Velkommen!

•••

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

Kort: Praktisk informasjon

- Undervisningstilbud
  - https://www.uio.no/studier/emner/matnat/ifi/IN1010/v21/undervisningstilbud/
  - Jeg har konkrete spørsmål/problemer med min kode -> Labtime!
  - Jeg vil ha mer liveprogrammering -> Plenumstime!
  - Jeg vil jobbe med andre (og kanskje en kjapp recap av forelesning) -> Gruppetime!
  - Jeg vil ha en recap av de vanskeligste konseptene fra forelesning -> Repetisjonsgruppe!
- Skriv alt dere lurer på i chatten enten til everybody eller bare til meg 😊
  - Si i fra hvis dere faller av! Enten hvis noe er vanskelig, eller hvis dere glemte å følge med! Bare si i fra!
- Start på oblig5 ASAP! Ingen/lite hjelp å få i påskeferien.
  - Dere har 2uker + påske på å gjøre den 😊

Repetisjon forrige uke

Forrige gang

Tråder

Lock

Monitor

Condition

Send meg (Johanna) en direktemelding i chatten

Venter til alle har sendt meg svar:

1. Hvorfor kan tråder være lurt?
2. Hvordan lager vi tråder?
3. Hva er en lock?
4. Hva er en condition?

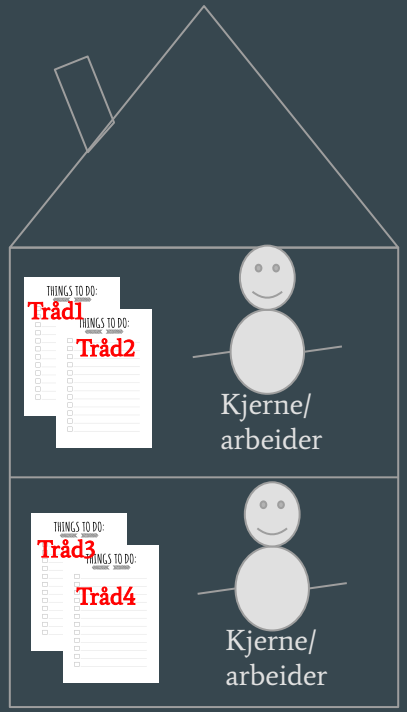
Hvorfor trenger vi tråder?

Kjernene finnes i CPUen, en CPU kan ha flere kjerner(vanlig), og en datamaskin kan ha flere CPUer(ikke så vanlig).

1. Hvis flere kjerner kan jobbe på programmet vårt samtidig går det raskere.
2. Hvis en kjerne kan jobbe på ulike deler av programmet vårt “samtidig” går det raskere.

Skjer ikke egentlig samtidig, de bytter på hva de jobber med

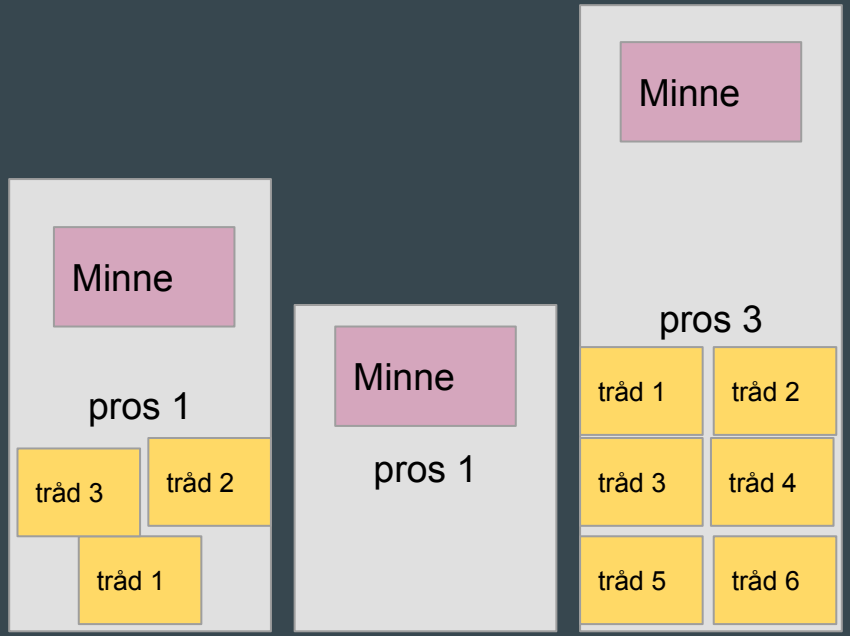
Både 1) og 2) løses med tråder(gjøre det mulig å dele opp programmet).



