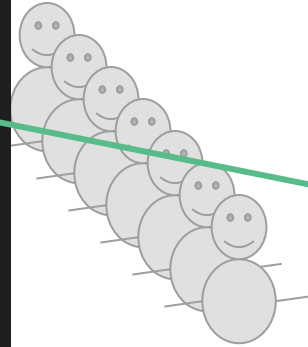
Full kode ligger ut på rep.kurs siden

# BRUK MONITOR DESIGN PRINSIPPET

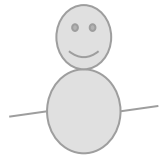


```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

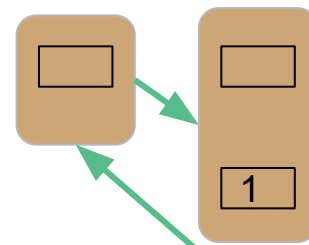
public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));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



Full kode ligger ut på rep.kurs siden

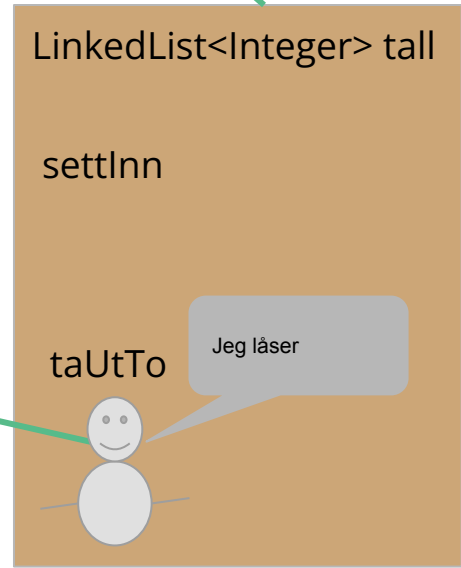
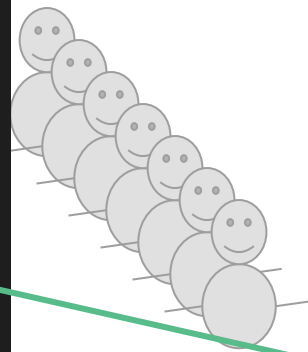


# BRUK MONITOR DESIGN PRINSIPPET



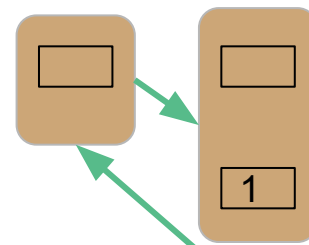
```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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();
    }
}
```



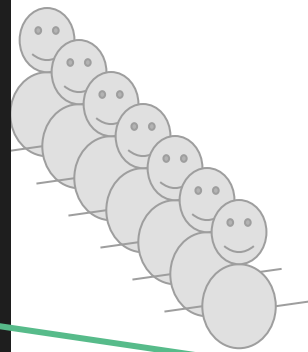
Full kode ligger ut på rep.kurs siden

# BRUK MONITOR DESIGN PRINSIPPET



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

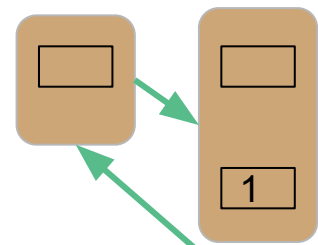
public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));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



Full kode ligger ut på rep.kurs siden

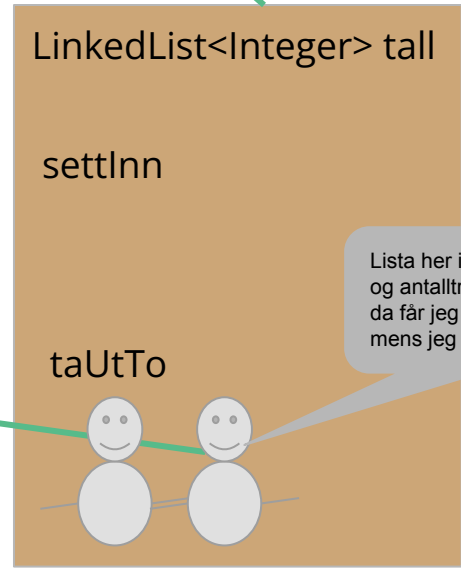
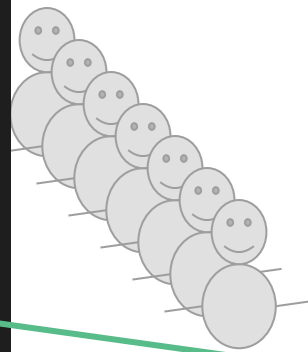


# BRUK MONITOR DESIGN PRINSIPPET



