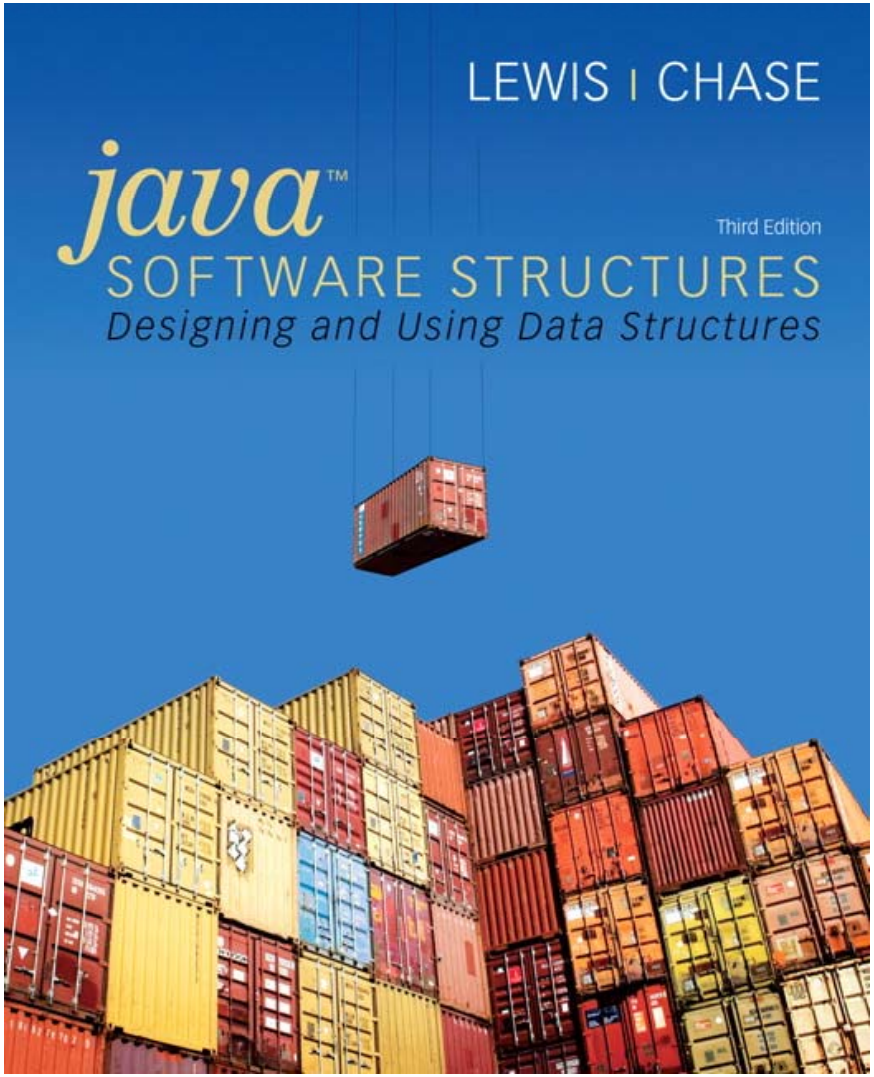
HØGSKOLEN I BERGEN



TOD-063
Datastrukturer og Algoritmer

Forside fra lærebokens Nord-Amerikanske utgave

Tar for seg praktisk problemstilling:

- Hvordan håndtere containere som blir lastet fra containerskip i en travel havn på best mulig vis
- Setter krav til:
  - Effektiv lagringsstruktur
  - Effektive algoritmer for å håndtere
  - Kombinasjonen av de to

… som er noen av de problemstillingene vi skal se på i dette kurset

Timeplanen TOD-063

HØGSKOLEN I BERGEN

	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
08:15-09:55			Data Klasse/Lab A622/Lab		
10:05-11:45	Felles foreles A-715au	Felles foreles A-715au	INF Klasse/Lab A627/Lab		
12:15-13:55				Data Klasse/Lab A622/Lab	INF Klasse/Lab A627/Lab

Lærere: Sven Olai Høyland
Svein-Ivar Lillehaug
Harald Soleim

+ labassistenter

Faget på its Learning

HØGSKOLEN I BERGEN

1, 2 eller 3 sider….

?

