

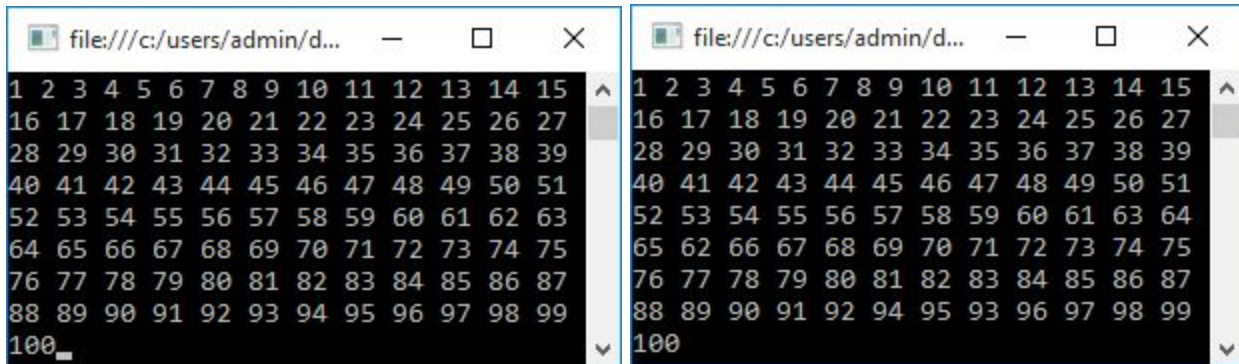
PG3300 Innlevering – Oppgave 2

Race conditions

En “Race condition” er et begrep brukt i sammenheng med multithreading. Betingelsen går ut på at flere tråder prøver å aksessere og endre samme type data på en gang. Et eksempel på hvordan dette kan skje er å ta i utgangspunktet to tråder som har tilgang til en variabel:

- Første tråd leser variabelen
- Neste tråd leser samme variabel
- Begge tråder endrer verdi (de “racer” for å endre verdi sist).
- Siste endring blir bevart, og verdien til en av trådene blir overskrevet.

Slike “race conditions” kan gi uventet (og feil) resultat, noe som kan svekke og endre programmets intensjon. Race conditions kan unngås ved bruk av “Locks”(Forklarer dette i neste oppgave)



```
file:///c:/users/admin/d... 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98 99
100_
```

```
file:///c:/users/admin/d... 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 63 64
65 62 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 94 95 93 96 97 98 99
100
```

Her er et eksempel (koden ligger på neste side) på Race conditions. Konsollvinduet til venstre har lock-statement. Konsollvinduet til høyre har ikke det, og det har derfor oppstått “Race conditions”. Her blir rekkefølgen på tallene feil, i dette tilfellet kommer 63 etter 61, 62 etter 65, 94 etter 92, og 93 etter 95

Locks

“Lock” er en type statement i C# (finnes også i de fleste språk) som forsikrer at kodesnutt inne i en lock blir kjørt og aksessert av bare en tråd helt til låsen blir “unlocked”.

Locks har flere bruksområder, noen eksempler på det er “static” og “local”. Vi bruker en static lock når vi vil bruke den i statiske metoder og låse globale data/cache i alle instanser. En local lock blir derimot brukt lokalt og sikrer tråden for en instansen.

Her er et eksempel på bruken av Locks i et program. Metoden addAmount() inkrementerer x-verdien hver gang den blir kalt på. For å lage en “Lock” rundt denne funksjonen, så lager vi først en variabel av typen “Object”. Deretter bruker vi “lock”-statement med “thisLock” variabelen og omringer kodesnutten vi vil låse.

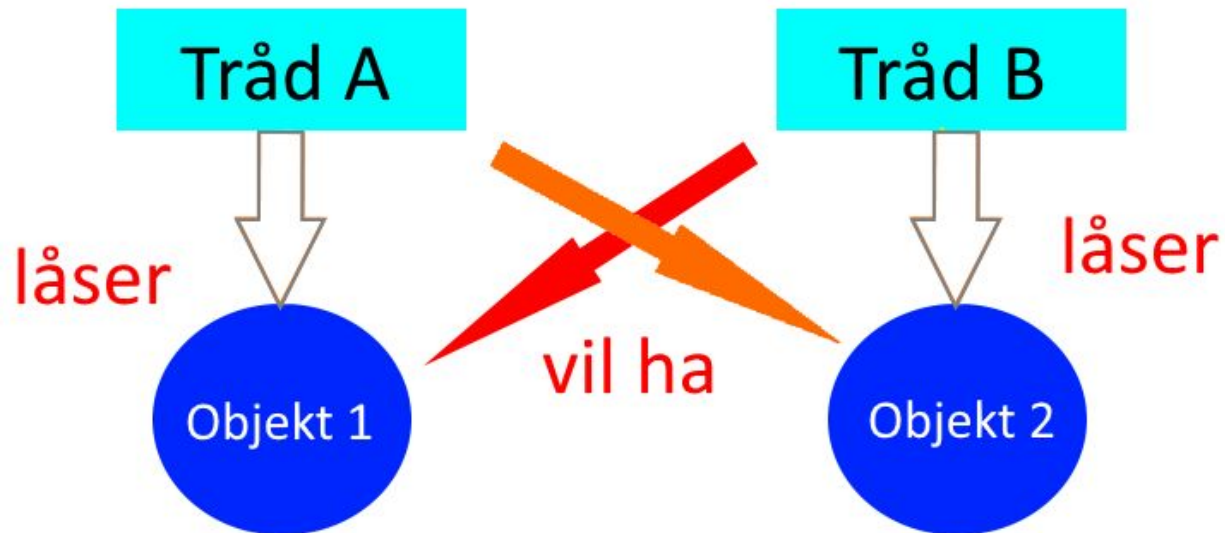
For å teste at “Race conditions” ikke oppstår, så oppretter vi og starter tråder ved å bruke “ThreadStart” og kjøre den i en for-løkke. Vi kjører deretter koden med og uten lock-statement for å se om variabelen blir overskrevet og om lock-statementen fikser problemet.

```
class Program
{
    private Object thisLock = new Object();
    private int x = 0;
    1 reference
    private void addAmount()
    {
        lock (thisLock)
        {
            x++;
            Console.Write(x + " ");
        }
    }
}
0 references
static void Main(string[] args)
{
    Program p = new Program();
    for(int i = 0; i < 100; i++)
    {
        Thread t = new Thread(new ThreadStart(p.addAmount));
        t.Start();
    }
    Console.ReadKey();
}
}
```

Deadlocks

“Deadlocks” er en betegnelse på et problem/situasjon som kan oppstå i programmering. Problemet oppstår når det er to eller flere prosesser med “lock-statement” som prøver å aksessere og bruke samme ressurs. Det skjer ved at en prosess(A) (tar i bruk lock-statement) som bruker en ressurs(X), venter på at ressurs(Y) som blir brukt av en annen prosess(B) (som også tar i bruk lock-statement) skal bli fullført, og samtidig venter på ressurs(X) som blir brukt av prosess(A) . Resultatet blir at begge prosessene i programmet venter på hverandre og ingenting skjer.

Her er en illustrasjon:



I dette eksempelet låser Tråd A objekt 1 og Tråd B låser Objekt 2. Tråd A vil aksessere Objekt 2 og Tråd B vil aksessere Objekt 1. Siden begge objektene er låst til en tråd og begge trådene prøver å få tak i det andre objektet, har det oppstått en Deadlock.