```
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();
    }
}
```



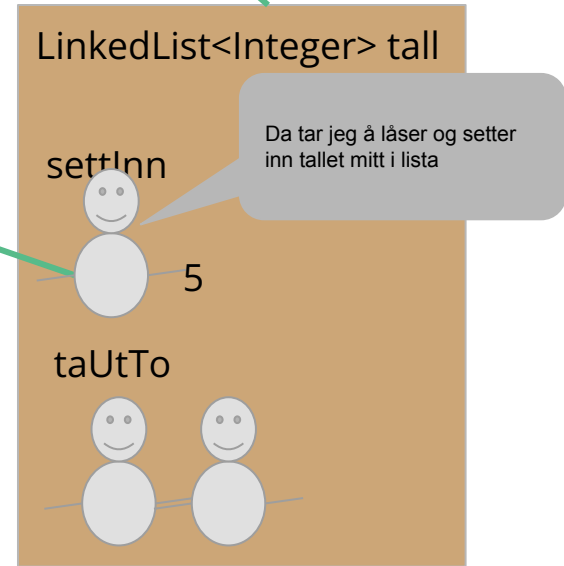
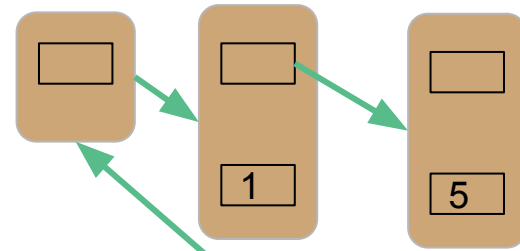
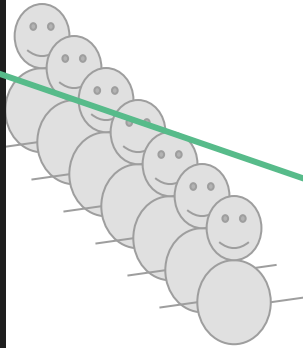
Lista her ikke to elementer og antalltraader er ikke 0 jaja da får jeg vente og låse opp mens jeg venter

Full kode ligger ut på rep.kurs siden

# BRUK MONITOR DESIGN PRINSIPPET

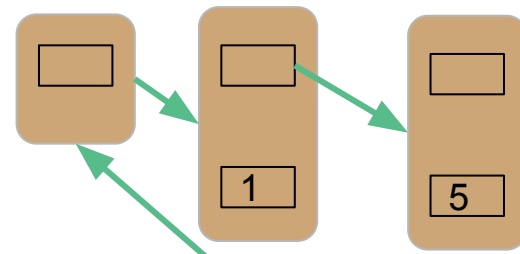
```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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();
    }
}
```



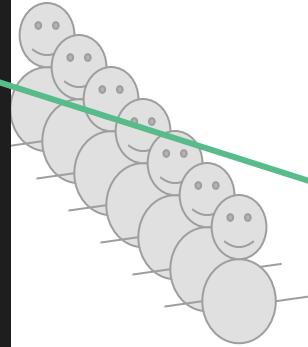
Full kode ligger ut på rep.kurs siden

# BRUK MONITOR DESIGN PRINSIPPET



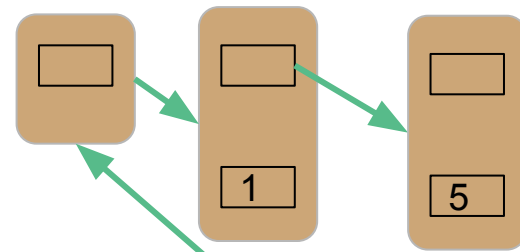
```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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();
    }
}
```



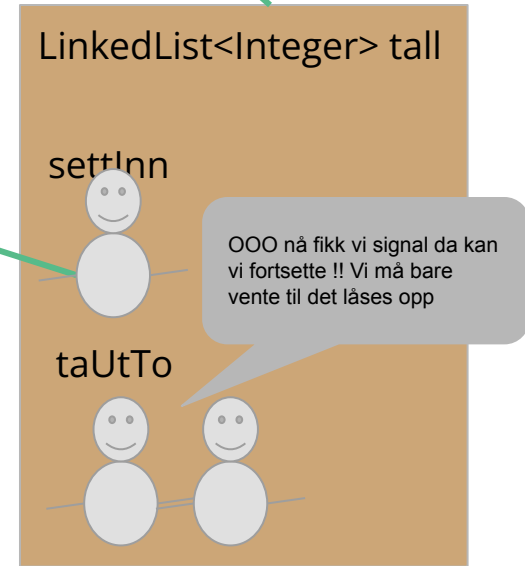
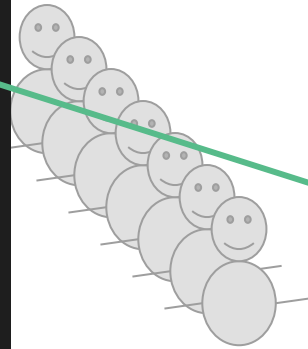
Full kode ligger ut på rep.kurs siden

# 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();
    }
}
```

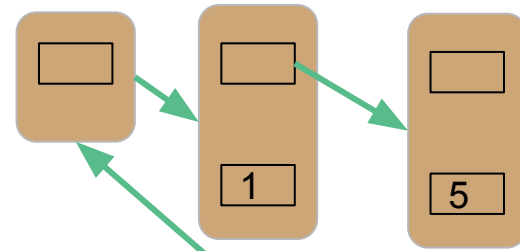
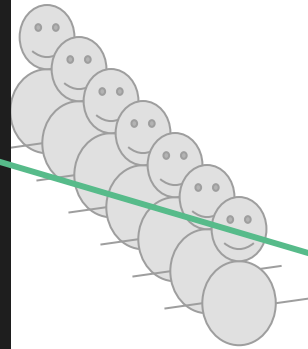


Full kode ligger ut på rep.kurs siden

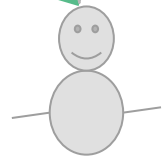
# BRUK MONITOR DESIGN PRINSIPPET

```
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();
    }
}
```