CPU/kontoret/fabrikken

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



Tråder step by step (fra boka)

1. Write a class that implements the `Runnable` interface. That interface has a single method called `run`:

```
public interface Runnable
{
   void run();
}
```

2. Place the code for your task into the `run` method of your class:

```
public class MyRunnable implements Runnable
{
   public void run()
   {
      Task statements
      . . .
   }
}
```

3. Create an object of your subclass:

```
Runnable r = new MyRunnable();
```

4. Construct a `Thread` object from the runnable object:

```
Thread t = new Thread(r);
```

5. Call the `start` method to start the thread:

```
t.start();
```

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

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

Repetisjon denne uken

I dag

Tråder

CountDownLatch

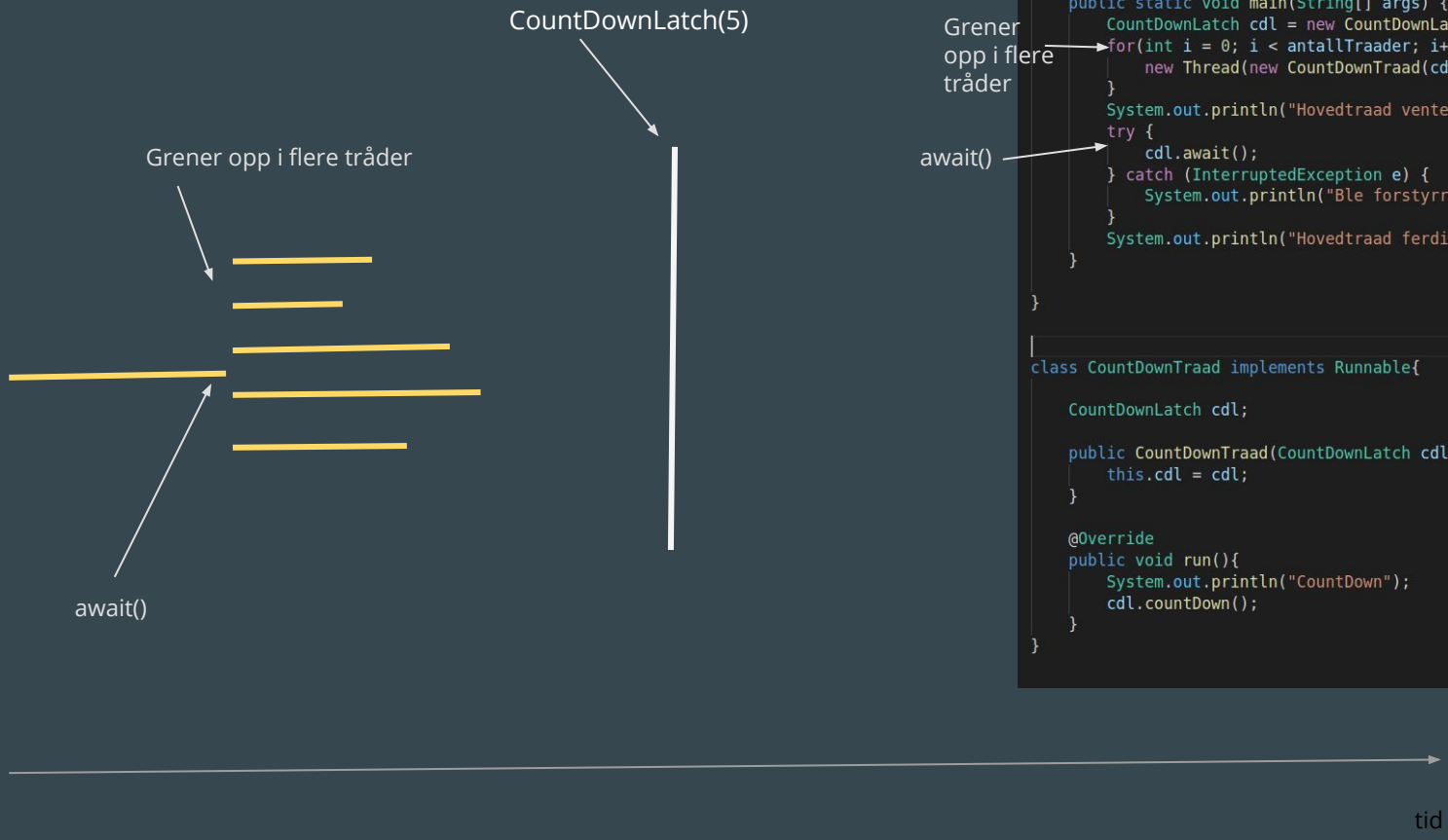
Cyclic barrier

CountDownLatch

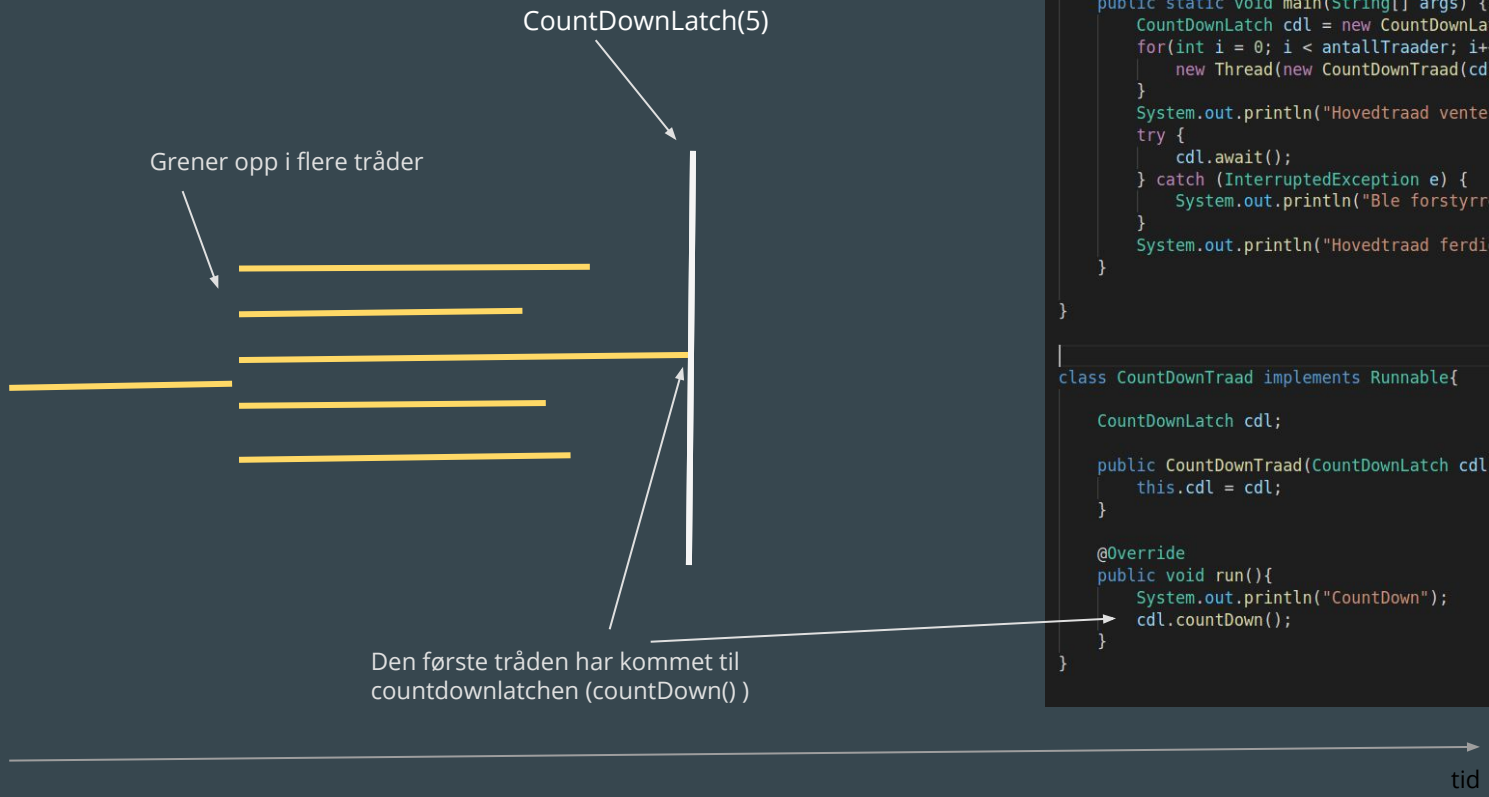
- Konstruktøren tar inn antall tråder den skal vente på
- Nyttige metoder:
  - await(), Tråden blir stoppet her og står å venter til countern har telt seg ned til 0. Denne metoden kan kaste unntak! (Derfor må man bruke try-catch)
  - countDown(): Teller ned counteren en gang.
  - getCount(): returnerer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

