



# IN1010 - Seminar 10

- Tråder
- 

# Praktisk

Husk å sjekke emnesiden regelmessig

Undervisningstilbud:

- <https://www.uio.no/studier/emner/matnat/ifi/IN1010/v21/undervisningstilbud/>
- Jeg har konkrete spørsmål/problemer med min kode -> Labtime!
- Jeg vil ha mer liveprogrammering -> Plenumstime!
- Jeg vil jobbe med andre (og kanskje en kjapp recap av forelesning) -> Gruppetime!
- Jeg vil ha en recap av de vanskeligste konseptene fra forelesning -> Repetisjonsgruppe!

Oblig 5 er nå publisert (ca. 3 uker på dere). HUSK at denne skal inn rett etter påsken, som betyr at det ikke vil være noen labtimer/gruppetimer og at gruppelærere/faglærere ikke er tilgjengelig den siste uken på mattermost. Kan derfor være lurt å prøve å bli ferdig før ferien stater.



# Oppmøteregistrering:

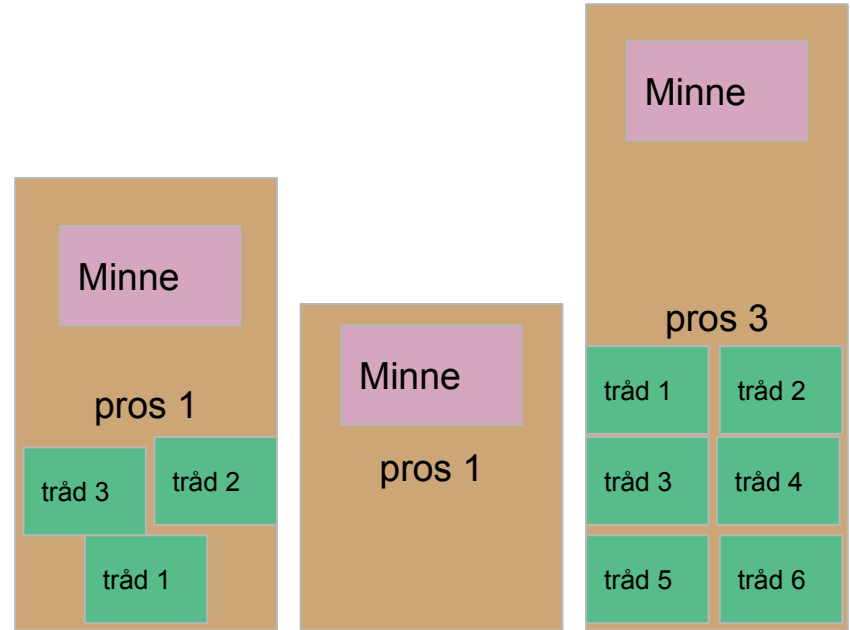
<https://nettskjema.no/a/188717>



Repetisjon fra forrige uka

# 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

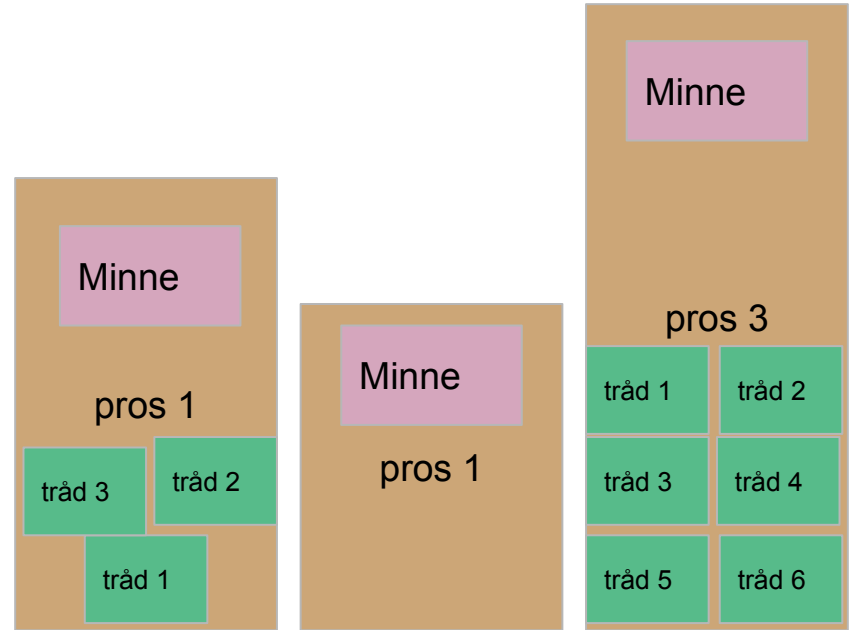


# Hvordan lage en tråd ??

- Runnable steg 1
  - <https://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html>
  - Et interface
  - 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 vår Runnable klasse i konstruktøren