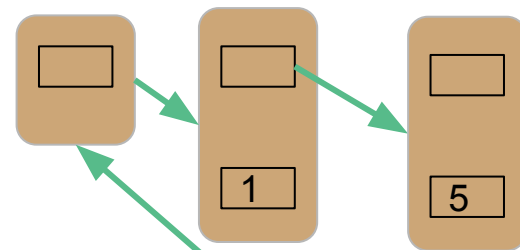


Da låser jeg å stikker



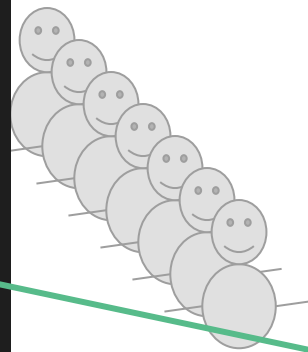
Full kode ligger ut på rep.kurs siden

# BRUK MONITOR DESIGN PRINSIPPET

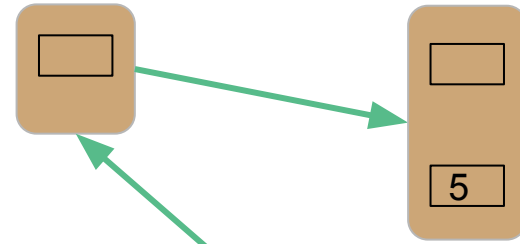


```
public void settInn(int nyttTall){
    laas.lock();
    try{
        tall.add(nyttTall);
        tomListe.signalAll();
    }finally{
        laas.unlock();
    }
}

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();
    }
}
```

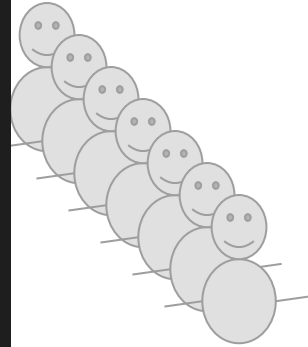


# BRUK MONITOR DESIGN PRINSIPPET



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

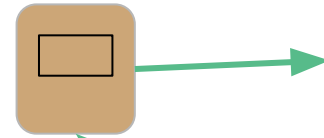
public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));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



returTall

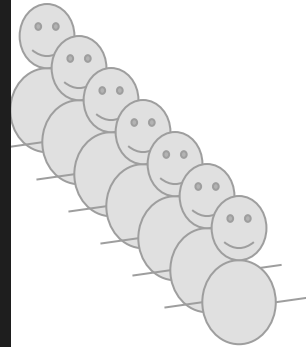


# BRUK MONITOR DESIGN PRINSIPPET



```
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();
    }
}
```

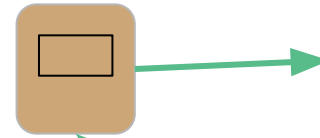


returTall

1	5
---	---

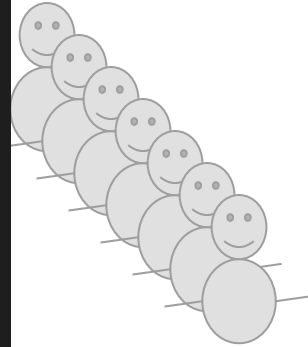


# BRUK MONITOR DESIGN PRINSIPPET



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

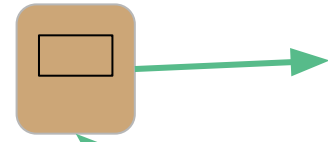
public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));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



returTall

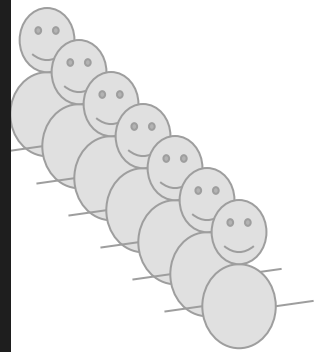
1	5
---	---

# BRUK MONITOR DESIGN PRINSIPPET



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

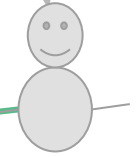
public ArrayList<Integer> taUtTo(){
    laas.lock();
    try{
        while(tall.size() < 2){
            if(antallTrader == 0) return null;
            tomListe.await();
        }
        ArrayList<Integer> returTall = new ArrayList<>();
        returTall.add(tall.remove(0));
        returTall.add(tall.remove(0));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



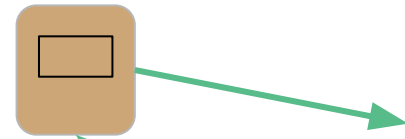
Da stikker jeg

returTall

1	5
---	---

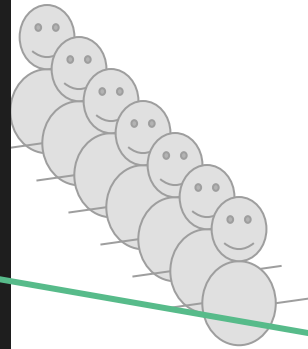


# BRUK MONITOR DESIGN PRINSIPPET



```
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();
    }
}
```



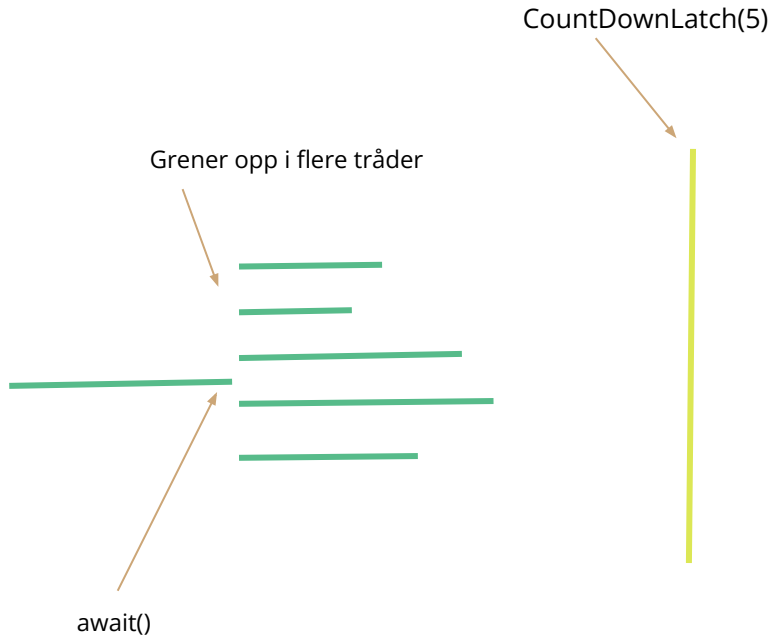
# CountDownLatch

- Konstruktøren tar inn antall tråder den skal vente på
- Nyttige metoder:
  - `await()`, Tråden blir stoppet her og står å venter til counteren 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 counteren (sagt med andre ord hvor langt den har kommet i nedtellingen)