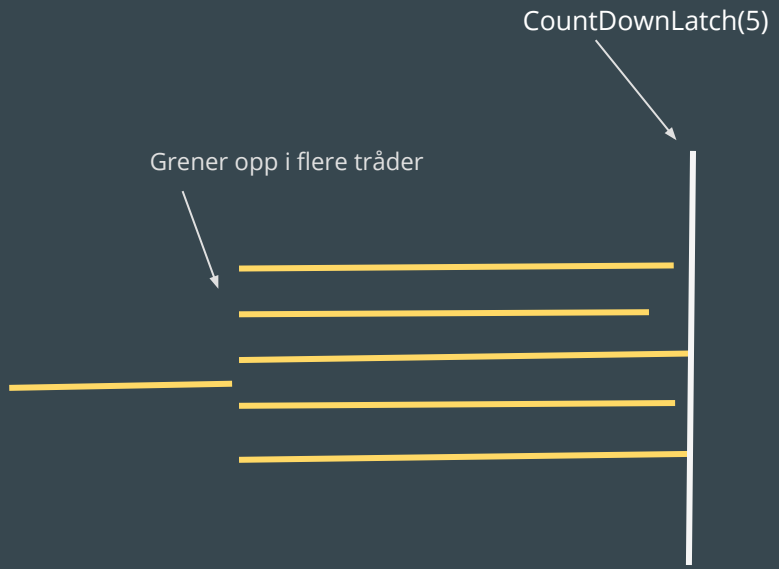



```java
class EksempelCountDwon {

    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");
    }

}

class CountDownTraad implements Runnable{

    CountDownLatch cdl;

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

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

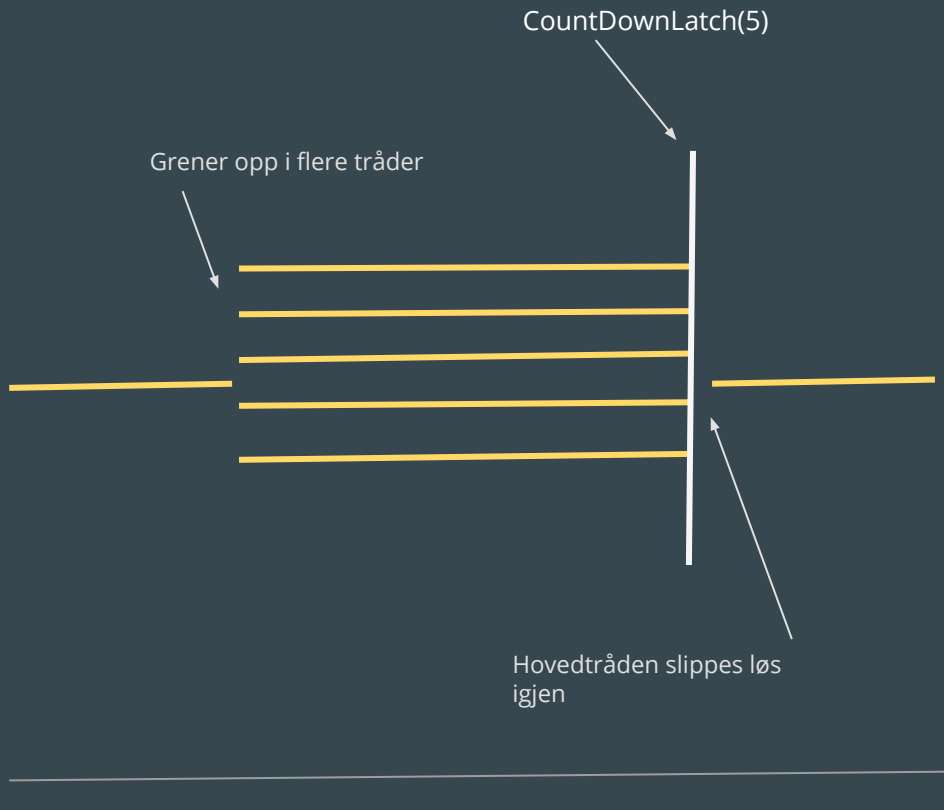



```java
class EksempelCountDwon {

    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");
    }

}

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();
    }
}
```

Send meg(Johanna) en direktemelding i chatten

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();
    }
}
```

