IN1010 uke 11

Gruppe 2

Agenda

- Intro til tråder
  - Parallellitet
  - Hva er tråder?
  - Synkronisering
    - → Låser
    - → Monitor
- ☕ Pause 13:00-13:15

- Jobbe med oppgaver/trix/oblig!

Hvorfor parallellisere?

Eksempel: Lage pizzaer

Alternativ 1

1 person lager alle pizzaene, én etter én

Det tar 5 min å lage én pizza

→ 50 min å lage 10 pizzaer



Eksempel: Lage pizzaer

Alternativ 2

Flere personer hjelper til med å lage pizzaene

Det tar 5 min å lage én pizza for én person

→ ~~50~~ 10 min å lage 10 pizzaer med 5 personer

eller

→ ~~50~~ 5 min å lage 10 pizzaer med 10 personer



Parallellisering

Altså:

- Vi kan fordele pizzabakeren sitt arbeid til flere bakere som lager pizza samtidig → spare mye tid
  - Denne forskjellen i tid blir fort veldig stor hvis vi skal bake maaaaange pizzaer

- Dette kalles parallellisering

Sekvensiell vs parallell

Sekvensiell

- Ting gjøres i rekkefølge (etter hverandre)



Parallell

- Ting gjøres samtidig



Parallellisering og tråder

- *Tråder*: Kan tenkes på som pizzabakerne

- Hittil har vi bare skrevet program som kjøres sekvensielt, av *main-tråden* (hovedtråden)
  - Som når 1 pizzabaker lagde 10 pizzaer

- Vil dele opp jobben → lage flere tråder som deler på oppgaven

Tråder i Java: Eksempel

- Vi trenger flere bakere for å lage pizzaer samtidig. Altså flere “tråder”

- Oppgave til hver person (baker): Lag en pizza
  - Oppgavene trådene gjør er objekter av grensesnittet Runnable som inneholder 1 metode: *run()*

    → Skriver oppgaven til enhver tråd inni run()-metoden

Tråder i Java: Eksempel

```java
class PizzaBaker implements Runnable {
    @Override
    public void run() {
        // Oppgaven
        for (int i = 1; i <= 5; i++) {
            System.out.println("- Minutt " + i + " av baking...");
        }
        System.out.println(x:"Bakeren bakte en pizza.");
    }
}
```

Tråder i Java: Eksempel

- Nå som vi har skrevet kode for oppgaven, kan vi lage tråd-objekter
- Lages ved hjelp av klassen *Thread*
- Vi kaller på *start()* på tråden for å starte den!

```
class PizzeriaParallell {
    Run | Debug
    public static void main(String[] args) {
        // Starter tråder
        for (int i = 0; i < 5; i++) {
            Thread nyTraad = new Thread(new PizzaBaker());
            nyTraad.start();
        }
    }
}
```

oppgaven

tråden

Tråder i Java: Eksempel

```java
class PizzeriaParallell {
    Run | Debug
    public static void main(String[] args) {
        // Starter tråder
        for (int i = 0; i < 5; i++) {
            Thread nyTraad = new Thread(new PizzaBaker());
            nyTraad.start();
        }
```

```java
class PizzaBaker implements Runnable {
    @Override
    public void run() {
        // Oppgaven
        for (int i = 1; i <= 5; i++) {
            System.out.println("- Minutt " + i + " av baking...");
        }
        System.out.println(x:"Bakeren bakte en pizza.");
    }
}
```

Tenk at koden i *run()* tar 5 min å kjøre.
1. Hvor lang tid tar hovedprogrammet?

2. Hvor mange pizzaer blir bakt?

Tråder i Java: Eksempel

Svar:
1) Ca. 5 min (kanskje litt mer for å starte opp trådene)

2) 5 pizzaer

```java
class PizzeriaParallell {
    Run | Debug
    public static void main(String[] args) {
        // Starter tråder
        for (int i = 0; i < 5; i++) {
            Thread nyTraad = new Thread(new PizzaBaker());
            nyTraad.start();
        }
```

```java
class PizzaBaker implements Runnable {
    @Override
    public void run() {
        // Oppgaven
        for (int i = 1; i <= 5; i++) {
            System.out.println("- Minutt " + i + " av baking...");
        }
        System.out.println(x:"Bakeren bakte en pizza.");
    }
}
```

Tenk at koden i *run()* tar 5 min å kjøre.
1. Hvor lang tid tar hovedprogrammet?