DOKUMENTASJON: <https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/CountDownLatch.html>

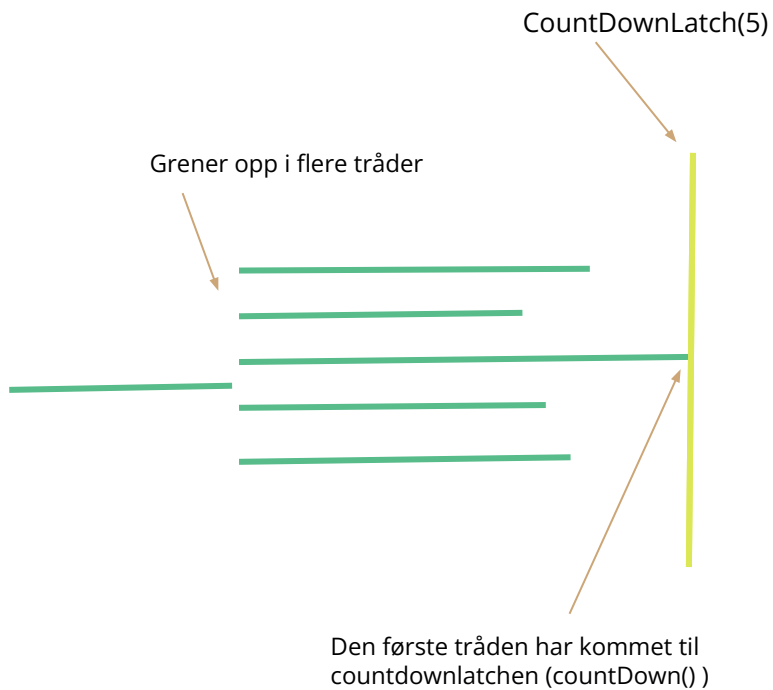
```
class EksempelCountDwon {  
  
    private static int antallTraader = 5;  
    Run | Debug  
    public static void main(String[] args) {  
        CountdownLatch cdl = new CountdownLatch(5);  
        for(int i = 0; i < antallTraader; i++){  
            new Thread(new CountdownTraad(cdl)).start();  
        }  
        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();  
    }  
}
```



```
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();  
    }  
}
```

tid



```

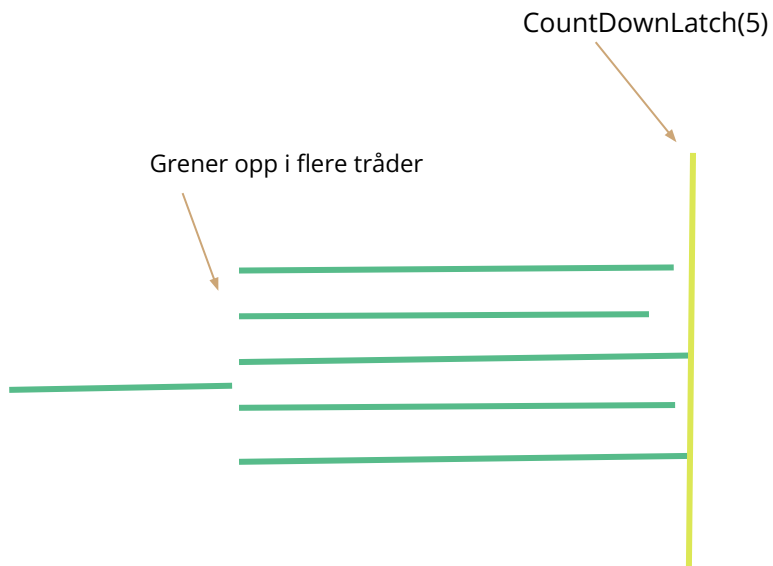
class EksempelCountDwon {
    private static int antallTraader = 5;
    Run | Debug
    public static void main(String[] args) {
        CountdownLatch cdl = new CountdownLatch(5);
        for(int i = 0; i < antallTraader; i++){
            new Thread(new CountdownTraad(cdl)).start();
        }
        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();
    }
}

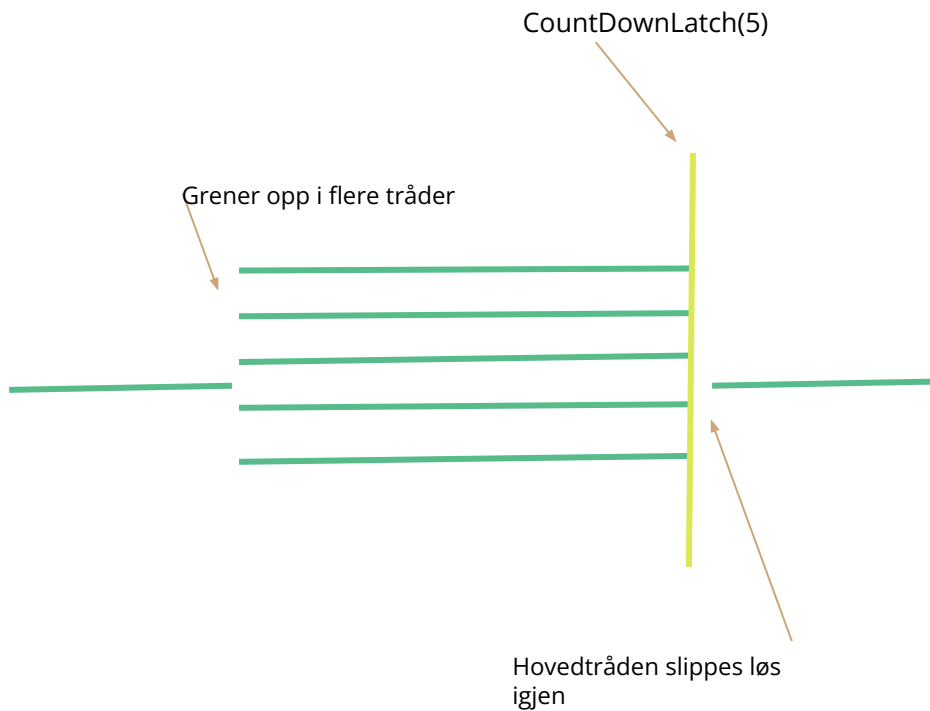
```



```
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();  
    }  
}
```