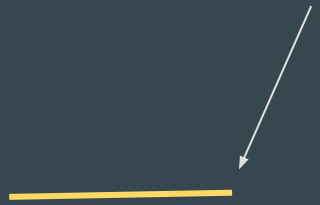
CyclicBarrier

- 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");
        }
    }
}
```

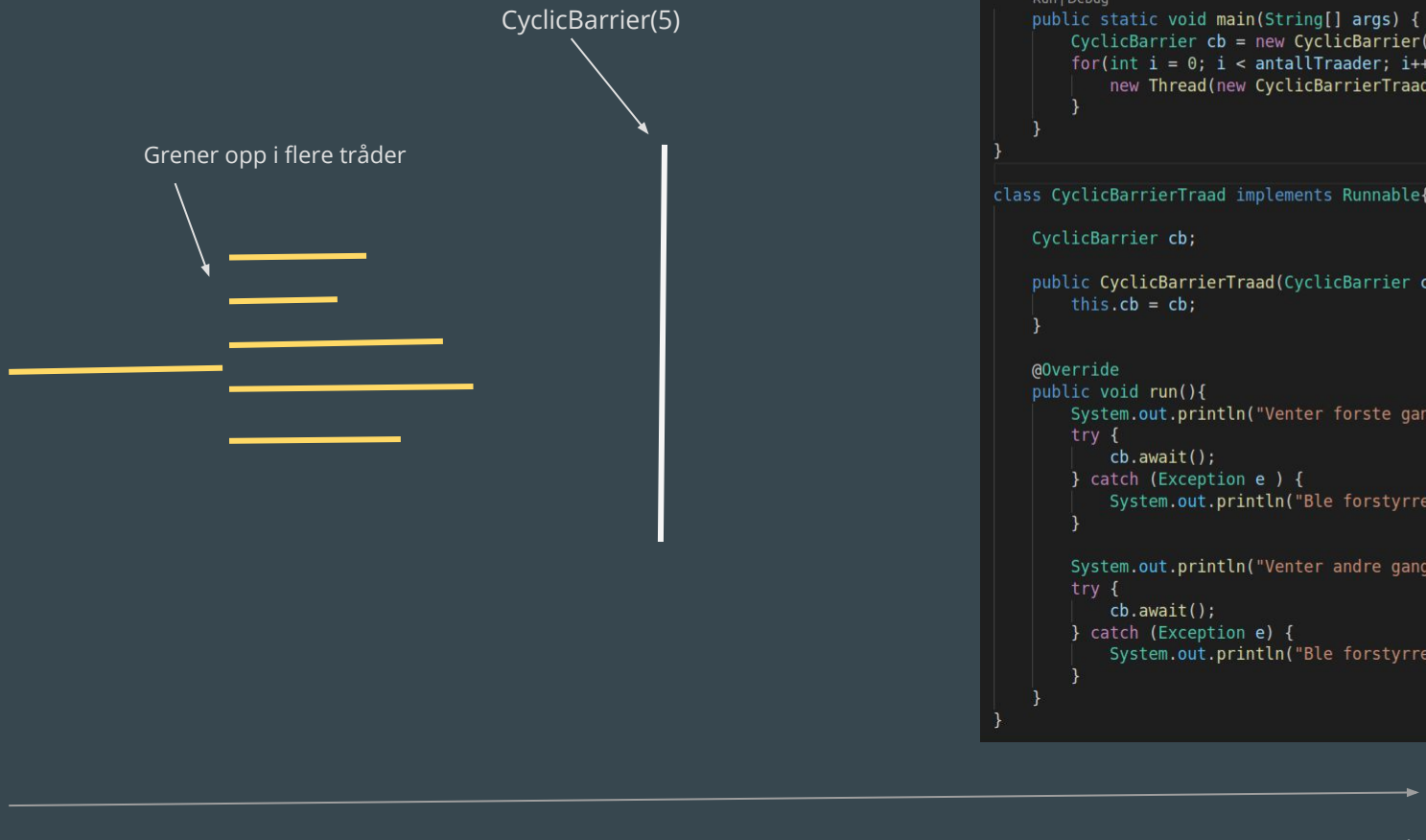
Grener opp i flere tråder



```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();
        }
    }
}
```

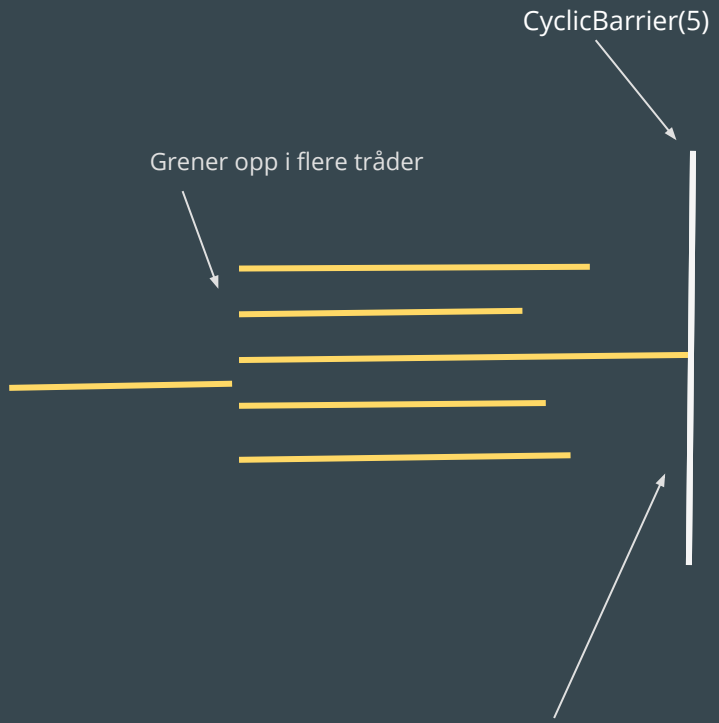
```java
class CyclicBarrierTraad implements Runnable{

