

Gruppemedlemer:

Eirik Innselset Bjørdal

Brage Bakken

Oppgave 3)

Hvis vi avslutter rekken på $n-1$ så blir svaret:

$$\frac{1+n-1}{2} * (n-1) = \frac{n-1(n)}{2} = \frac{n^2-n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

Oppgave 3a)

- i. $O(n^2)$
- ii. $O(n)$
- iii. $O(n^3)$
- iv. $O(\log n)$

Oppgave 3b)

Algoritmen har en tilordning i begynnelsen ($\text{sum} = 0$) og en tilordning for hver gang løkken kjører ($\text{sum} = \text{sum} + i$). Hver gang løkken kjører blir i halvert og løkken kjører så lenge i er større enn, eller lik, 1. Antall ganger i kan deles på 2 før det blir mindre enn, eller lik, 1 kan uttrykkes som $\log_2(i)$. Dette er altså hvor mange ganger algoritmen kjører. Antall tilordninger blir derfor $1 + \log_2(i)$. Uttrykt i O-notasjon blir dette

$$O(1) + O \log_2 n = O \log n$$

Oppgave 3c)

Algoritmen har en tilordning før løkken, den ytre løkken kjører n ganger og den indre løkken kjører $O \log n$ ganger hver gang den ytre kjører. Det totale antall tilordninger blir derfor

$$1 + n * O \log n, \text{ i O-notasjon blir dette } O(1) + O(n \log n) = O(n \log n)$$

Oppgave 3d)

Arealet i O-notasjon blir $O(r^2) = O(n^2)$

Omkretsen blir $O(r) = O(n)$

Oppgave 3e)

Den ytre løkken kjører $n-1$ ganger. Den indre løkken kjører $n - 1$ - indeks for hver gang den ytre kjører. Siden 1 og indeks er konstanter så er de ikke interessante.

I O-notasjon blir dette $O(n) * O(n) = O(n^2)$

Oppgave 3f)

I O-notasjon og rangert fra raskest til tregeest:

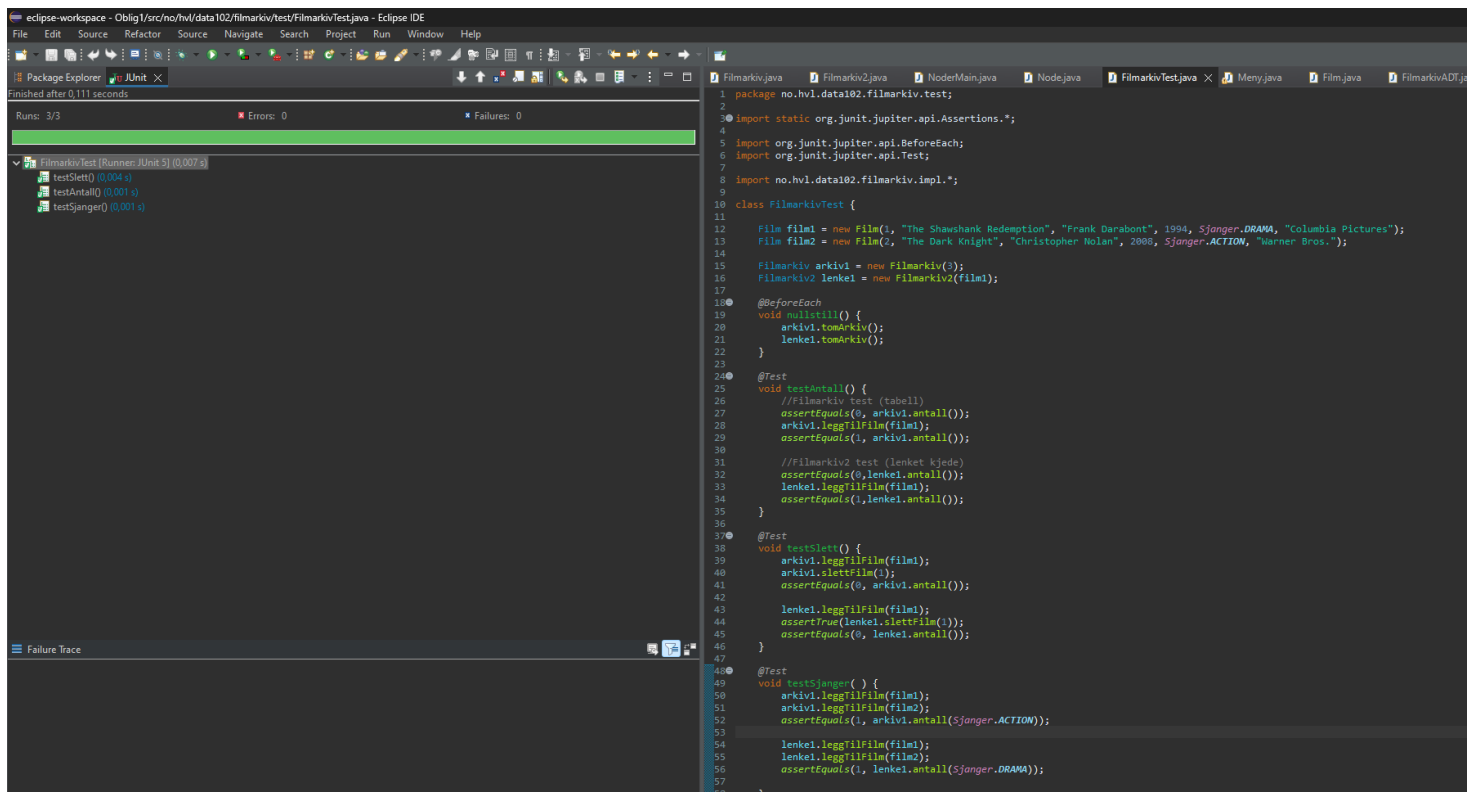
- i. $10 \log_2 n + 20 = O(\log n)$
- ii. $4 \log_2 n + 2n = O(n)$
- iii. $20n + 2n \log_2 n + 11 = O(n \log n)$
- iv. $8n + 4n^3 = O(n^3)$

