Mentimeter

Tråder del 2

uke 10 gruppe 1

Dagens plan

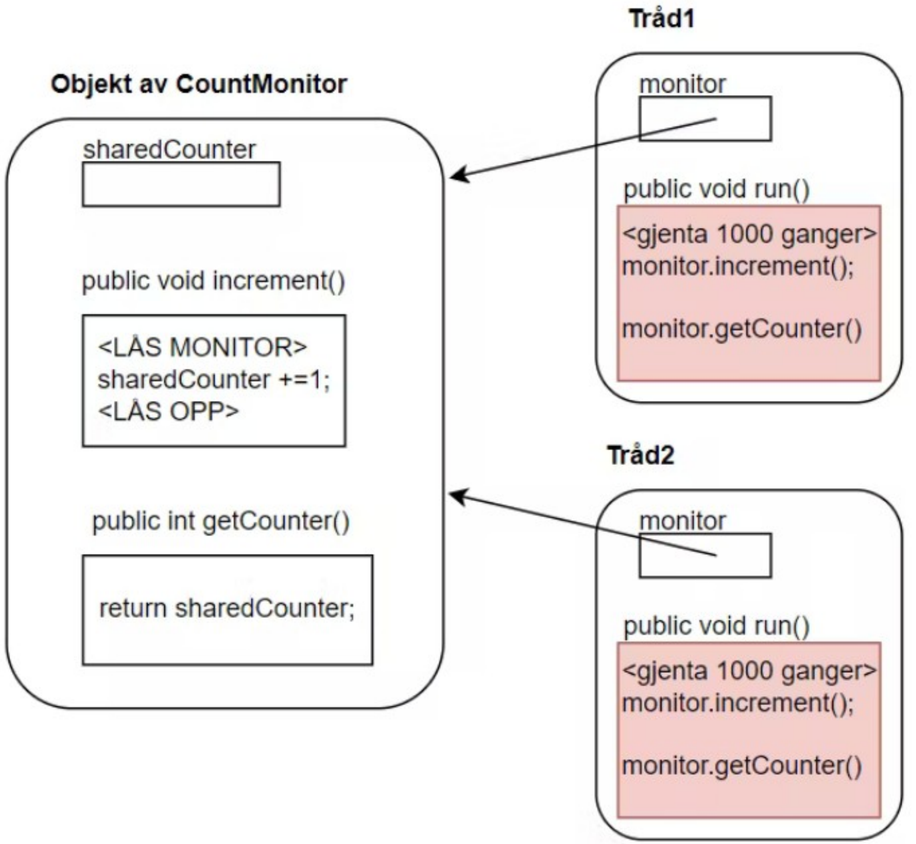
- Gjennomgang av forelesning
- Live koding
- Oppgavejobbing

Monitor

- Tråder kan jobbe på samme data (eks hente ut og endre en int)
- Monitor er et objekt som innkapsler delt data
- Monitor har synkroniserte metoder for å jobbe med data
  - Synkroniserte metoder gjør at kun en tråd kan jobbe på ressursen (lock og unlock)

Huskeregler for tråder

- Trådene (run-metodene) kaller på de synkroniserte metodene i monitor for å aksessere delt data
- Ingen låsing, venting eller signalisering i trådene (run), kun i monitor
- Men en tråd kan kalle på Thread.sleep(), .interrupt() eller .join()