    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



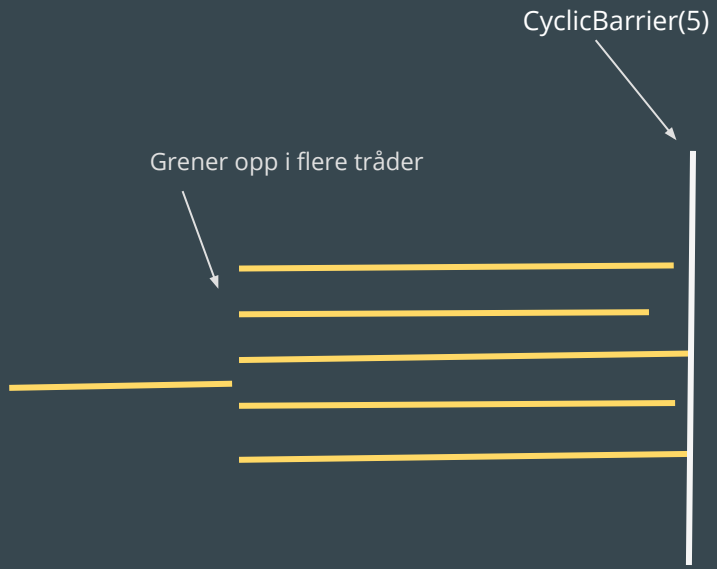

```java
class EksempelCyclic {

    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();
        }
    }
}
```

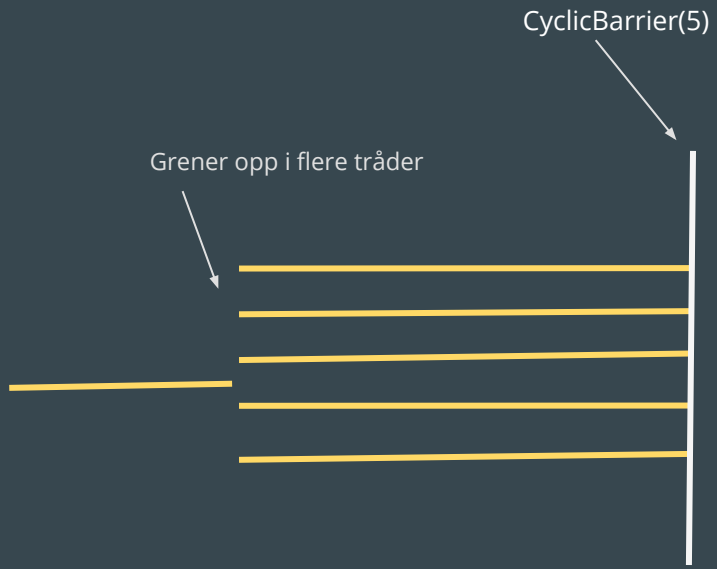
```java
class CyclicBarrierTraad implements Runnable{

    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");
        }
    }
}
```

