Plenum: Tråder 1

Ahmed Waseem Saeed

April 2024

Introduksjon

Velkommen til ny plenumstime! Denne uken markerer den første av to uker der tråder er i fokus. Denne uken skal vi innledningsvis vise et enkelt eksempel med tråder, før vi går over til et mer sammensatt eksempel. Her skal vi vise hvordan ressurser og data kan deles mellom flere tråder på en ryddig og korrekt måte, ved hjelp av monitorer og Conditions. Merk at tråder tradisjonelt har vært et tema mange sliter litt med i starten, derfor anbefaler vi å sette av god tid til å bearbeide stoffet slik at det sitter.

Oppvarming: Prosess-tråder

Lag en klasse Prosess som kun tar inn en prosess-id (pId). Klassen implementerer Runnable og run-metoden skal kun skrive ut informasjon om prosessen, og at den kjører. Deretter lager du et hovedprogram som oppretter ti slike tråder (i en løkke) og kjører dem. Skriv til slutt ut i hovedprogrammet "Nå er programmet ferdig" for å tydeliggjøre at alle trådene har kjørt.

Hva skjer når du kjører hovedprogrammet? Skriver alle trådene ut informasjonen? For å fikse dette skal vi benytte .join() for å håndtere at alle trådene er ferdig før vi avslutter programmet. Modifiser hovedprogrammet slik at du kaller .join() på hver enkelt tråd før du avslutter hovedprogrammet. Dette krever potensielt at du må bruke en beholder for å samle trådene.

Merk at det å sørge for at alle trådene er ferdig er ofte en utfordring ved parallelle programmer. Tenk nøye gjennom når trådene skal være ferdig, når du programmerer slike programmer.

Bakeri, bakere og kunder

Vi skal nå modellere et bakeri. Denne verden skal bestå av bakere (produsenter) og kunder (konsumenter) som skal dele ressurser (bakverk) i et bakeri (en monitor).

Bakeri

Klassen Bakeri opptrer som vår monitor. Det vil si at både kunder og bakere har en referanse til et Bakeri-objekt, og at dataen her blir en delt ressurs. Vi trenger derfor en lås i klassen. I tillegg trenger klassen en oversikt over antall bakverk som er klare for å bli spist. Vi trenger også en Condition ikkeTomt, som vi skal bruke for å signalisere ventende kunder med.

Implementer følgende metoder:

```
/*
 * Metoden skal produsere bakvarer ved å øke antall
 * bakvarer, samt signalisere til ventende kunder.
 * I tillegg skal metoden printe ut at bakeren
 * (med id) har laget en bakvare.
*/
public void lagBakvare(int bakerId);


/*
 * Metoden skal la en kunde spise en bakvare.
 * i denne konteksten betyr å printe "Spise"
 * ut informasjon. Husk å redusere antall bakverk og
 * vente hvis det ikke er noen bakverk tilgjengelig.
*/
public void spisBakvare(String kunde);
```

Baker

Klassen Baker representerer produsentene våre. Den trenger en referanse til monitoren, samt en id (heltall) for hver baker. Hver baker skal sette inn 50 bakvarer hver. For hver bakvare en baker setter inn, skal tråden sove 50 millisekunder.

Kunde

Klassen Kunde representerer konsumentene våre og skal ha et navn, samt referanse til monitoren. Hver kunde spiser fire bakvarer. La også hver kunde sove i 50 millisekunder etter en bakvare er konsumert.

Testprogram

Skriv et testprogram som oppretter fire bakere og fire kunder. Kjør koden og sørg for at programmet terminerer.

Hva skjer hvis du øker antall bakvarer som spises av hver kunde til 20, og reduserer antall bakere til 1? Kjør koden. ***Vranglås.*** Dette må vi håndtere!

Håndtere vranglåsen

En taktikk for å håndtere vranglåser er å identifisere når produsentene er ferdig med å produsere, og deretter varsle alle ventende konsumenter at vi kan avslutte. Derfor skal vi utvide monitoren vår med en metode, *ferdigBakt()*, som skal bruke Condition-en til å varsle alle ventende kunder om at det ikke finnes flere bakvarer. Hvilke andre endringer må du foreta i koden for at dette skal fungere tilstrekkelig?

Hint: en boolean variabel som indikerer om bakerne er ferdig bakt for i dag.

Du må også endre testprogrammet nå, slik at du oppdager når bakerne er ferdig.

Hvis tid

Akkurat nå forsøker Kunde-trådene å spise bakvarer selv om bakeriet er ferdig bakt for dagen (helt til de har telt ned i løkka). Fiks dette slik at dersom bakeriet er ferdig, så skal Kunde-trådene slutte å kjøre. Her står du fritt frem til hvordan du vil løse dette. En tilnærming kan være å endre retur-verdien til spisBakvare(), som kan brukes til å indikere at det ikke er noe mer å spise.