Seminaroppgaver uke 9 og 10

Oppgave 1: Vi skal lage et program med kaffedrikkere (tråd), en barista (tråd), og ett bord (monitor). Bordet kan ha uendelig mange kaffer på seg.

A. Lag klassen Barista som er en tråd. Alle har et array med ulike kaffedrikker

   private final String[] drikker = {"Americano", "Café au lait", "Caffè latte", "Caffè mocca", "Espresso", "Cortado"};

   I tillegg tar Baristaen inn et bord (Som du skal lage i oppgave E) som vedkommende kan servere kaffen på, og en id.

B. Implementere metoden run til barista. Hver barista lager 10 kaffer, og disse drikkene skal velges tilfeldig fra liste drikker (TIPS: bruk java.util.Random for å få et tall mellom 0 og lengden til drikker). Baristaen skal sa skrive ut sin id og hvilken drikke som blir laget. Før vedkommende servere kaffen på bordet. Når 10 kaffer er servert skal baristaen sende inn at det er "tomt".
C. Lag klassen Kaffedrikker som er en tråd. Kaffedrikker skal ta inn et bord vedkommende kan hente kaffe fra, og en id.
D. Implementer metode run i Kaffedrikke. Denne skal ha en teller som teller antall kaffer som kaffedrikkeren får drukket. Så lenge kaffedrikeren får beskjed om at det ikke er tomt skal vedkommende printe sin id og hvilken kaffe som ble drukket. Når det ikke er flere kaffer for kaffedrikkeren å drikke skal vedkommende printe sin id og hvor mange kaffekopper som ble drukket.
E. Lag klassen Bord. Lag en metode som serverer kaffe (legger de til i bordet), husk at det kan være uendelig mange kaffer på bordet om gang. Lag også en metode hentKaffe som henter en kaffe fra bordet så lenge det er en kaffe der, og signaliserer til Kaffedrikker når det er tomt.
F. Lag klasseen Hovedprogram som lager et Bord, 2 baristaer og 10 kaffedrikkere. Test gjerne med ulike verdier for å se hvordan programmet oppfører seg)

Oppgave 2: Vi tenker oss en brusautomat (vår monitor) som inneholder et gitt antall brusbokser. Videre har vi kunder (tråder) som forsøker å ta brusbokser ut av automaten. Vi har også én person (tråd) som er ansvarlig for å fylle opp automaten når den er tom. PS: husk at du må bruke locks der det passer seg (kun en tråd kan endre på en verdi om gangen).

A. Brusautomaten vår har en metode refill (som den som er ansvarlig for å fylle opp brusautomaten bruker). Metoden skal fylle opp brusatomaten til dens kapasitet. Men dette kan kun gjøres når den er tom. Derfor må denne metoden vente (tips Condition) til det er tomt i brusautomaten. Når brusmaskinen blir fylt opp skal det printes en liten beskjed som gir sier nettopp dette
B. Brusautomat skal ha metoden kjopBrus, denne metoden skal la de som kaller på den ta seg en brus (dette gjøres ved å senke antall brus som er i brusmaskinen). Så lenge det ikke er noen brus i maskinen må man stå å vente (tips Condition). Når man får kjøpt en brus skal det printes ut en beskjed om nettopp dette.
C. Lag klassen MaskinFyller, denne personen prøver å fylle opp maskinen hvert 5 sekund

D. Lag klassen BrusDrikker, hver brus drikker venter mellom 0-3 sekunder før den tar en ny brus (Dette kan gjøres ved bruk av random)). BrusDrikker skal også ta inn et random tall mellom 5-15 som er antall brus de har planer om å drikke

Barrierer

En barriere tenkes på som et samlingspunkt i koden, der alle stopper og venter på at alle andre også har kommet frem. Deretter kan vi la alle trådene fortsette, *samtidig*.

- 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.

Oppgave 3:

a) Når er det fordelaktig å benytte CountDownLatch? Og når er det fordelaktig å benytte CyclicBarrier? Diskuter forskjellen mellom de 2.

b) Du ønsker å skrive ut noe på skjermen hver 5. gang en oppgave er utført, uavhengig av hvilke tråder som har gjort oppgaven. Hva slags barriere ville du benyttet her?

Oppgave 4:

1. Lag et program som starter opp 8 tråder. Alle trådene skal si hei 2 ganger, men ingen av trådene har lov til å si hei den andre gangen før alle trådene er ferdig med å si hei den første gangen.
2. Utvid deloppgave 1, nå skal trådene skrive ut hei 3 ganger. En tråd kan ikke skrive ut hei den tredje gangen, før alle trådene er ferdig med å skrive hei ut den andre gangen.

Oppgave 5:

Endre oppgave 2 fra forrige uke, slik at personen som fyller opp brusautomaten blir ferdig når alle kundene er ferdig.

Oppgave 6:

Skriv et program som starter noen tråder (akkurat hvor mange bestemmer du selv). Du skal lage en klasse Deltaker(Tråder), og de konkurrerer i å sende det største tallet til monitoren (Tallet de sender skal genereres tilfeldig, og skal være positive heltall). Den tråden som har det største tallet skal slutt skrive ut at den har vunnet konkurransen, men for at en tråd skal vite om den har skrevet ut det høyeste tallet må den vente til alle trådene er ferdig. Monitoren trenger to metoder, en for for å ta imot et nytt tall ,og en for å returnere det. Husk at kun en tråd kan endre på data om gangen.