Uke 9

Parallelitet - Hva er det?

- Kjøre to eller flere oppgaver samtidig
  - Motsatt fra å ha kun en main tråd som kjører sekvensielt
  - Naturlig vs ekte
- Ekte parallell
- Naturlig parallell

Tråder

- Tråder kan:
  - Vente
  - Sove
  - Avbrytes
  - Dele felles data med andre tråder
  - Starte andre tråder
- Implementerer Runnable

Eksempel:

Eksempel med Runnable

```
interface Runnable {
    void run();
}

class MyTask implements Runnable {
    /* Say hello, but think for a while between each time. */
    public void run() {
        for (int i = 1; i < 5000; i++)
            if (i % 1000 == 0)
                System.out.printf("Hello for the %d-th time!\n", i / 1000);
    }
}
```

Eksempel med Thread og start fra main-tråden

```
public Main {
    public static void main(String[] args) {
        Runnable task = new MyTask();
        Thread thread1 = new Thread(task);
        Thread thread2 = new Thread(task);
        thread1.start(); thread2.start(); // Will make a call to task.run()
    }
}
```

Eksempeloutput (ikke alltid samme utskrift, siden vi har med tråder å gjøre)

```
Hello for the 1-th time!
Hello for the 1-th time!
Hello for the 2-th time!
Hello for the 3-th time!
Hello for the 2-th time!
Hello for the 4-th time!
Hello for the 3-th time!
Hello for the 4-th time!
```

Låser og monitor

- Låser sikrer at det er kun en tråd om gangen som bruker en kritisk felles ressurs
  - Unngår race-condition (oppdatere samme verdi samtidig)

- Conditions, await() og signalAll() brukes for passiv venting

Oppgave 1

Du skal lage et program med kaffedrikkere (tråder), baristaer (tråder) og et bord (monitor). Bordet kan ha uendelig mange kaffekopper plassert på seg; det vil si at bordet aldri blir fullt.

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) vedkommende kan servere kaffen på, og en id.

b) Implementér metoden run i Barista. Hver barista lager ti kaffe, og disse drikkene skal velges tilfeldig fra en liste med drikker (Tips: bruk java.util.Random for å få et tall mellom 0 og lengden til drikker). Baristaen skal skrive ut sin id og hvilken drikke som blir laget før vedkommende serverer kaffen på bordet. Når ti kaffe er servert, skal baristaen melde i fra at vedkommende er ferdig (inkludert id-en til baristaen).

c) Lag klassen Kaffedrikker, som er en tråd. Kaffedrikker tar inn et bord vedkommende kan hente kaffe fra, og en id.

d) Implementer metode run i Kaffedrikker. 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 gangen. 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 klassen Hovedprogram som lager et bord, to baristaer og ti 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. Husk at du må bruke låser der det passer seg (kun én tråd kan endre på en verdi om gangen).

a) Brusautomaten har en metode refill (som den som er ansvarlig for å fylle opp brusautomaten bruker). Metoden skal fylle opp brusautomaten 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 beskjed om nettopp dette.

b) Brusautomat skal ha metoden kjopBrus. Den 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 og vente (tips: Condition). Når man får kjøpt en brus skal det printes ut en beskjed om nettopp dette.

c) Lag klassen Maskinfyller. Personen prøver å fylle opp maskinen hvert 5 sekund.

d) Lag klassen Brusdrikker. Hver brusdrikker venter mellom 0 og 3 sekunder før den tar en ny brus (dette kan gjøres ved bruk av random). Brusdrikker skal også ta inn et tilfeldig tall mellom 5 og 15, som er det antall brus drikkeren har planer om å drikke.

e) Lag et hovedprogram der du tester ut brusautomat-systemet.

Oppgave 3

Oppgaven din er å implementere et sykehus. Et angitt antall pasienter vil mottas og behandles på sykehuset. Mottak og behandling skal skje parallelt.

Sykehuset skal ivareta følgende oppgaver:

- Motta pasienter og plassere dem i en kø.
- Behandle pasienter, én av gangen, i rekkefølgen de kommer inn.
- Stenge når det ikke er flere pasienter igjen å behandle (programmet terminerer).

Her er en oversikt over spesifikasjoner for klassene og metodene:

- *Sykehus:* Holder rede på pasientkøen, mottak og behandling av pasientene.
- *MottaPasienter:* Legger til pasienter i sykehusets kø.
- *BehandlePasienter:* Behandler pasientene i køen. Hver behandling tar to sekunder.
- *leggTilPasient(String pasientId):* Legger en pasient (pasientId) til i køen og signaliserer tilstandsendring.
- *behandlePasient():* Sjekker om det er pasienter til stede og behandler dem. Hvis det ikke er pasienter til stede, må det ventes inntil videre.
- *signaliserAllePasienterLagtTil():* Denne metoden blir kalt når alle pasientene har blitt lagt til i køen og det ikke er forventet flere.

Det er viktig at det skrives ut beskjeder i terminalen når en gitt pasient mottas eller behandles. Test at sykehuset fungerer ved å implementere et hovedprogram.