# Delt data

- Data flere tråder deler
- Prosessens data
- Skrivning
- Lesing
- Lesing og skrivning
- Er du usikker lås



Repetisjon fra denne uka



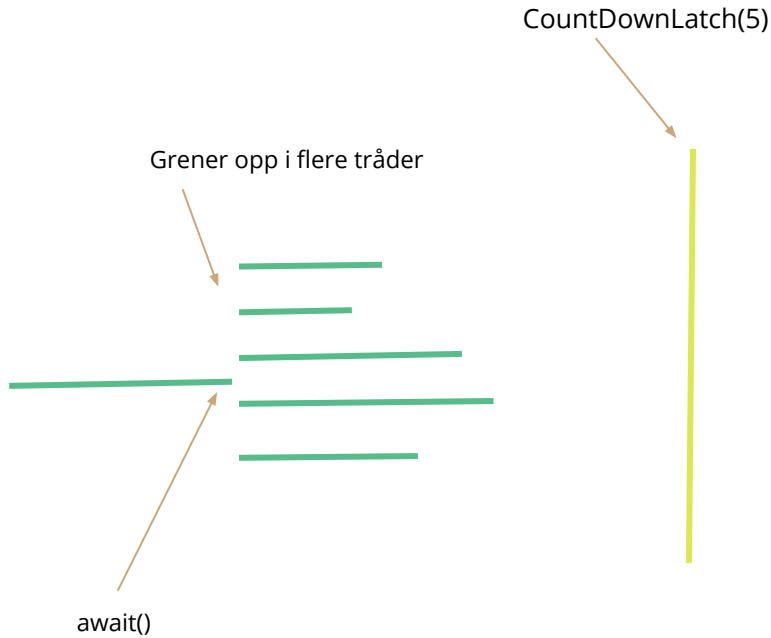
# CountDownLatch

- 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 forstyrrret");  
        }  
        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");
    }
}

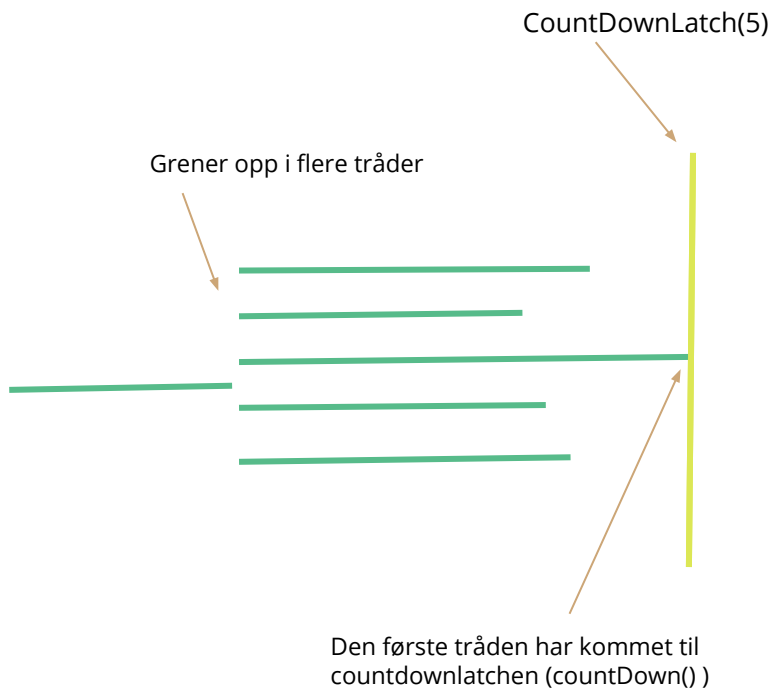
class CountdownTraad implements Runnable{
    CountdownLatch cdl;