Vi forsøker med en felles fagside for de to klassene

(besluttet etter enighet blant studentene)

HØGSKOLEN I BERGEN

Først litt generelt om faget TOD-063

- Fag nr. 2 i programmeringsutdanningen
- Utvikling av mindre programsystem
- Lage og analysere datastrukturer og algoritmer for generelle problem
- Lære å implementere ulike datastrukturer (i Java)
- Lære å analysere effektivitet for algoritmer

HØGSKOLEN I BERGEN

Verktøy for programutviklere

- Maskinutstyr, nett
- Systemprogram (operativsystem, kompilatorer, editor/IDE …)
- Krever kunnskap om (setter krav til dere…), må kjenne til / kunne beherske:
  - Datastrukturer/algoritmer for generelle problem
  - Metoder/teknikker for programutvikling (modularisering, OOP, topp-ned, stegvis forfining, ADT, …)
  - Programtesting/feilfinning

HØGSKOLEN I BERGEN

Presentasjon av høgnivå datastrukturer i boka

- Grensesnitt mot brukere (public-metoder, “interface”)
- Eksempel på bruk
- Implementering i Java (på en eller flere måter)
- Analyse av effektivitet (tidsbruk, plasskrav)
- Ev. sammenligning med tilsvarende ferdige klasser i Java API

HØGSKOLEN I BERGEN

Gruppering av stoffet i boken

…det som kurset skal dekke

- Intro, analyse (kap. 1 og 2)
- Grunnleggende element og teknikker: Mengder, tabell, lenket struktur, rekursjon, søk/sorter (kap. 3, 4, 7, 8)
- Lineære høgnivå-strukturer: Stabel, kø, lister (kap. 3, 5, 6, 11)
- Ikkje-lineære høgnivåstrukturar: Tre, haug, hash-tabell, graf (kap. 9, 10, 11, 12, 13, 14)

- Gir grovt sett pensum: Det meste frem til og med kap. 14
- Læreboken er helt fersk, så endelig pensum bestemmes underveis.

HØGSKOLEN I BERGEN

Bokas kapittel 1: Intro

- Identifisere forskjellige aspekt rundt programvare og kvalitet
- Motivere for bruk/valg av datastrukturer basert på kvalitative vurderinger
- Introduksjon av begrepet datastruktur

  … og introduksjon av flere enkle datastrukturer

HØGSKOLEN I BERGEN

Utvikling av programvare (software development)

- *Software Engineering (Programvareutvikling)* er studiet av de teknikker og teorier som støtter utviklingen av programvare av høy kvalitet
- Har som fokus å kontrollere utviklingsprosessen for til enhver til å oppnå gode resultat
- Vi skal etterstrebe å:
  - tilfredsstille våre klienter – de personer eller det firma/ organisasjon som har bestilt "produktet"
  - møte behovene til brukerne – folket som skal bruke produktet

HØGSKOLEN I BERGEN

Mål med programvareutvikling

- Løse de rette problemene
  - krever ofte litt å finne ut av hva disse er
  - god kommunikasjon mot brukerne er et must
  - husk: som oftest vanskeligere enn hva en først tror
- Levere løsninger i tide og i samsvar med budsjett
  - alltid noen 'trade-offs' å ta stilling til
- Levere høy-kvalitets løsninger
  - må her også ta hensyn til forskjellige *stakeholders* og at alle disse skal komme best mulig ut av det hele (de med interesse i produktet: brukere, oppdragsgiver, investorer, osv…)

Karakteristikker for god kvalitet i SW

HØGSKOLEN I BERGEN

I **bold**: fokus TOD063

Korrekthet

Pålitelighet

Robusthet

Brukervennlighet

Lett å vedlikeholde

Gjenbrukbarhet

Plattformeruavh.

Effektivitet

Quality Characteristic	Description
Correctness	The degree to which software adheres to its specific requirements.
Reliability	The frequency and criticality of software failure.
Robustness	The degree to which erroneous situations are handled gracefully.
Usability	The ease with which users can learn and execute tasks within the software.
Maintainability	The ease with which changes can be made to the software.
Reusability	The ease with which software components can be reused in the development of other software systems.
Portability	The ease with which software components can be used in multiple computer environments.
Efficiency	The degree to which the software fulfills its purpose without wasting resources.