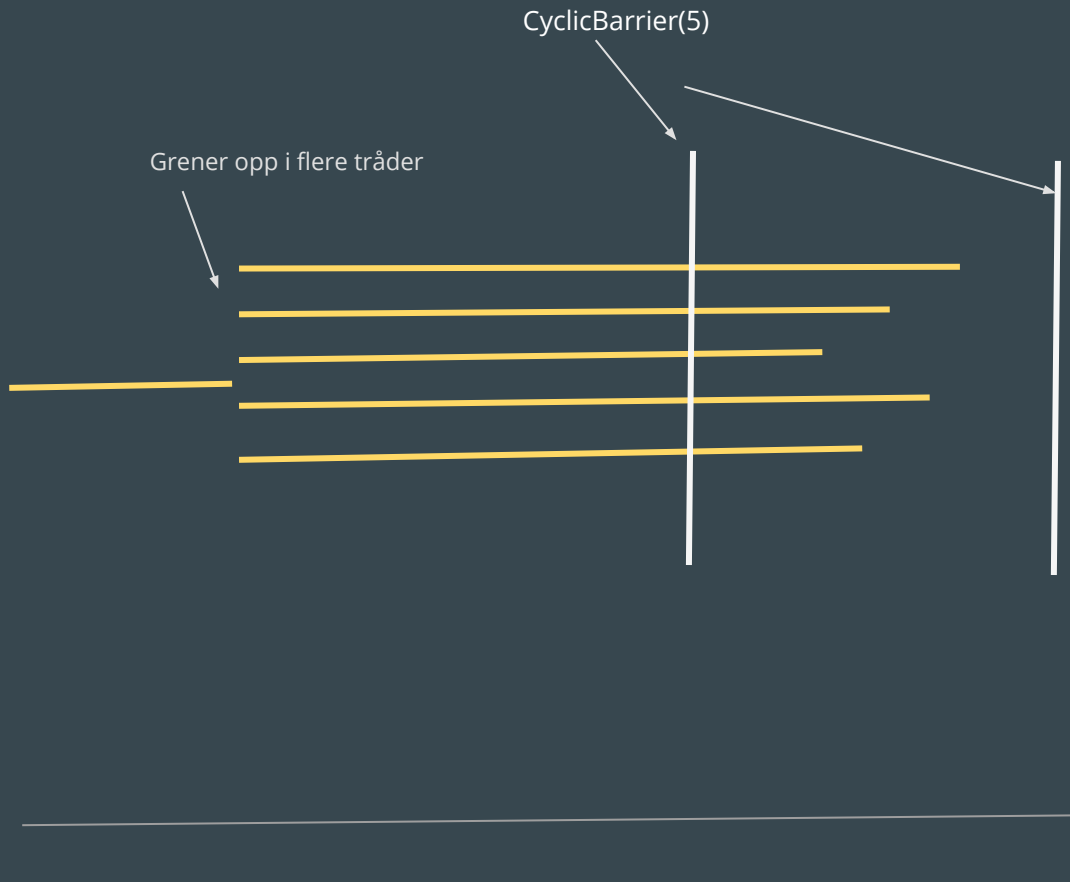
tid




```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();
        }
    }
}
```

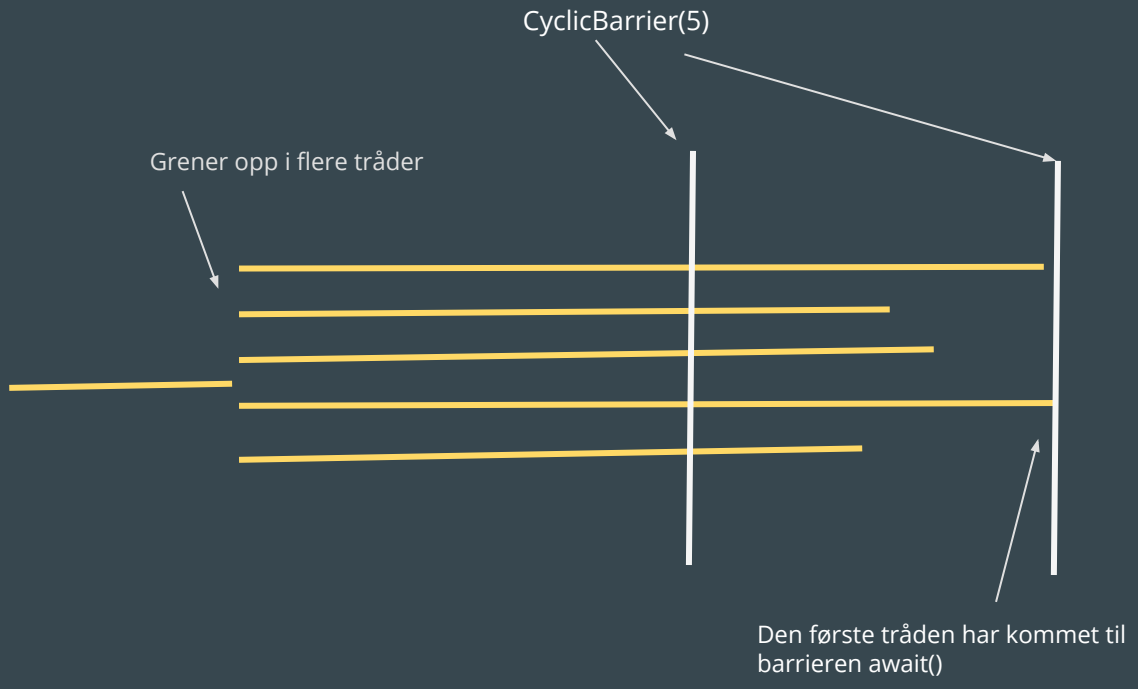
```java
class CyclicBarrierTraad implements Runnable{

    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");
        }
    }
}
```