    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");
    }
}

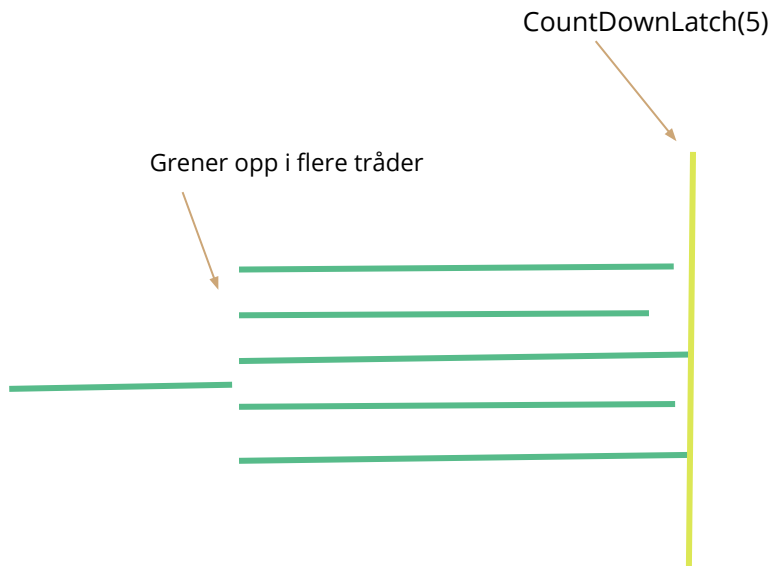
class CountdownTraad implements Runnable{
    CountdownLatch cdl;

    public CountdownTraad(CountdownLatch cdl){
        this.cdl = cdl;
    }

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

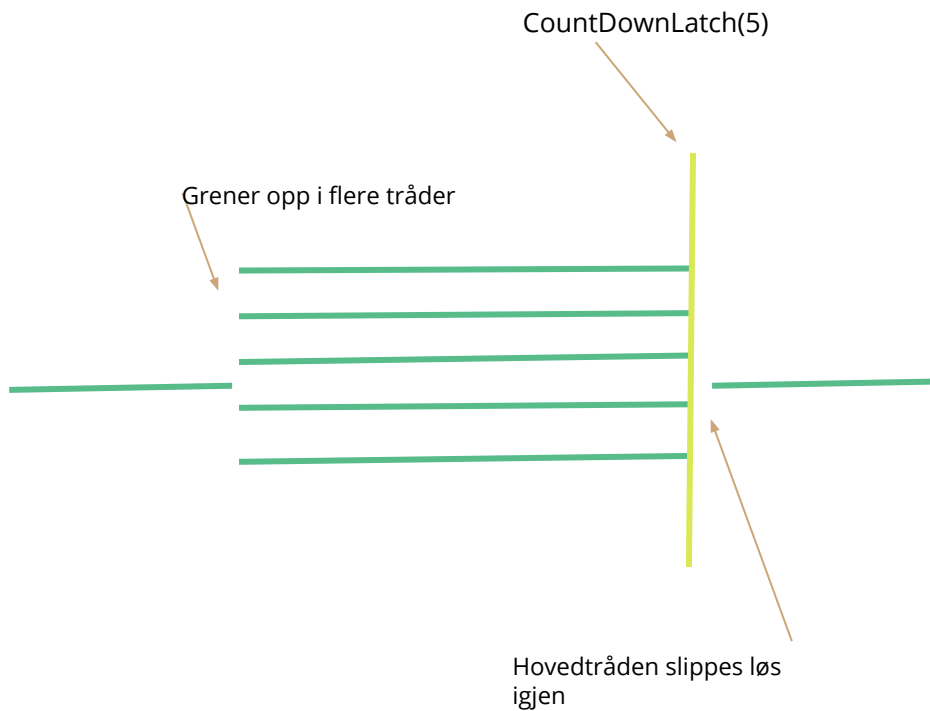
```

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();  
    }  
}
```

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 chatten til Marlen(chat)

En forventet utskrift i terminalen etter at koden er kjørt.