Oppgave3g)

Vekstfunksjonen til tid()-metoden $T(n) = cn$, der c er en konstant fordi løkken utføres n ganger og for hver gang løkken utføres så er c de faste operasjonen som blir gjennomført. Det blir der for c operasjoner per n gjennomgang av løkken, også uttrykt som $c*n$ eller cn .

Skjermbilder fra programmet:

Junit testing:



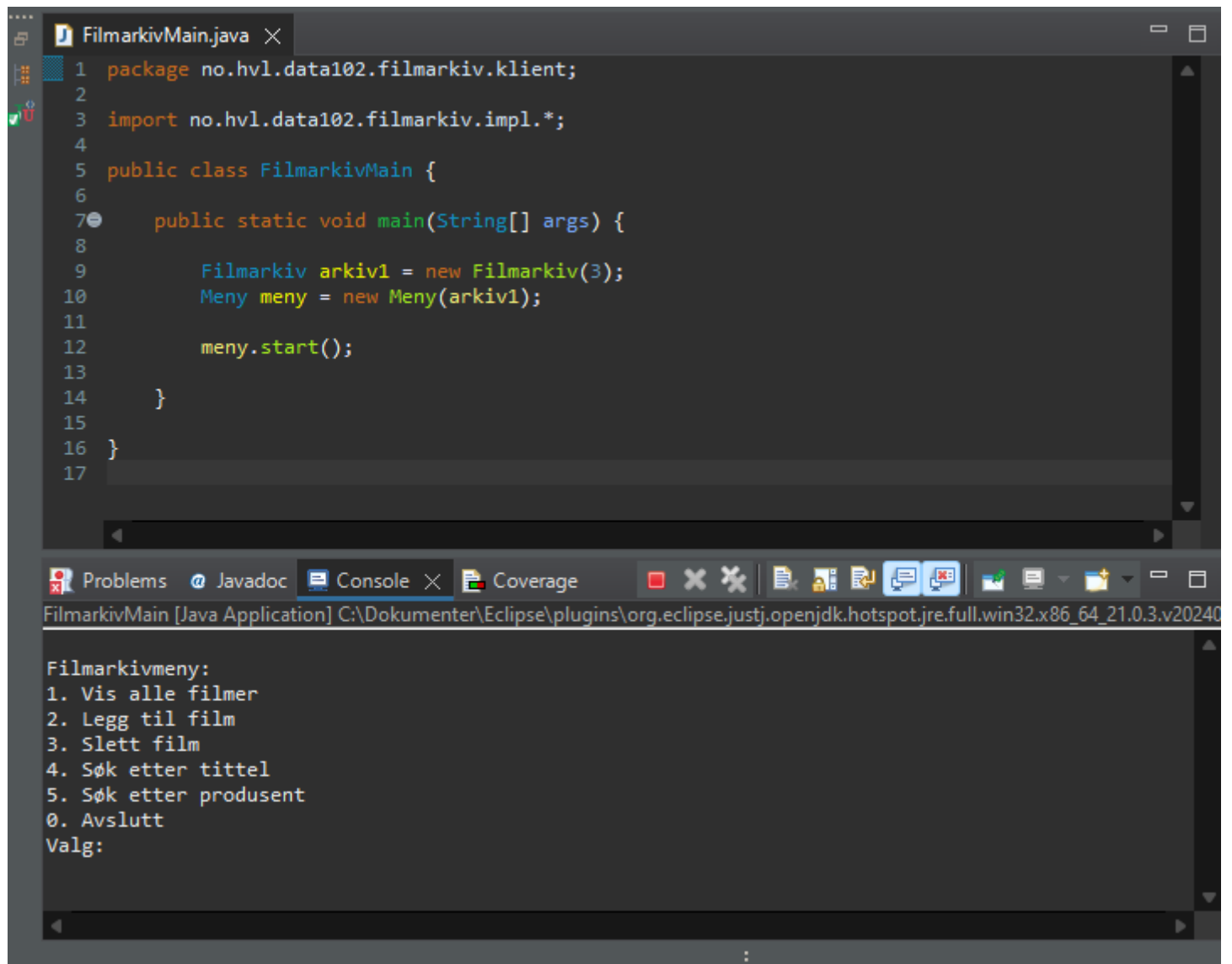
The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays the test results for 'FilmmarkivTest' (Runner: JUnit 5) with 0 errors and 0 failures. The test results table shows:

