DAT102 - Obligatorisk innlevering 1

Leveringsfrist: 1. februar 2026

Gruppemedlemmer - *Navn (Github brukernavn)*

- Bartosz Paszkiewicz (Paszkiewicz)
- Lars Birger Bergmål (larsbirger)
- Markus Fosse Høvring (Markus-Hovring)
- Daniel Aarsand (brasswhisper-hub)

Innleveringsinstruksjoner

Innleveringen skal leveres inn som en zip med navnet **Oblig1_Grx.zip**, hvor x i Gr**x** er gruppe nummeret. **Zip-en skal inneholde følgende**:

- **En pdf med**:
  - Liste av gruppemedlemer som er med på innleveringen.
  - Skjermbilde fra kjøring av programmene.
  - Svar på teorispørsmål.
- **Et IntelliJ-prosjekt med løsninger på programmeringsoppgavene**

1. Prosjektstruktur og Filer

Kildekoden er organisert i følgende pakker i henhold til kravene:

no.hvl.dat102.filmarkiv.adt

- `FilmarkivADT.java`: Grensesnitt som definerer de pålagte operasjonene for arkivet.

no.hvl.dat102.filmarkiv.impl

- `Film.java`: Dataobjekt (POJO) for lagring av filminformasjon.
- `Sjanger.java`: Enum-klasse med oversikt over filmsjangre.
- `Filmarkiv.java`: Implementasjon basert på en tabell (Array).
- `LinearNode.java`: Hjelpeklasse for den kjedede strukturen.
- `Filmarkiv2.java`: Implementasjon basert på en kjedet struktur (LinearNode).

no.hvl.dat102.filmarkiv.klient

- `FilmarkivMain.java`: Inneholder main-metoden som starter programmet.
- `Meny.java`: Håndterer brukerlogikk og kaller på tekstgrensesnittet.
- `Tekstgrensesnitt.java`: Håndterer all kommunikasjon (Scanner/System.out) med brukeren.

no.hvl.dat102.filmarkiv.test

- `FilmarkivTest.java`: JUnit 5 tester for å verifisere funksjonaliteten i arkivene.

2. Implementasjonsdetaljer

Tabell-implementasjon (Filmarkiv)

Denne versjonen bruker en `Film[]`-tabell. Metoden `utvid()` sørger for at kapasiteten dobles automatisk når tabellen er full. Ved sletting av en film flyttes det siste elementet i tabellen til plassen som ble ledig for å unngå tomme plasser ("hull") i strukturen.

Kjedet struktur (Filmarkiv2)

Denne versjonen bruker noder (`LinearNode`) som peker til neste element. Ved innsetting legges nye filmer alltid først i kjeden ($O(1)$), mens sletting og søking krever at vi vandrer gjennom kjeden til vi finner riktig element.

3. Oppgave 3

a)

Vi finner O-notasjonen ved å se på det raskest voksende leddet:

1. $4n^2 + 50n - 10 \rightarrow \mathbf{O(n^2)}$
2. $10n + 4\log_2 n + 30 \rightarrow \mathbf{O(n)}$
3. $13n^3 + 22n^2 + 50n + 20 \rightarrow \mathbf{O(n^3)}$
4. $35 + 13\log_2 n \rightarrow \mathbf{O(\log n)}$

b)

Løkken halverer variabelen `i` for hver iterasjon. Antall iterasjoner blir derfor $\log_2 n$. **Effektivitet:** $O(\log n)$.

c)

Den ytre løkken går n ganger. Den indre løkken dobles for hver gang og går $\log n$ ganger. **Effektivitet:** $O(n \log n)$.

d)

Areal: $2\pi r^2 \Rightarrow O(r)^2$

Omkrets: $2\pi r \Rightarrow O(r)$

e)

Metoden har to nøstede løkker som sammenligner alle elementer med hverandre. For n elementer gir dette en kvadratisk vekst. **Effektivitet:** $O(n^2)$.

f)

1. $t_4(n) = 4\log_2 n + 2n \rightarrow O(n)$
2. $t_3(n) = 20n + 2n\log_2 n + 11 \rightarrow O(n \log n)$
3. $t_1(n) = n^2 + 5n + 10 \rightarrow O(n^2)$
4. $t_2(n) = 10\log_2 n + 2^n + 20 \rightarrow O(2^n)$

g)

Ved måling av en lineær løkke ($O(n)$) ser vi at tidsbruken dobles når datamengden n dobles. Variasjoner i målingene skyldes OS-bakgrunnsprosesser, og bør derfor midles over flere kjøringer.

4. Kjøring og Testing

1. Åpne prosjektet i IntelliJ.
2. Kjør `FilmarkivMain` for å starte programmet.
3. Kjør `FilmarkivTest` for å bekrefte at alle metoder passerer kravene.