FIGURE 1.1 Aspects of software quality

Husk1: Pålitelig programvare feiler sjelden, og når den feiler minimaliseres følgene av feilen/avbruddet/programkræsjet

Husk2: Kvalitet har forskjellig betydning for forskjellige folk

HØGSKOLEN I BERGEN

Et praktisk eksempel

Problemstilling:

- Containere som losses i en havn
- Kunne kanskje laste disse direkte over på ventende trailere og tog?
  - effektivt for de som jobber på kaien (får ting unna)
  - men ikke spesielt effektivt for befraktningsselskapene og jernbanen som kanskje ikke alltid er klar for å være der (ikke i korrespondanse med deres optimale logistikk)

HØGSKOLEN I BERGEN

eksempel forts.

- Hva vet vi om hver enkelt container?
  - samme størrelse og form
  - hver enkelt har sin unike ID
  - ID gir en del informasjon, blant annet om hvor containeren skal
  - havnearbeiderne har ikke behov for å vite hva som er i hver enkelt container

HØGSKOLEN I BERGEN

eksempel forts..

- Containerne kan kanskje lastes rett av og settes på kaia?
  - Effektivt med tanke på lossing av båt
  - Ikke effektivt med tanke på å finne igjen containerne for å få dem på rette trailere/tog når disse er klare for å hente dem
  - Krever i verste fall et lineært søk over hele kaiområdet når en container søkes for avhenting

HØGSKOLEN I BERGEN

eksempel forts…

- Hva om vi maler et stort tabellsystem på kaien med nummerering og det hele slik at hver lagret container kan indekseres på dens ID?
  - ID er unik for alle lagrede containere
  - 'kaitabellen' ville bli svært STOOOR (sannsynligvis ikke plass for den)
  - kaitabellen ville være stort sett tom
  - en betydelig sløsing med kaiplass!

HØGSKOLEN I BERGEN

eksempel forts....

- Hva om vi mekker om løsningen med kaitabell slik at denne kan ekspandere og trekke seg sammen for å unngå tomme plasser?
  - vil alltid ha rekkefølge på IDer, og ingen tomme plasser
  - enkelt å finne container etter ID
  - MEN, vil tvinge containere til å flyttes flere ganger 'frem og tilbake' ettersom nye containere kommer til og andre fjernes – altså en hel masse overhead i arbeid!

HØGSKOLEN I BERGEN

eksempel forts.....

- På tide å tenke litt på hva vi vet om hver container…
  - ID nummeret gir oss også destinasjonen for hver container
  - Hva om vi lager en kaitabell hvor hver celle i tabellen representerer en ny tabell med plass for containere?

eksempel forts……

- Med vår nye løsning kan vi lagre containerne for hver destinasjon på en rekke forskjellige måter:
  - lagre i den rekkefølge de lastes av med den første losset som den første som skal fraktes videre (kø)
  - lagre i den rekkefølge de lestes med den siste losset som den første som skal fraktes videre (stabel/stack)
  - lagre på ID for hver destinasjon (ordnet liste)
- Eksempelet gir en liten pekepinn på hvordan forskjellige datastrukturer har sine fordeler/bakdeler for forskjellige problemstillinger

HØGSKOLEN I BERGEN

Algoritmer og effektivitetsanalyse

- Generelt: Finne tidsforbruk og krav til ekstra lagerplass ved utføring av en algoritme
- Ofte: Ser bare på tidsforbruk (nå: tid viktigere enn billig lagerplass)
- Ved analyse: - Prøver å regne ut antall viktige/ dominerende operasjoner når en algoritme blir utført
  - mest relevant ved store datamengder
- Ved testkjøring: Generer store mengder ”kunstige” data og tar tiden ved kjøring av to eller flere alternative algoritmer for et problem på en bestemt PC.

HØGSKOLEN I BERGEN

Algoritmer og effektivitetsanalyse, forts.

- Ved analyse av kjøring av en metode på n data prøver en å finne en formel for tidsforbruk der konstanter og funksjoner av n inngår

  Eksempel: $tid = k_1 + k_2 * n$

  $$tid = k_1 + k_2 * n + k_3 * n^2$$

  $$tid = k_1 + k_2 * \log_2 n$$

  Det viktigste resultatet er størrelsesordenen på det største leddet når n blir svært stor.

  Eks.: $O(1)$, $O(n)$, $O(n^2)$, $O(\log_2 n)$

HØGSKOLEN I BERGEN

Neste gang

- Mandag 11. januar
  - Kap. 2. Effektivitetsnalyse av algoritmer (en intro)
  - Se litt på første obligatoriske øvingsoppgave