INF1010 — Repetisjonskurs i tråder

Eivind Storm Aarnæs
eivinsaa@student.matnat.uio.no

19. mai og 21. mai 2014

"Hello, World!"

- La oss starte med det vanlige eksempelet:
- "Hello, World" – Alle trådene skriver ut samtidig.

"Hello, World!"

- La oss starte med det vanlige eksempelet:
- "Hello, World" – Alle trådene skriver ut samtidig.
- Ta med tilfeldig venting.

Kort om tråder

- En tråd er en parallell beregning inne i en prosess.

Kort om tråder

- En tråd er en parallell beregning inne i en prosess.
- Tråder deler minne.

`wait`, `notify()` og `notifyAll()`

- Vi kan la tråder vente og vi kan vekke dem igjen:

wait, notify() og notifyAll()

- Vi kan la tråder vente og vi kan vekke dem igjen:
  - `wait()` – Lar en tråd sove til den blir vekket.

`wait`, `notify()` og `notifyAll()`

- Vi kan la tråder vente og vi kan vekke dem igjen:
  - `wait()` – Lar en tråd sove til den blir vekket.
  - `notify()` – Vekker en tilfeldig tråd.

`wait`, `notify()` og `notifyAll()`

- Vi kan la tråder vente og vi kan vekke dem igjen:
  - `wait()` – Lar en tråd sove til den blir vekket.
  - `notify()` – Vekker en tilfeldig tråd.
  - `notifyAll()` – Vekker alle trådene.

Hva er & hvorfor synkronisering

- Begrense tilgang til visse biter av koden.

Hva er & hvorfor synkronisering

- Begrense tilgang til visse biter av koden.
- Pga. delt minne kan tråder ødelegge for hverandre.

Synkronisering i Java

- To måter,

Synkronisering i Java

- To måter,
- synkroniserte metoder,
  - eks.: `public synchronized void foo() { /* ... */ }`

Synkronisering i Java

- To måter,
- synkroniserte metoder,
  - eks.: `public synchronized void foo() { /* ... */ }`
- og sykroniserte blokker,
  - eks.: `synchronized (someObject) { /* ... */ }`

Eksempel: Producer/consumer

- Vanlig modell i parallellprogramering:

Eksempel: Producer/consumer

- Vanlig modell i parallellprogramering:
  - Et antall Produsenter produserer data,

Eksempel: Producer/consumer

- Vanlig modell i parallellprogramering:
  - Et antall Produsenter produserer data,
  - et antall Konsumenter bruker data når tilgjengelig.

Eksempel: Producer/consumer

- Vanlig modell i parallellprogramering:
  - Et antall Produsenter produserer data,
  - et antall Konsumenter bruker data når tilgjengelig.
  - En monitor sørger for riktig kommunikasjon mellom dem.

Låser og barierer

- Tillater mer avansert synkronisering:

Låser og barierer

- Tillater mer avansert synkronisering:
  - `ReentrantLock` og `Condition` – lar det deg gjøre det samme som `wait`/`notifyAll` + mer.

Låser og barierer

- Tillater mer avansert synkronisering:
  - `ReentrantLock` og `Condition` – lar det deg gjøre det samme som `wait`/`notifyAll` + mer.
  - `CountDownLatch` – låser til `countDown()` er kallet riktig antall ganger. Hver instans kan bare brukes én gang.

Låser og barierer

- Tillater mer avansert synkronisering:
  - `ReentrantLock` og `Condition` – lar det deg gjøre det samme som `wait`/`notifyAll` + mer.
  - `CountDownLatch` – låser til `countDown()` er kallet riktig antall ganger. Hver instans kan bare brukes én gang.
  - `CyclicBarrier` – låser til riktig antall tråder venter. Tillater også en ekstra handling når riktig antall tråder venter. Hver instans kan gjenbrukes.

Eksempel: Producer/Consumer med låser

Skriv monitoren det forrige eksempelet med `ReentrantLock` og `Condition` istedet for `wait`/`notifyAll`.