Eks Program som teller. live koding

Conditions

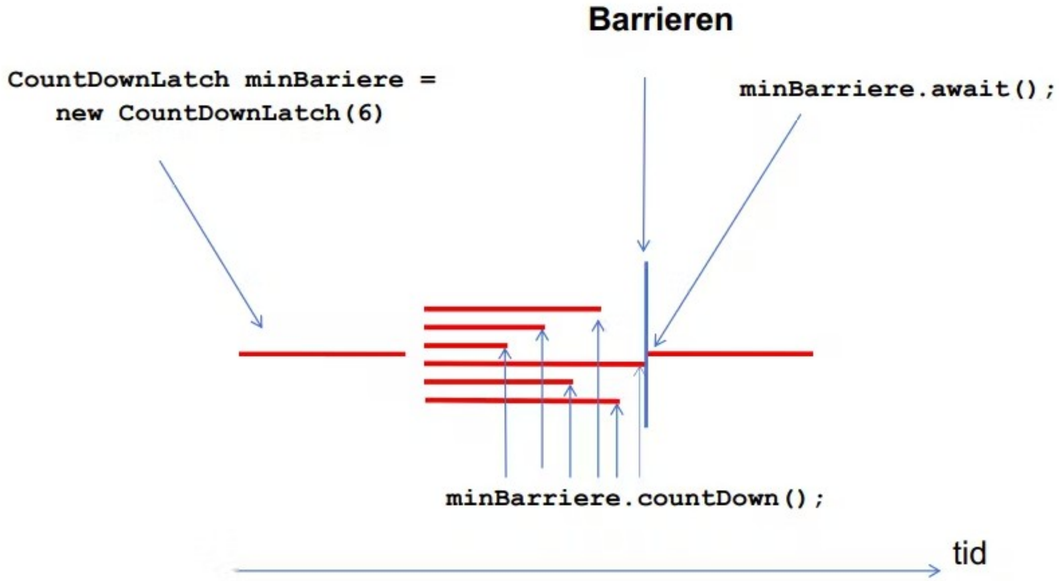
- Noen ganger må en tråd vente på at en betingelse er oppfyllt før den kan gjøre noe
- Vi bruker java.util.concurrent.locks.Condition for å oppnå dette
- En Condition kan vi tenke på som en ventekø tilknyttet en monitor
- Vi kan plassere tråder i denne køen, i påvente av at en betingelse skal bli sann
- Når dette skjer kan vi signalisere tråden, slik at den kan fortsette
- eks restaurant

```
interface Condition {
    void await();
    void signal();
    void signalAll();
}
```

Mentimeter

Barrierer

- En barriere er et samlingspunkt i koden
- Alle trådene starter samtidig, noen tråder er raskere enn andre.
- En barriere får trådene til å vente til alle trådene har kommet seg fram til barrieren
- Deretter kan alle trådene fortsette samtidig



Flere måter å lage barriere

- .join() fra forelesning. Legger alle tråder i en array og starter dem opp i for-løkke.
  - Så går vi i en for-løkke igjen og går gjennom alle trådene i arrayen og kaller på join()
- CountDownLatch(int count) - Alle slipper gjennom barrierer når .countDown() har blitt kalt count ganger.
  - Vi kan velge å vente ved barrieren med .await
- CyclicBarrier(int count) - Når count tråder har kalt på .await(), slipper alle gjennom
  - Barrieren kan så brukes på nytt på samme måte.

Mentimeter

```
class Arbeider implements Runnable {
   private final CountDownLatch ferdigSignal;

   Arbeider(CountDownLatch ferdig) {
      ferdigSignal = ferdig;
   }
   public void run() {
      try {
        System.out.println("Gjor arbeid!");
        gjorArbeid();
        ferdigSignal.countDown();
      } catch (InterruptedException ex) {} // return;
   }

   void gjorArbeid() { ... }
 }


class TestCountdown { // ...
   void main() throws InterruptedException {
     int antTraader = 5;
     CountDownLatch ferdigSignal = new CountDownLatch(antTraader);

     for (int i = 0; i < antTraader; ++i) {
       new Thread(new Arbeider(ferdigSignall)).start();
     }
       ferdigSignal.await();           // venter paa alle traadene
       System.out.println("Alle er ferdige!);
   }
 }
```

La oss endre forrige oppgave til å ha med countdownlatch

Oppgaver

→ ukesoppgaver

→ eller oblig 5😮