Test Name	Time (s)
testSlett()	0.004
testAntall()	0.001
testSjanger()	0.001

On the right, the Java editor shows the source code for 'FilmmarkivTest.java'. The code defines a 'Film' class with attributes for title, director, year, genre, and studio. It also defines a 'Filmmarkiv' class with methods for adding, removing, and counting films. The 'FilmmarkivTest' class contains JUnit tests for these methods.

```
1 package no.hvl.data102.filmmarkiv.test;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 import org.junit.jupiter.api.BeforeEach;
6 import org.junit.jupiter.api.Test;
7
8 import no.hvl.data102.filmmarkiv.impl.*;
9
10 class FilmmarkivTest {
11
12     Film film1 = new Film(1, "The Shawshank Redemption", "Frank Darabont", 1994, Sjanger.DRAMA, "Columbia Pictures");
13     Film film2 = new Film(2, "The Dark Knight", "Christopher Nolan", 2008, Sjanger.ACTION, "Warner Bros.");
14
15     Filmmarkiv arkiv1 = new Filmmarkiv(3);
16     Filmmarkiv2 lenkel = new Filmmarkiv2(film1);
17
18     @BeforeEach
19     void setUp() {
20         arkiv1.tomArkiv();
21         lenkel.tomArkiv();
22     }
23
24     @Test
25     void testAntall() {
26         //Filmmarkiv test (tabell)
27         assertEquals(0, arkiv1.antall());
28         arkiv1.leggTilFilm(film1);
29         assertEquals(1, arkiv1.antall());
30
31         //Filmmarkiv2 test (lenket kjede)
32         assertEquals(0, lenkel.antall());
33         lenkel.leggTilFilm(film1);
34         assertEquals(1, lenkel.antall());
35     }
36
37     @Test
38     void testSlett() {
39         arkiv1.leggTilFilm(film1);
40         arkiv1.slettFilm(1);
41         assertEquals(0, arkiv1.antall());
42
43         lenkel.leggTilFilm(film1);
44         assertEquals(1, lenkel.slettFilm(1));
45         assertEquals(0, lenkel.antall());
46     }
47
48     @Test
49     void testSjanger() {
50         arkiv1.leggTilFilm(film1);
51         arkiv1.leggTilFilm(film2);
52         assertEquals(1, arkiv1.antall(Sjanger.ACTION));
53
54         lenkel.leggTilFilm(film1);
55         lenkel.leggTilFilm(film2);
56         assertEquals(1, lenkel.antall(Sjanger.DRAMA));
57     }
58 }
```

FilmarkivMain:



The screenshot shows the Eclipse IDE with the `FilmarkivMain.java` file open. The code defines a package, imports, and a `main` method that creates `Filmarkiv` and `Meny` objects and starts the menu.

```
1 package no.hvl.data102.filmarkiv.klient;
2
3 import no.hvl.data102.filmarkiv.impl.*;
4
5 public class FilmarkivMain {
6
7     public static void main(String[] args) {
8
9         Filmarkiv arkiv1 = new Filmarkiv(3);
10        Meny meny = new Meny(arkiv1);
11
12        meny.start();
13    }
14 }
15
16 }
17
```

The console output shows the menu options:

```
Filmarkivmeny:
1. Vis alle filmer
2. Legg til film
3. Slett film
4. Søk etter tittel
5. Søk etter produsent
0. Avslutt
Valg:
```

```
FilmarkivMain.java X
1 package no.hvl.data102.filmarkiv.klient;
2
3 import no.hvl.data102.filmarkiv.impl.*;
4
5 public class FilmarkivMain {
6
7     public static void main(String[] args) {
8
9         Filmarkiv arkiv1 = new Filmarkiv(3);
10        Meny meny = new Meny(arkiv1);
11
12        ...
13    }
14}
```

Problems Javadoc Console Coverage

FilmarkivMain [Java Application] C:\Dokumenter\Eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240...

Filmarkivmeny:

1. Vis alle filmer
2. Legg til film
3. Slett film
4. Søk etter tittel
5. Søk etter produsent
0. Avslutt

Valg: 1

Filmnummer: 1

Tittel: The Shawshank Redemption

Produsent: Frank Darabont

Lanseringsår: 1994

Sjanger: DRAMA

Filmselskap: Columbia Pictures

Filmnummer: 2

Tittel: The Dark Knight

Produsent: Christopher Nolan

Lanseringsår: 2008

Sjanger: ACTION

Filmselskap: Warner Bros.