tid





```

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();
    }
}

```

# Skriv i menti kode: 5351 3306

En forventet utskrift i terminalen etter at koden er kjørt.

PS: her er det flere riktige svar.

```
class EksempelCountDown {  
    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)

```
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");  
        }  
    }  
}
```

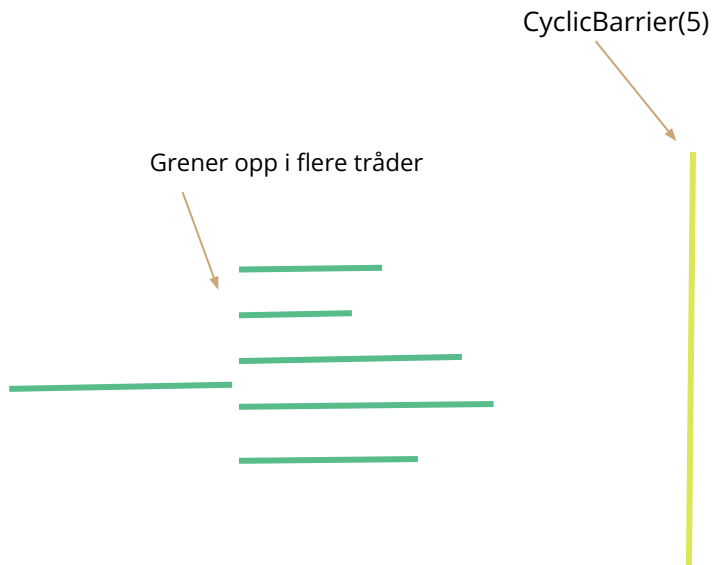
Grener opp i flere tråder



```
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");  
        }  
    }  
}
```

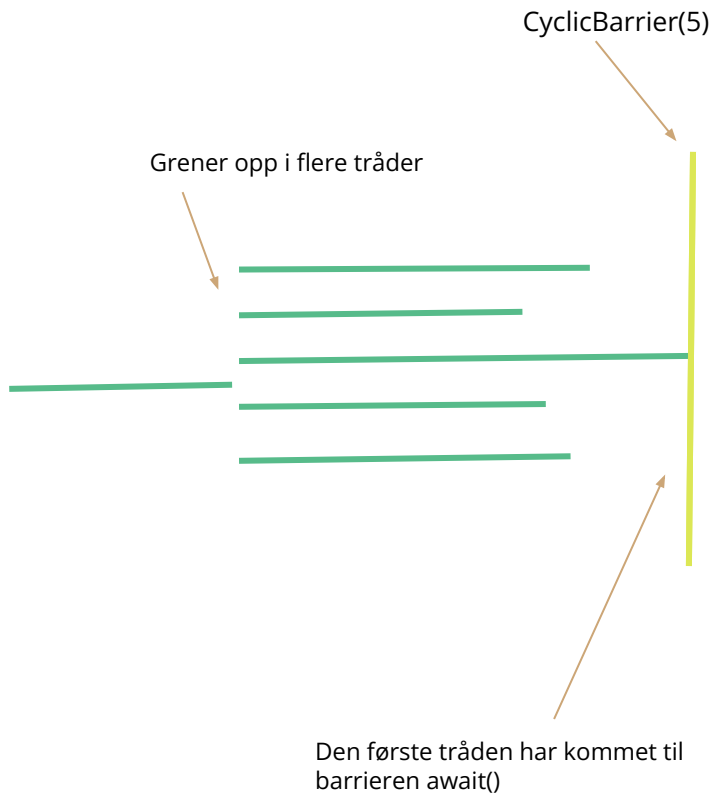


tid



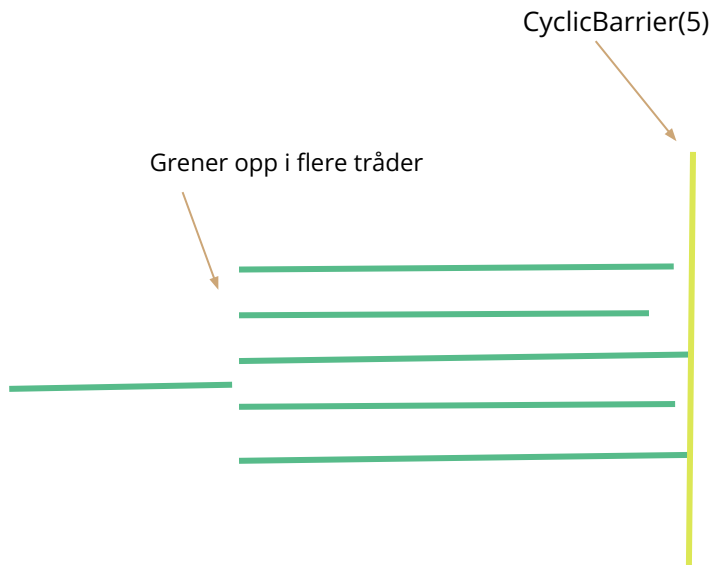
```
class EksempeCyclic {  
    private static int antallTraader = 5;  
    Run|Debug  
    public static void main(String[] args) {  
        CyclicBarrier cb = new CyclicBarrier(5);  
        for(int i = 0; i < antallTraader; i++){  
            new Thread(new CyclicBarrierTraad(cb)).start();  
  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

