
USN
Universitetet
i Sørøst-Norge

PB1090 – Objekt orientert programmering
Vår 2024

Oppsummering: Forelesninger og lab - uke16
Campus Bakkenteigen

lars.e.opdal@usn.no

CV – Lars Erik Opdal

Akademia:

Master i Informatikk fra UiO 2016

UH-ped course 1 (UHUL – Teaching, learning and assessment in the higher education sector)

2021

UH-ped course 2 (UHUDT – Education in a digital age)

2022

Industri – Offentlig og privat sektor

Konsulent/systemutvikler - Java og C#

USIT, Simula, UiO, sikt.no, Bane Nor, Statens Vegvesen

2016 - 2021

Grønn databehandling - Bevisst ressursbruk i programvareutvikling

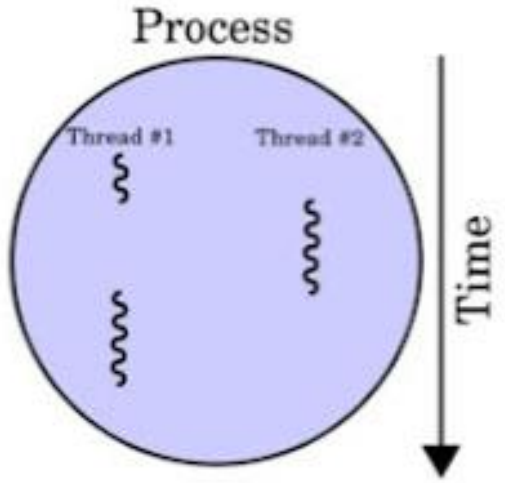
Agenda:

Tema1: Parallell programmering med tråder i Java
- Hvordan utnytte «ubrukt» hardware/CPU/cores
- Krav
- Hensikt – Ytelse
- Validering
- Verktøy – tidtaking

Tema2: Minne lekkasjer – forebygge og hinder dem i Java og OOP generelt
- Bruk av static (keyword) i Java
- Analyse med bruk av GC
- Verktøy – IDE, profiler, bygge-systemet

Tilslutt: Spørsmål & Svar

Grønn databehandling - Bevisst ressursbruk i programvareutvikling



Tema1: Parallell programmering med tråder i Java

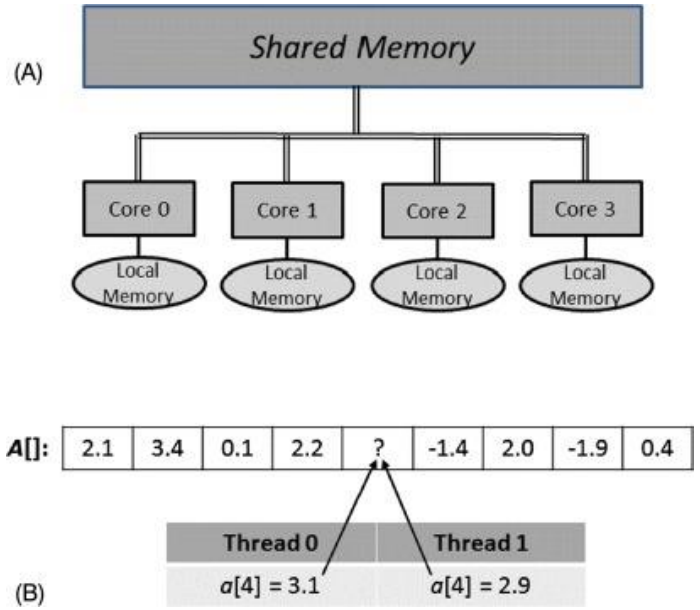
- Operativsystemet administrerer alt - Prosesser (og delvis et antall tråder i hver prosess)
  - Prosesser (P1, P2, ... , Pn)
    - Prosess er utføringen av et program
    - Er isolert fra hverandre, kan i utgangspunktet bare «snakke» direkte til operativsystemet
  - En tråd:
    - Er parallelle eksekveringer inne i én prosess
    - Alle tråder i en prosess deler prosessens minne (ser de samme variable og programkode)

NB. Gjelder OS på desktop med multi-core CPU. Det kan være forskjellig på server OS

Grønn databehandling - Bevisst ressursbruk i programvareutvikling

Tema1: Parallell programmering med tråder i Java

- Motivasjon: Hvordan utnytte «ubrukt» hardware/CPU/cores
  - En vanlig PC/laptop i dag har multi-core CPU med 4/8/16 cores/kjerner og delt minne
  - Delt minne betyr her at flere tråder kan «se» samme data
  - Men data fra tråder må synkronisere!