PS: her er det flere riktige svar.

```
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)



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

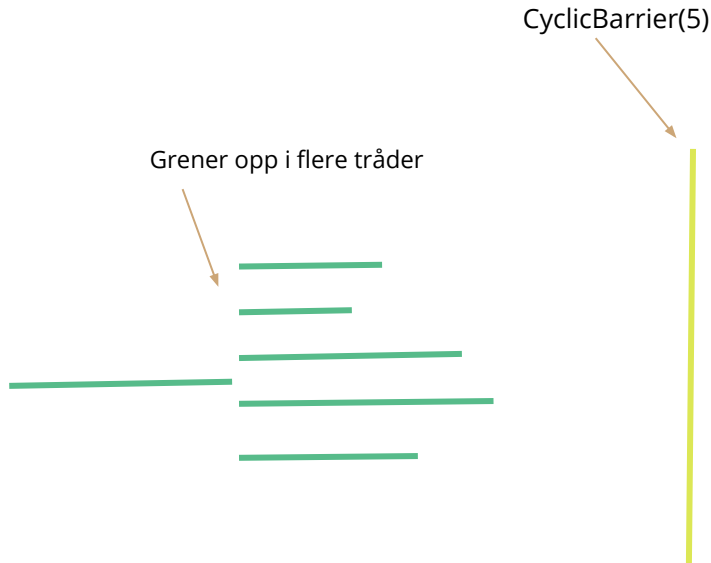
Grener opp i flere tråder



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

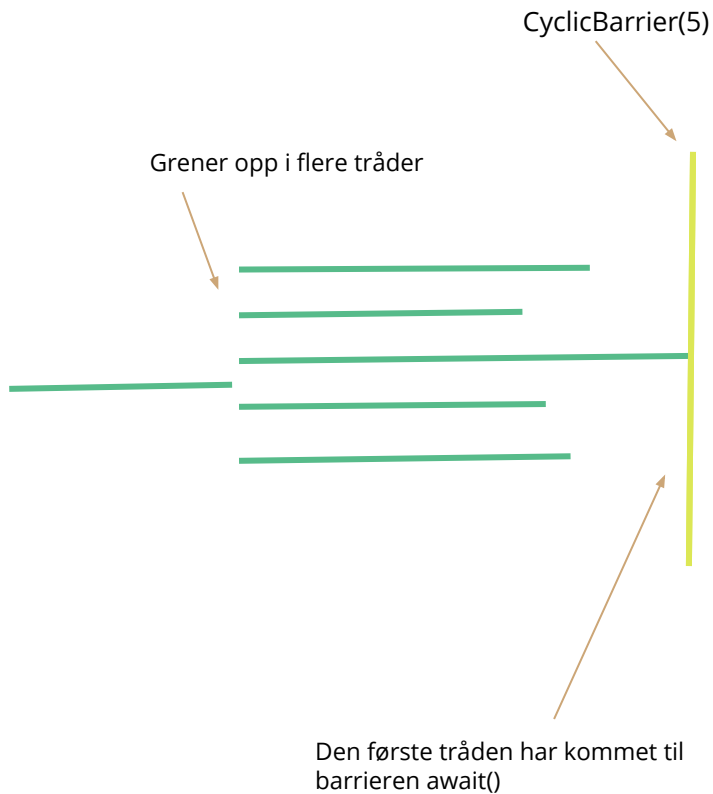


tid



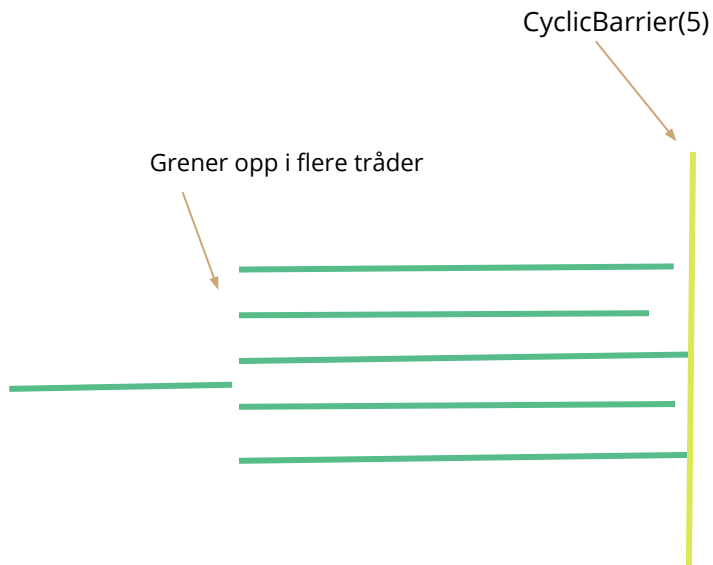
```
class EksempeCyclic {  
    private static int antallTraader = 5;  
    Run|Debug  
    public static void main(String[] args) {  
        CyclicBarrier cb = new CyclicBarrier(5);  
        for(int i = 0; i < antallTraader; i++){  
            new Thread(new CyclicBarrierTraad(cb)).start();  
  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