tid



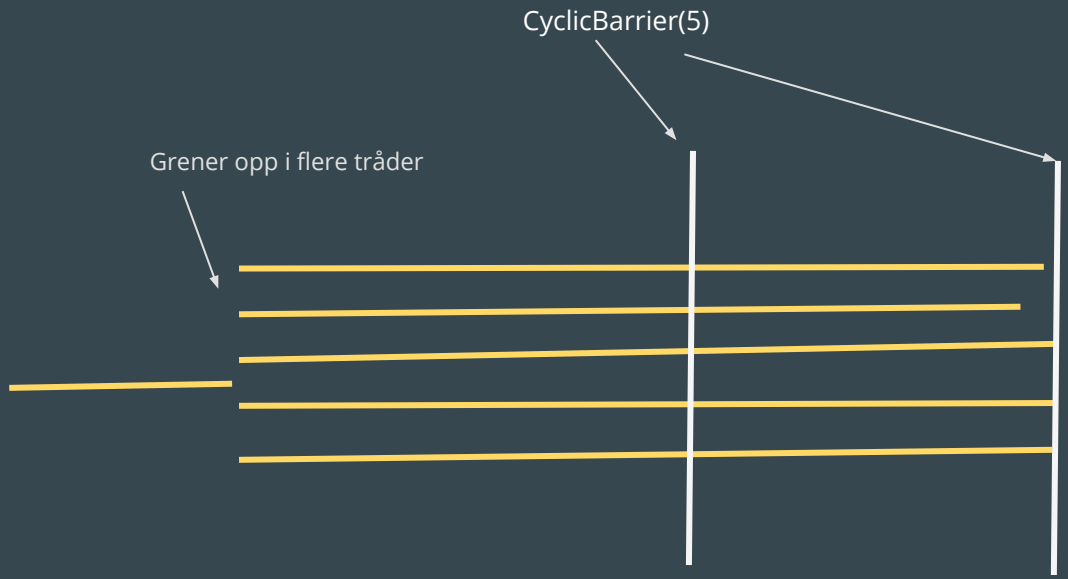

```java
class EksempelCyclic {

    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");
        }
    }
}
```



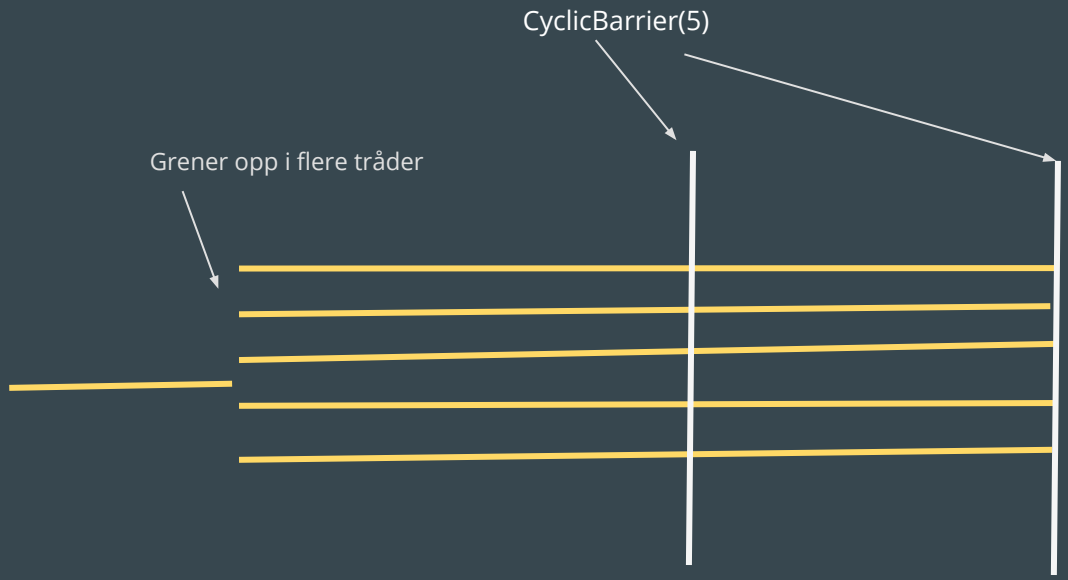

```java
class EksempelCyclic {

    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");
        }
    }
}
```



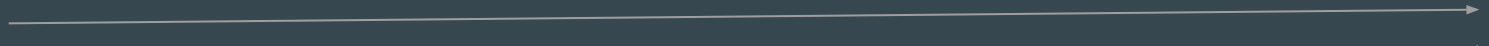

```java
class EksempelCyclic {

    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");
        }
    }
}
```



Send meg(Johanna) en direktemelding i chatten

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");
    }
}
```

Send meg(Johanna) en direktemelding i chatten

Hva 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");
    }
}
```

Enum

- Når vi ønsker at en variabel skal være av en forhåndsbestemt type konstant. Ved hjelp av et enum så bestemmer vi at variabelen må være av de forhåndsbestemte typene.
- Deklareres som en klasse eller interface, men nå med ordet “enum”.

LIVE-KODING

Jobbe selv

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