Grønn databehandling - Bevisst ressursbruk i programvareutvikling

Tema1: Parallell programmering med tråder i Java

- Krav
  - Vi har en algoritme (sekvensiell) for å løse et problem.
  - Den gir riktig eller forventet svar.
  - Det er mulig å måle eksekveringstid nøyaktig.
  - (vanskelig med «spagetti kode», mange funksjonskall eller goTo staments)
  - Algoritme må *refaktoreres* for å kunne kjøres i parallell.
  - Gjøres med Tråder (Thread class og Runnable interface) i Java

- Hensikt – Bedre ytelse

  Dersom krav er oppnådd og implementasjon er «riktig» får vi bedre ytelse ift. eksekveringstid/kjøretid.
  Gevinst: Ubrukt hardware blir benyttet

  Strømforbruk øker ikke når dette gjøres på en multi-core CPU. Bare hvis vi bruker en superComputer/cluster (HPC) dvs. beregningsklynge (tungregning)

Grønn databehandling - Bevisst ressursbruk i programvareutvikling

Tema1: Parallell programmering med tråder i Java

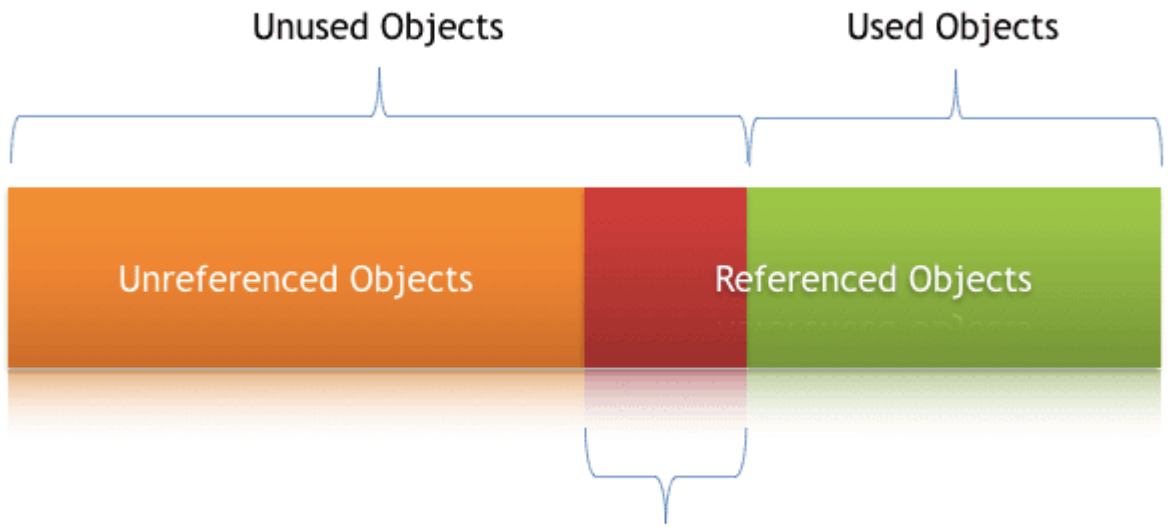
- **Verktøy** – tidtaking
  - Ta tiden på kode blokker (manuelt)
  - Kjøre enhetstester
  - Bruke profiler

- **Validering**

  Metrikk som brukes er **SpeedUp**, definer som $S = \frac{kjøretid\ i\ sekvensiell\ algoritme}{kjøretid\ i\ parallell\ algoritme} = \frac{3000ms}{300ms} = 10$

Grønn databehandling - Bevisst ressursbruk i programvareutvikling

Tema2: Minne lekkasjer – forebygge og hinder dem i Java og OOP generelt



java.lang.OutOfMemoryError.

Grønn databehandling - Bevisst ressursbruk i programvareutvikling

Tema2: Minne lekkasjer – forebygge og hinder dem i Java og OOP generelt

Kjente problemer:

- Konsekvent bruk av ***static*** i Java og C# (OOP)
- Manglende eller en forenklet implementasjon av egne klasser med @override metoder som *equals* & *hashcode*
- Manglende håndtering av ubrukte resurser med close() og try-with-resource(){}
- (data strømmer, filer og tilkoblinger)
- Indre klasse som bruker data fra ytre klasse, blir ikke fjernet med CG

Grønn databehandling - Bevisst ressursbruk i programvareutvikling

Tema2: Minne lekkasjer – forebygge og hinder dem i Java og OOP generelt

Hvordan fikse dem

- Verktøy:
  - IDE: fiks «warnings», bruk tips for refaktoring, code optimaztion (fjerne ubrukte imports), kode formatering, osv.
  - Monitorering og bruk av profiler
  - Bygge-systemer (devOps pipline)
  - Clean Code prinsipper: Refaktoering av kildekode (Spesielt viktig i gamle IT-systemer (legacy) med java7/8 og eldre)
  - Kjøpe mer minne (hardware) eller bruke sette av mer minne for prosessen?

- Analyser Garbage collector (GC)
- Bruke ny type Garbage Collector ZGC
- *Bruke IDE med profiler*

Grønn databehandling - Bevisst ressursbruk i programvareutvikling

Spørsmål & Svar

Ressurser for testing:
https://github.com/leo100584/PB1090