2. Hvor mange pizzaer blir bakt?

Tråder i Java: Eksempel

```java
class PizzeriaParallell {
    Run | Debug
    public static void main(String[] args) {
        // Starter tråder
        for (int i = 0; i < 5; i++) {
            Thread nyTraad = new Thread(new PizzaBaker(i));
            nyTraad.start();
        }
```

```java
class PizzaBaker implements Runnable {
    int id;
    public PizzaBaker(int id) {this.id = id;}
    @Override
    public void run() {
        // Oppgaven
        for (int i = 1; i <= 5; i++) {
            System.out.println("- Baker nr. " + id + ": Minutt "+ i + " av baking...");
        }
        System.out.println("Baker nr. " + id + " bakte en pizza.");
    }
}
```

Hver baker har fått en *id*. I hvilken rekkefølge blir de ferdige med å bake en pizza?

Tråder i Java: Eksempel

Hver baker har fått en *id*. I hvilken rekkefølge blir de ferdige med å bake en pizza?

<u>Svar:</u>
Rekkefølgen kan bli annerledes hver gang vi kjører (kjører i parallell)

```
lass PizzeriaParallell {
    Run | Debug
    public static void main(String[] args) {
        // Starter tråder
        for (int i = 0; i < 5; i++) {
            Thread nyTraad = new Thread(new PizzaBaker(i));
            nyTraad.start();
        }
```

```
izzaBaker implements Runnable {
    int id;
    public PizzaBaker(int id) {this.id = id;}
    @Override
    public void run() {
        // Oppgaven
        for (int i = 1; i <= 5; i++) {
            System.out.println("- Baker nr. " + id + ": Minutt "+ i + " av baking...");
        }
        System.out.println("Baker nr. " + id + " bakte en pizza.");
    }
}
```

Tråder i Java: Eksempel

Svar:
Rekkefølgen kan bli annerledes hver gang vi kjører (kjører i parallell)

Kjøring 1

```
- Baker nr. 1: Minutt 3 av baking...
- Baker nr. 1: Minutt 4 av baking...
- Baker nr. 1: Minutt 5 av baking...
- Baker nr. 2: Minutt 1 av baking...
- Baker nr. 2: Minutt 2 av baking...
- Baker nr. 2: Minutt 3 av baking...
- Baker nr. 2: Minutt 4 av baking...
- Baker nr. 2: Minutt 5 av baking...
- Baker nr. 3: Minutt 1 av baking...
- Baker nr. 3: Minutt 2 av baking...
- Baker nr. 3: Minutt 3 av baking...
- Baker nr. 3: Minutt 4 av baking...
- Baker nr. 3: Minutt 5 av baking...
- Baker nr. 0: Minutt 3 av baking...
- Baker nr. 0: Minutt 4 av baking...
- Baker nr. 0: Minutt 5 av baking...
Baker nr. 4 bakte en pizza.
Baker nr. 2 bakte en pizza.
Baker nr. 1 bakte en pizza.
Baker nr. 0 bakte en pizza.
Baker nr. 3 bakte en pizza.
```

Kjøring 2

```
- Baker nr. 1: Minutt 4 av baking...
- Baker nr. 1: Minutt 5 av baking...
- Baker nr. 0: Minutt 1 av baking...
- Baker nr. 0: Minutt 2 av baking...
- Baker nr. 0: Minutt 3 av baking...
- Baker nr. 0: Minutt 4 av baking...
- Baker nr. 0: Minutt 5 av baking...
- Baker nr. 2: Minutt 5 av baking...
- Baker nr. 4: Minutt 2 av baking...
- Baker nr. 4: Minutt 3 av baking...
- Baker nr. 4: Minutt 4 av baking...
- Baker nr. 4: Minutt 5 av baking...
- Baker nr. 3: Minutt 2 av baking...
- Baker nr. 3: Minutt 3 av baking...
- Baker nr. 3: Minutt 4 av baking...
- Baker nr. 3: Minutt 5 av baking...
Baker nr. 4 bakte en pizza.
Baker nr. 1 bakte en pizza.
Baker nr. 2 bakte en pizza.
Baker nr. 3 bakte en pizza.
Baker nr. 0 bakte en pizza.
```