tid



```
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");  
        }  
    }  
}
```

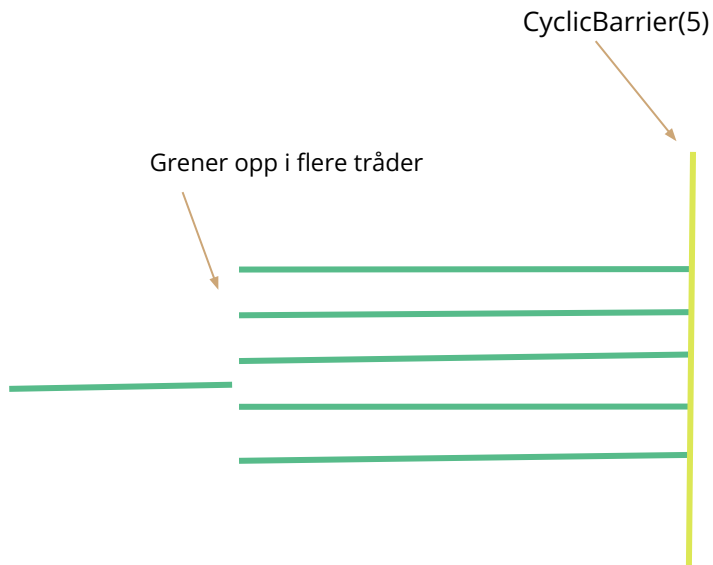
tid



```
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 første 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");  
        }  
    }  
}
```