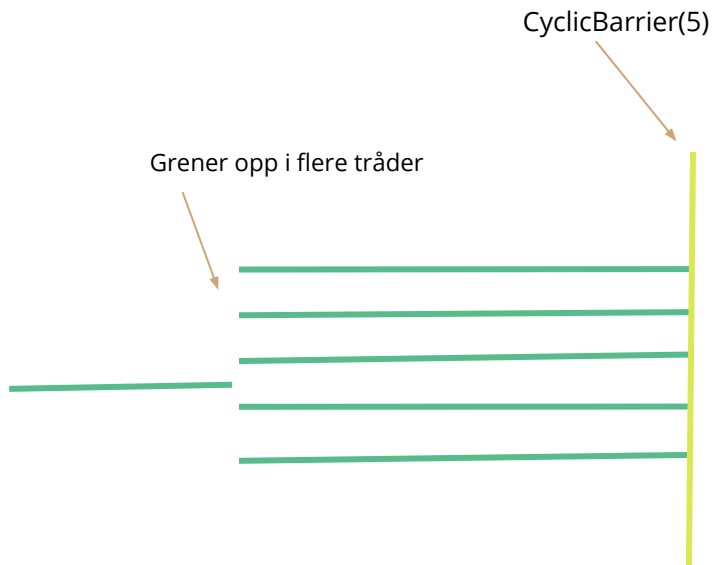
```
class EksempelCyclic {  
    private static int antallTraader = 5;  
    Run | Debug  
    public static void main(String[] args) {  
        CyclicBarrier cb = new CyclicBarrier(5);  
        for(int i = 0; i < antallTraader; i++){  
            new Thread(new CyclicBarrierTraad(cb)).start();  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



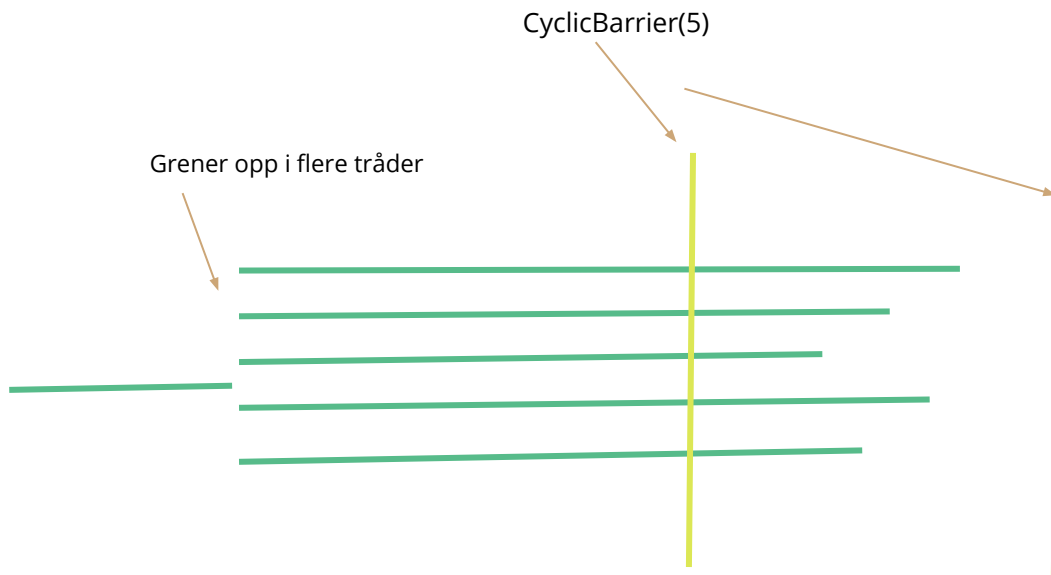
```
class EksempelCyclic {  
    private static int antallTraader = 5;  
    Run | Debug  
    public static void main(String[] args) {  
        CyclicBarrier cb = new CyclicBarrier(5);  
        for(int i = 0; i < antallTraader; i++){  
            new Thread(new CyclicBarrierTraad(cb)).start();  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