Synkronisering - dele data mellom tråder



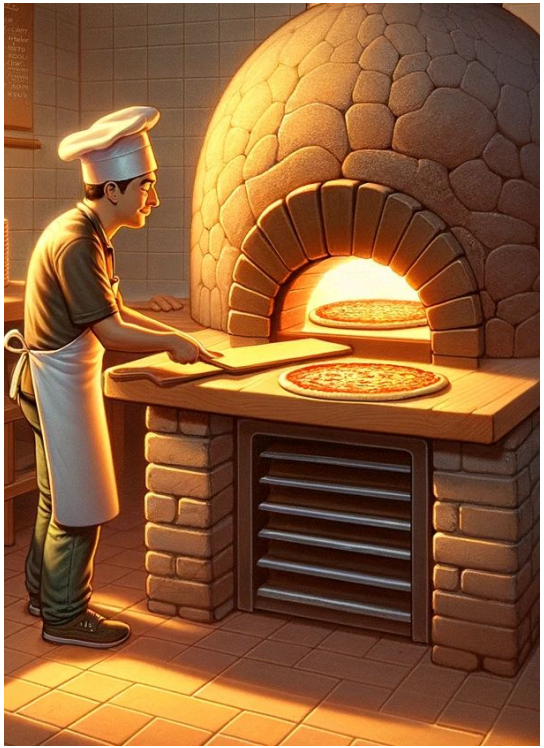
Eksempel uten synkronisering:

- En ovn har bare plass til 1 pizza
- To bakere ser bort på ovnen samtidig og ser at den er tom

  → Setter inn sin pizza så rask som mulig

  → Den ene pizzaen kommer oppå den andre, som blir ødelagt

Synkronisering - dele data mellom tråder



Eksempel med synkronisering:

- To bakere ser bort på ovnen samtidig og ser at den er tom

  → Baker 1 setter inn sin pizza så rask som mulig

  → Ovnen stenger seg automatisk, baker 2 må vente på at pizzaen skal bli ferdig før de putter sin egen. Hvis flere bakere kommer, står de i kø.

Synkronisering - dele data mellom tråder

I koden:

- Viktig å tenke på når man skriver koder med tråder!
- F.eks. hvis to tråder prøver å endre på et tall/array samtidig

Når/på hvilken måte tror du det kan gå galt for tråder å endre på samme tall eller array samtidig? Eksempler?

Synkronisering - dele data mellom tråder

I koden:

- Viktig å tenke på når man skriver koder med tråder!
- F.eks. hvis to tråder prøver å endre på et tall/array samtidig

Når/på hvilken måte tror du det kan gå galt for tråder å endre på samme tall eller array samtidig? Eksempler?

Eksempel på feil (hvis ikke synkronisering):

Vi vil at trådene skal øke *tall* med 1.

int tall = 2

Tråd 1: Leser tall = 2, øker med 1
Tråd 2: Leser tall = 2, øker med 1
Tråd 1: Setter tilbake tall = 3
Tråd 2: Setter tilbake tall = 3
(Tall burde bli 4 nå men er bare 3)

Synkronisering - låser

- Kan løse problemet gjennom å bruke en *lås*
  - = sørger for at kun 1 tråd av gangen kan aksessere et delt objekt
    - Andre tråder må vente på sin tur

- Bruker *ReentrantLock* i Java for dette, med metodene *lock()* og *unlock()*

Tråder og monitor

Kritiske regioner

- Som vi så kan det oppstå problemer når tråder endrer på felles data
  → disse områdene = kritiske regioner