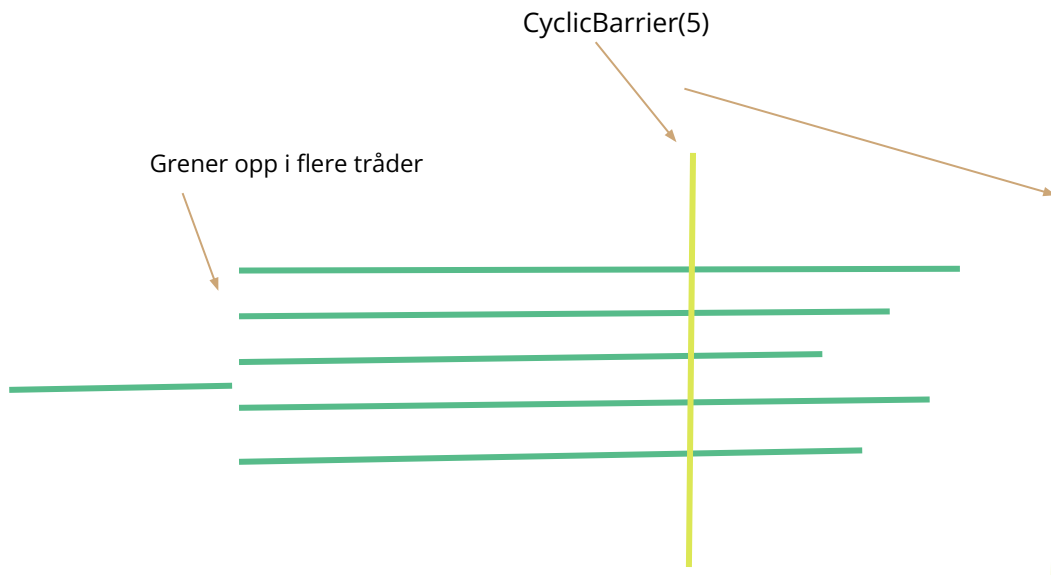






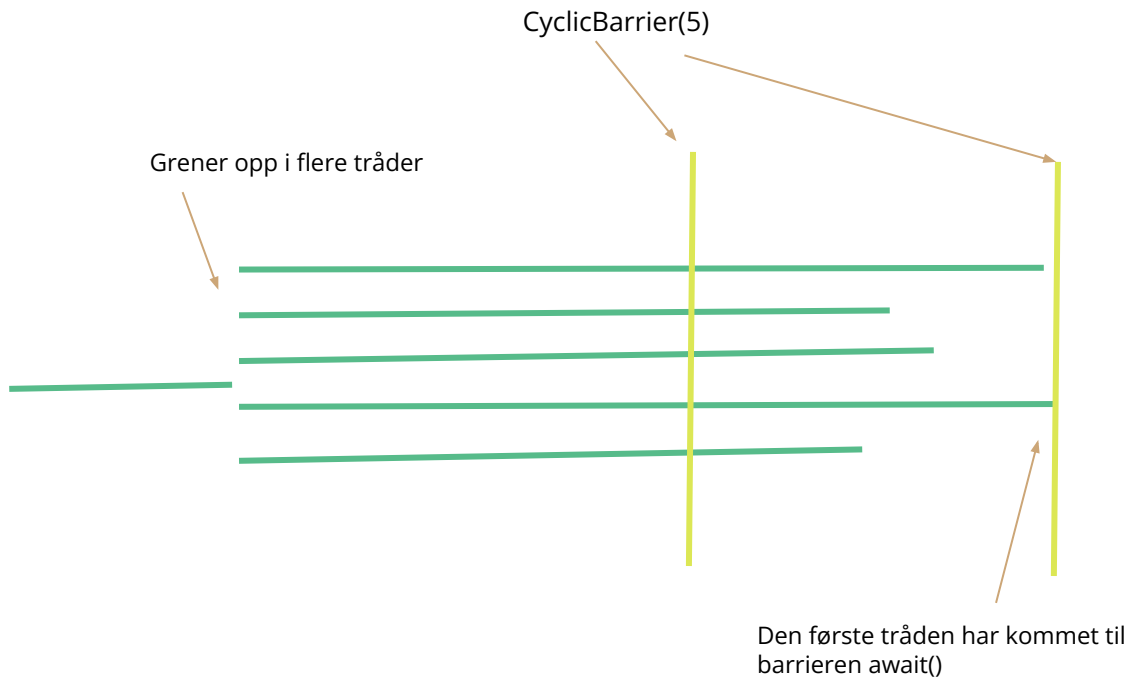
```
class EksempelCyclic {  
    private static int antallTraader = 5;  
    Run | Debug  
    public static void main(String[] args) {  
        CyclicBarrier cb = new CyclicBarrier(5);  
        for(int i = 0; i < antallTraader; i++){  
            new Thread(new CyclicBarrierTraad(cb)).start();  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





```
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");  
        }  
    }  
}
```

tid →



```

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;

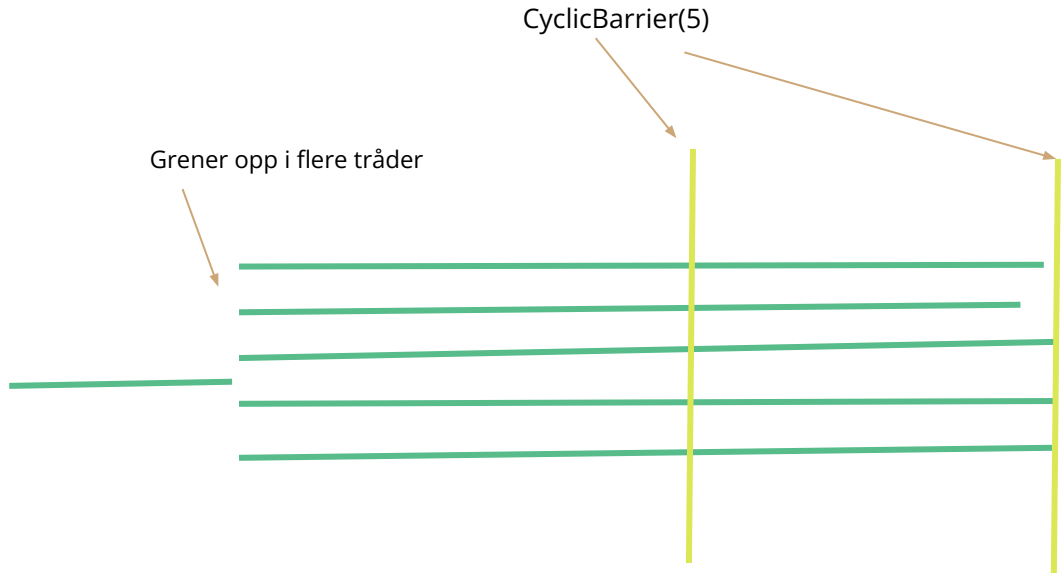
    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @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");
        }
    }
}

```

tid



```

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();
        }
    }
}

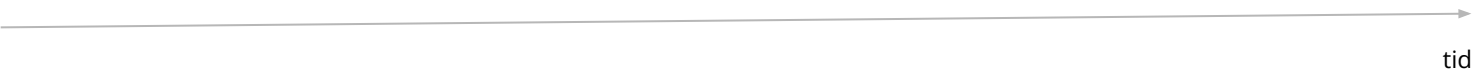
class CyclicBarrierTraad implements Runnable{
    CyclicBarrier cb;

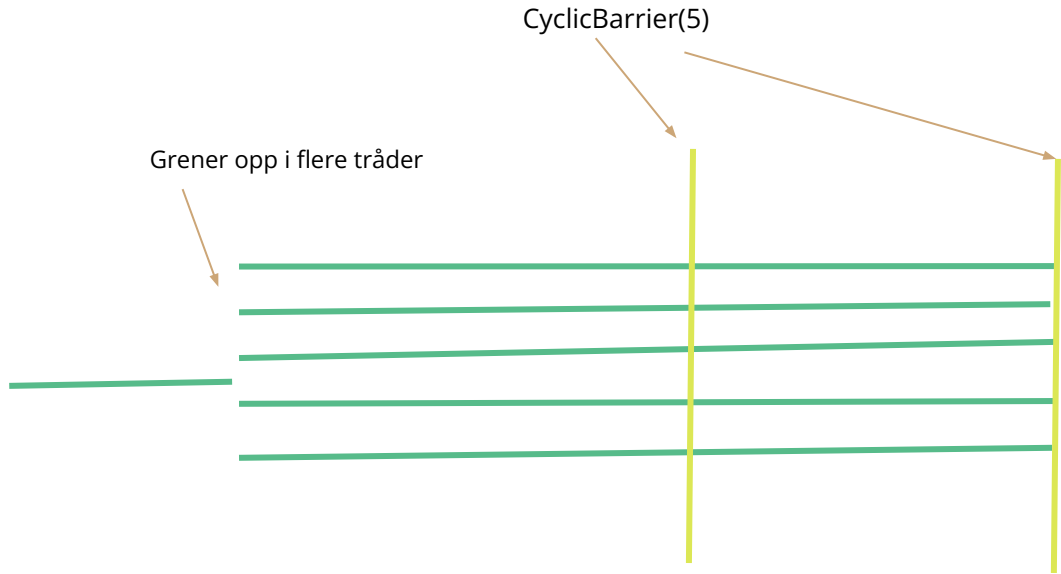
    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @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");
        }
    }
}

```