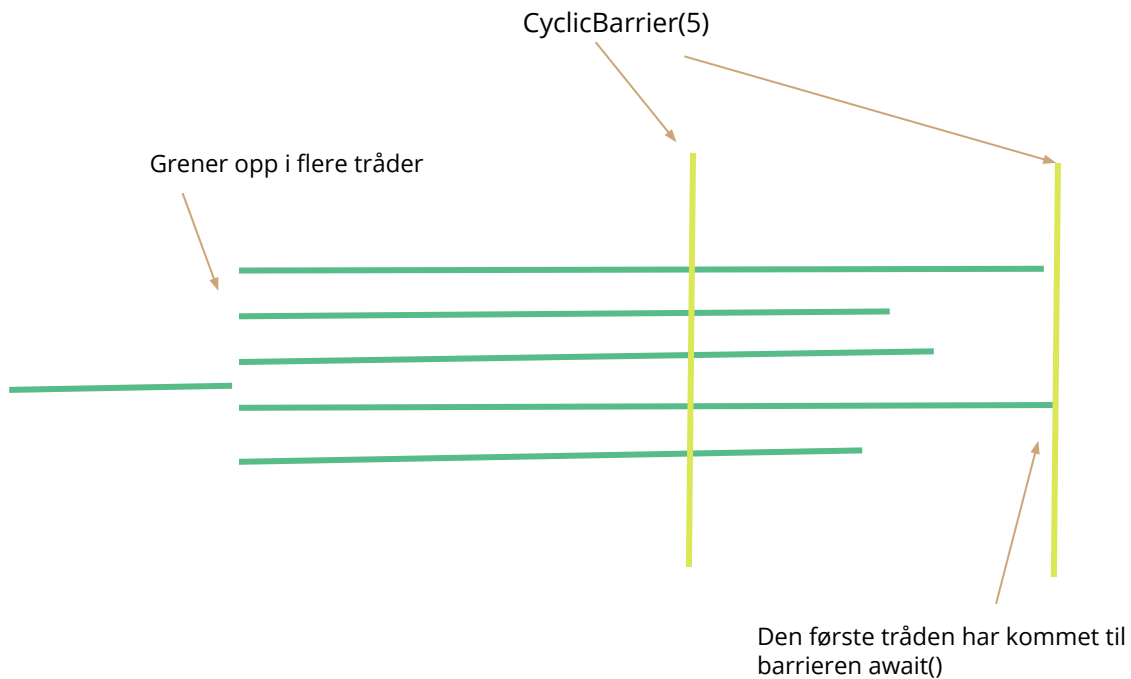
    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @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");
        }
    }
}

```

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;
    }

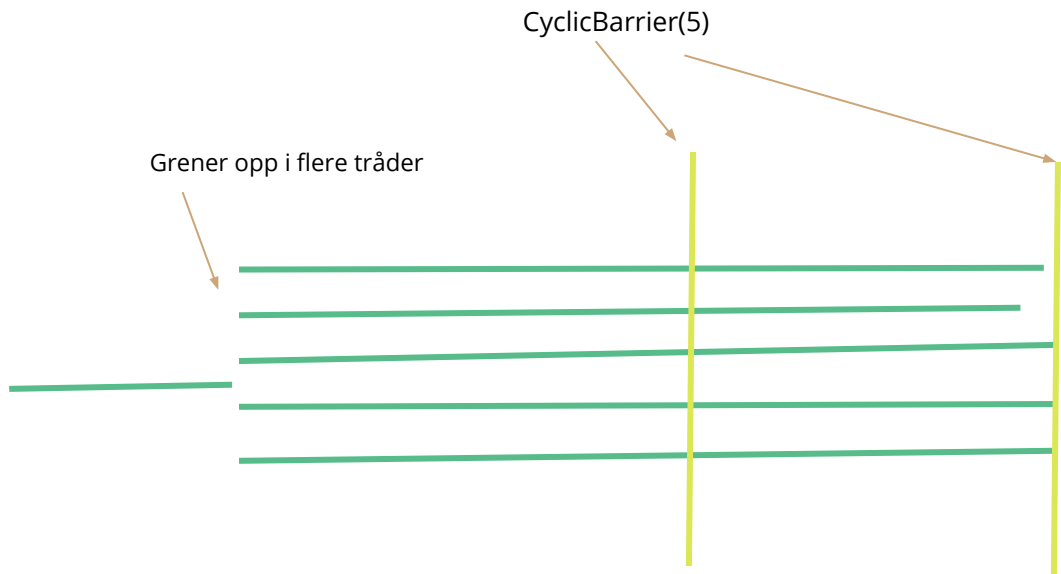
    @Override
    public void run(){
        System.out.println("Venter forste gang");
        try {
            cb.await();
        } catch (Exception e ) {
            System.out.println("Ble forstyrret");
        }

        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;

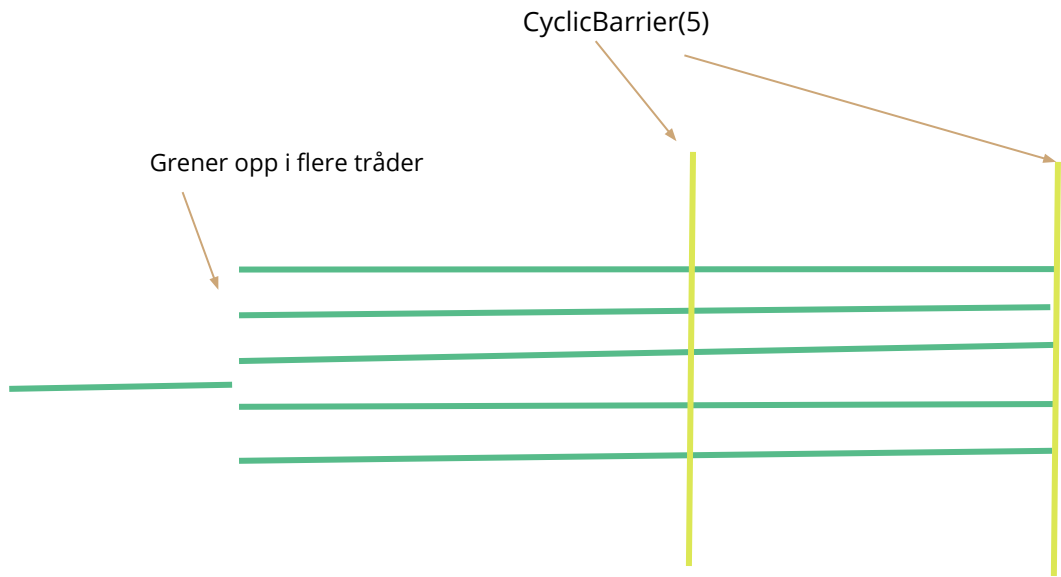
    public CyclicBarrierTraad(CyclicBarrier cb){
        this.cb = cb;
    }

    @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();
        }
    }
}

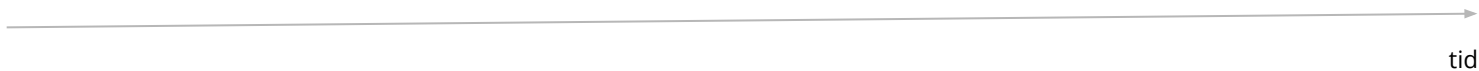
class CyclicBarrierTraad implements Runnable{
    CyclicBarrier cb;

    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 chatten til Marlen(chat)

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

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

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");  
    }  
}
```

# 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



# Oppmøteregistrering:

<https://nettskjema.no/a/188717>



# Ris, ros, forslag ?

<https://nettskjema.no/a/180345>



# Send Marlen(chat) en direkte melding i chatten

Vil du jobbe sammen med noen andre ? (ja /nei)

Hvis du har noen ønsker på hvem du vil jobbe med, så send det i samme melding

Svar gjerne også om svaret skulle være nei

# Breakoutrooms

1. Slå på kamera og ha en presentasjonsrunde
2. Jobb sammen med ukesoppgavene, de ligger på emnesiden -> grupper
  - a. Enten ved at én deler skjerm eller med [codecollab.io/](https://codecollab.io/)
  - b. OBS: codecollab er gratisjeneseter som UiO ikke har avtale med, sannsynligvis vil de samle data om dere. Dere kan fint løse oppgavene uten å bruke den tjenesten!
3. Bruk "ask for help"-knappen for å få hjelp 😊
4. Vi møtes her igjen for å gå gjennom oppgavene til slutt (dere bestemmer hvilke)

Jobbe med oppgaver