→ Må beskytte kritiske regioner med lås
  → bare en tråd av gangen kan aksessere

Monitor

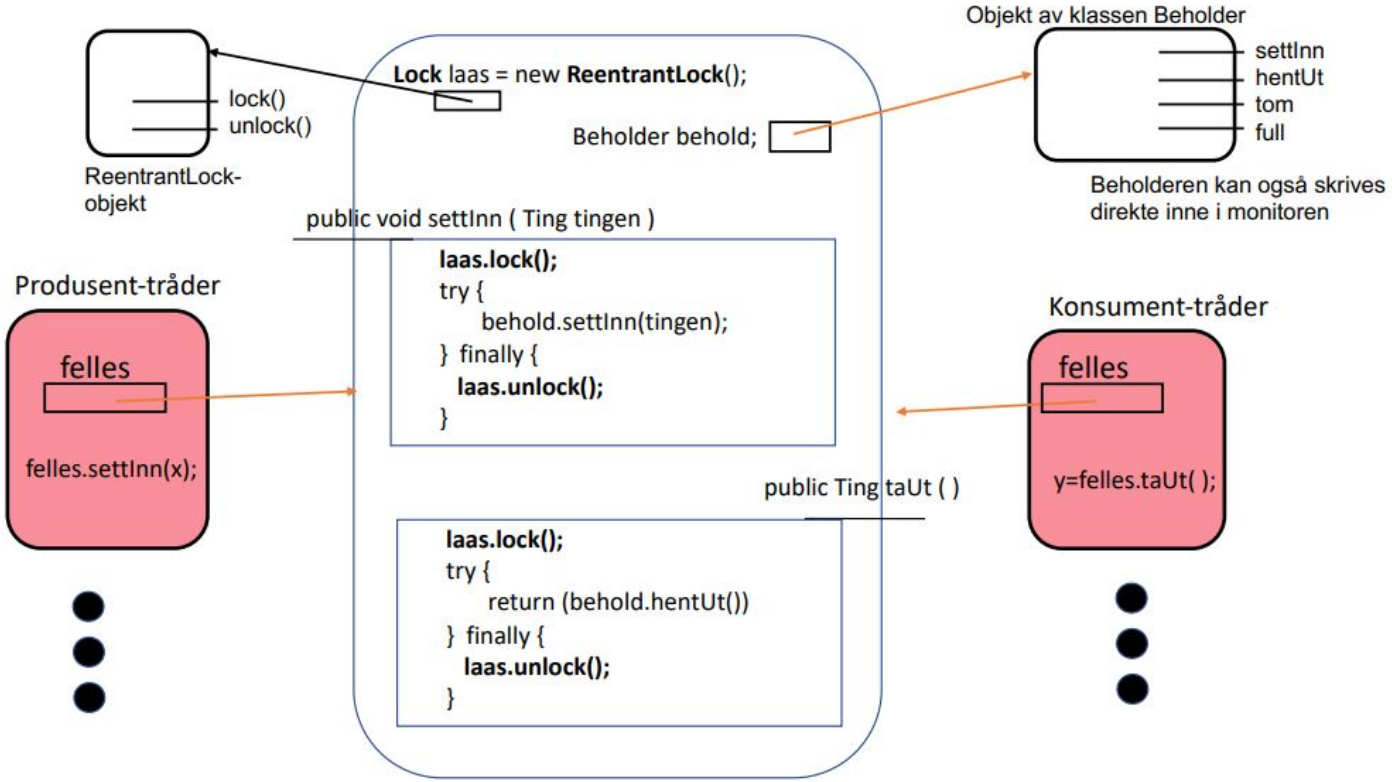
- Lager Monitor-klasse for å beskytte den delte dataen
- Har en klasse som vil bruke med tråder → lager en Monitor-klasse “rundt” den

Altså:

- Monitor er en beholder for den delte dataen (kritiske regioner)
- All låsing og opplåsing skjer her

→ Trådene kaller på metoder i monitoren for å endre på dataen på en sikker måte

Eksempel


lock()
unlock()
ReentrantLock-objekt
Lock laas = new ReentrantLock();
Beholder behold;
Objekt av klassen Beholder
settInn
hentUt
tom
full
Beholderen kan også skrives direkte inne i monitoren
public void settInn (Ting tingen)
laas.lock();
try {
behold.settInn(tingen);
} finally {
laas.unlock();
}
Produsent-tråder
felles
felles.settInn(x);
Konsument-tråder
felles
y=felles.taUt( );
public Ting taUt ( )
laas.lock();
try {
return (behold.hentUt())
} finally {
laas.unlock();
}

Jobb med oppgaver!

IN1010 Emnesiden → Grupper →
Gruppe 2 → Uke11

Husk å ta pause ☕