IN1010 Repetisjon
Tråder

Julian Fjeld

julianfj@uio.no

Oversikt

- Tråder
  sleep()
  join()
- Runnable
- Monitor
- Lock
- Condition
- Barrierer
  CountDownLatch

Hvorfor bruker vi tråder?

- De fleste datamaskiner i dag har flere prosessorer eller kjerner
- Kjerner kan jobbe på forskjellige ting samtidig
- Når vi lager programmer med tråder sier vi til datamaskinen at disse arbeidsoppgavene kan gjøres i hvilken rekkefølge som passer den, også samtidig
- Visse arbeidsoppgaver kan med andre ord gjøres mer effektivt

Tråder

- Tenk på tråder som arbeidere
- Effektive, dumme, halvblinde arbeidere. De skal ikke vite hva som skjer rundt seg, men la seg lede av arbeidsleder (monitor)
- En tråd kan få en oppgave å gjøre, en klasse vi skriver som implementerer grensesnittet Runnable
- Mange tråder kan gjøre oppgavene sine samtidig, men jobber de på samme ting må de vente til den er ledig
- Når vi bruker venting med tråder må vi huske å ta høyde for at de kan bli avbrutt (InterruptedException)
- Sleep(x) gjør at en tråd venter i x antall millusekunder
- Om traad er en referanse til en tråd kan vi bruke traad.join() for å vente til traad er ferdig med run-metoden sin

- Når vi oppretter en tråd trenger vi (som oftest) i IN1010 en oppgave (ny instans av en klasse som implementerer Runnable), og en referanse til en monitor

Monitor

- En monitor kan vi tenke på som en arbeidsleder
- Monitoren tar seg av all organisering av tråder og passer på at de ikke snubler i beina på hverandre
- Om to tråder vil aksessere samme data er det monitor som passer på at de havner i køer (lock() og unlock())
- En monitor har som regel en beholder, en lås, og eventuelle conditions til den låsen
- Den immplementerer også metoder som gjør at tråder kan interagere med beholderen, men passer på at bare én tråd kan gjøre det av gangen (lock() og unlock())

Lock

- Grensesnitt med 6 metoder, vi bruker bare 2, nemlig lock() og unlock()
- Vi bruker klassen ReentrantLock som implementerer Lock
- Når en tråd prøver å låse en lås som er låst, havner den i en kø og venter på at låsen låses opp før den fortsetter
- Denne køen er ikke synlig for oss som bruker låsen, men skjer i bakgrunnen og kan virke litt som magi når man ikke vet hvordan det fungerer

Condition

- Condition kan vi bruke om vi vil at tråder skal vente på en lås ved andre tilfeller enn at den bare er låst
- Disse tilfellene definerer vi selv, og må vi også passe på å opprettholde selv
- Om vi for eksempel venter på at en liste har mer enn 1 element i seg må vi passe på å gi signal til alle som venter ved hvert tilfelle det kan ha blitt satt inn et element
- En Condition opprettes via en lås med laas.newCondition() og hører til den låsen

Barrierer

- Barrierer er deler av koden der vi synkroniserer tråder
- Det betyr at én tråd, for eksempel tråden som kjører main(), venter på andre tråder
- Ofte kan man bruke join til dette (om man har en referanse til trådene)
- CountDownLatch er også en mulig løsning