```

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();
        }
    }
}

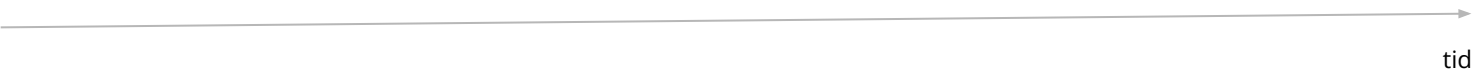
class CyclicBarrierTraad implements Runnable{
    CyclicBarrier cb;

    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");
        }
    }
}

```



# Skriv i menti kode: 5351 3306

Forventet utskift i terminalen etter at koden er kjørt.

PS: Her er det bare ett riktig svar

```
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");  
        }  
    }  
}
```

# 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)

```
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

Svar på menti, kode: 5351

3306

PS: her er det kun et riktig svar

```
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");  
    }  
}
```

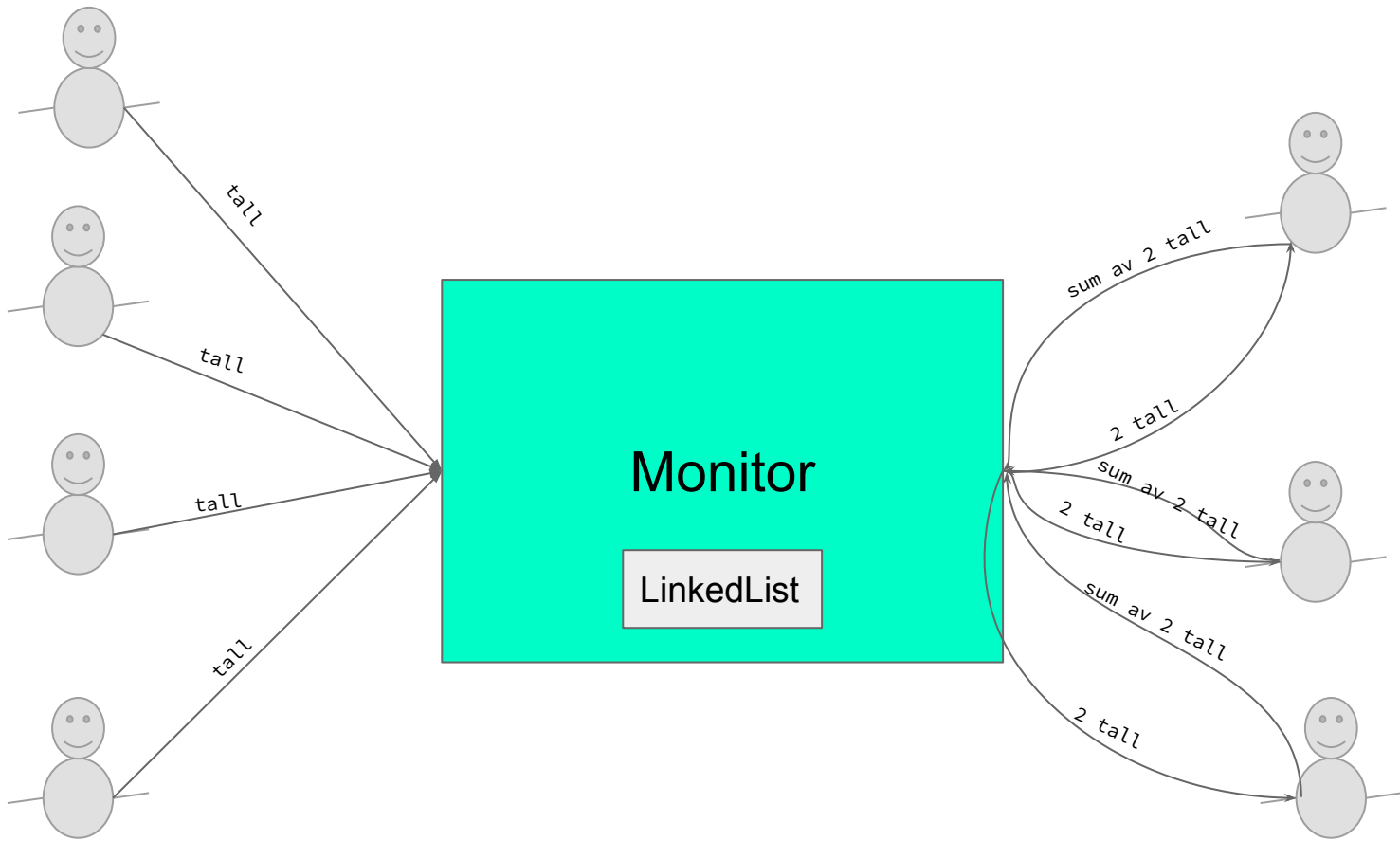


# LIVE-PROGRAMMERING (DEL 4)

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 :))



NOEN SPØRSMÅL ??