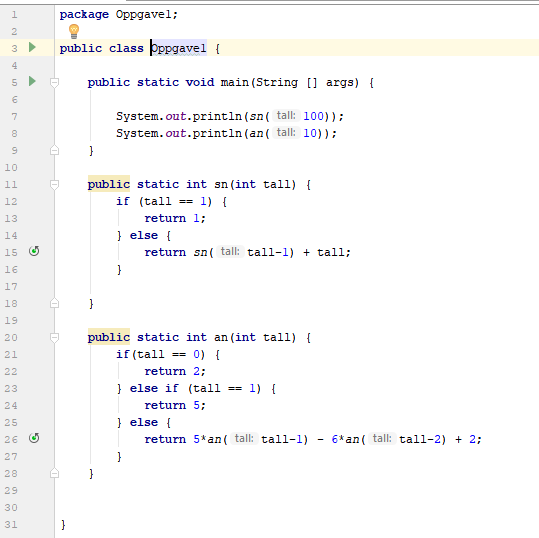
Oblig 3

Bruker programmet (IntelliJ IDEA Community Edition 2017.3.4 x64) til koding og kjøring.

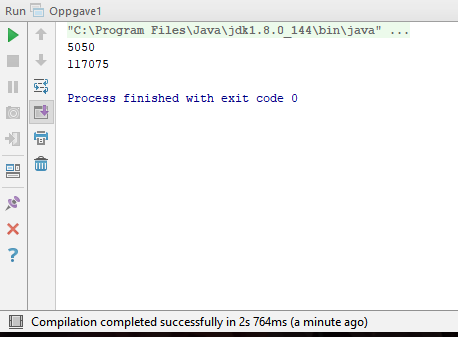
# Oppgave 1

## A og B)

Metoder og main

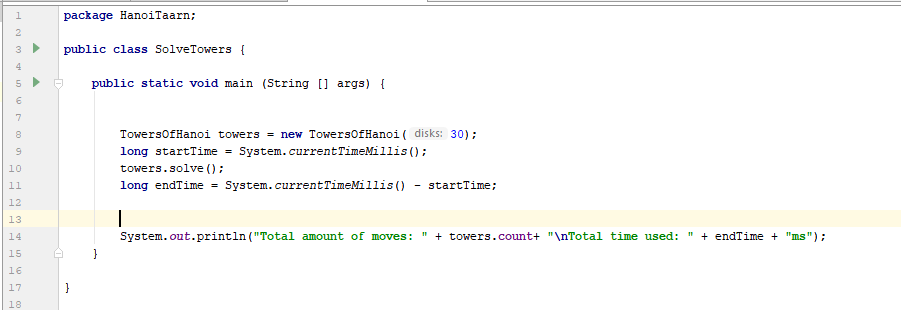


Utklipp av kjøring av main

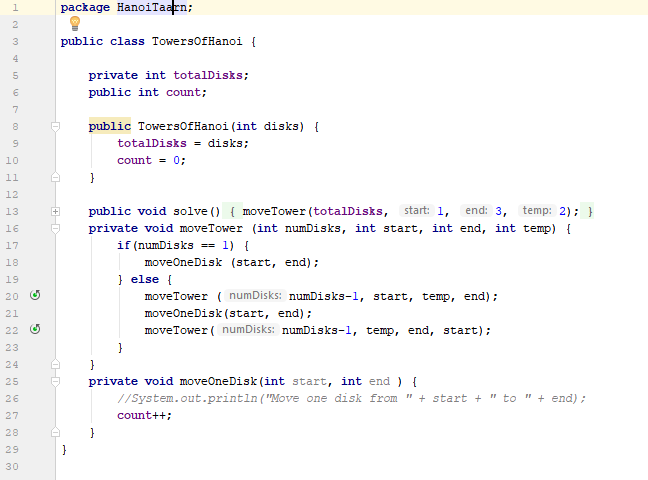


## C)

SolveTowers:



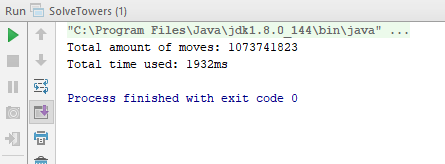
TowersofHanoi:



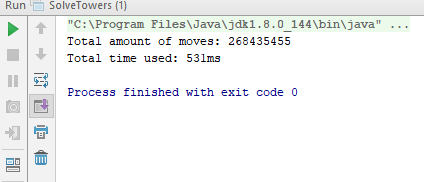
Utskrift av SolveTowers:

SolveTowers:

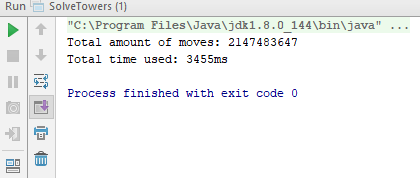
30 disks



28 disks



31 disks



## iii)



Tid32/Tid16 = 7575ms/1ms= 7575

((2^32)-1)/((2^16)-1)= 4294967295/65535= 65537

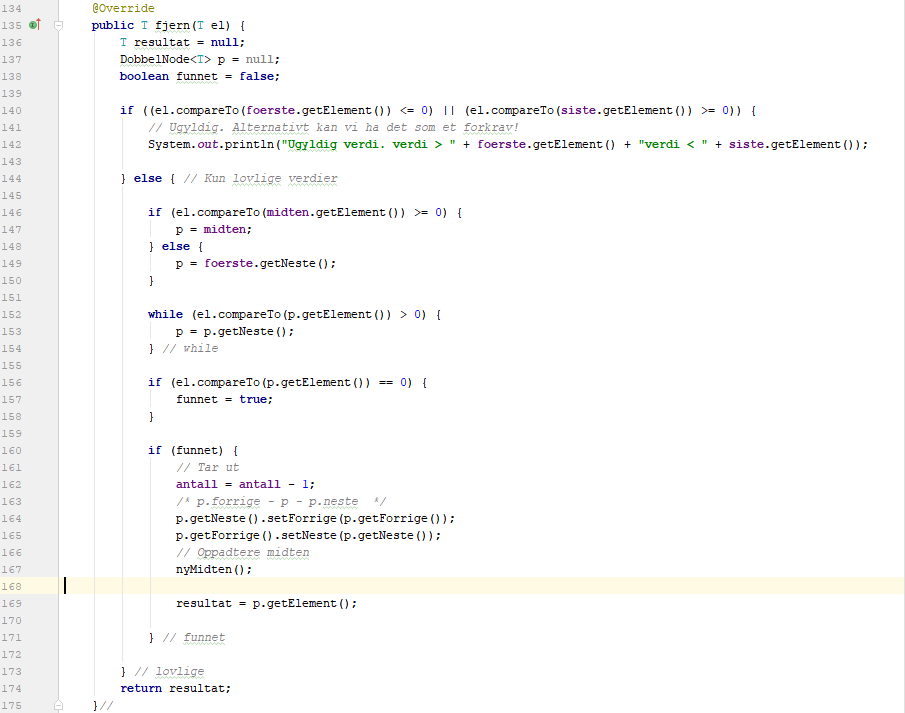
# Oppgave 2

## a)

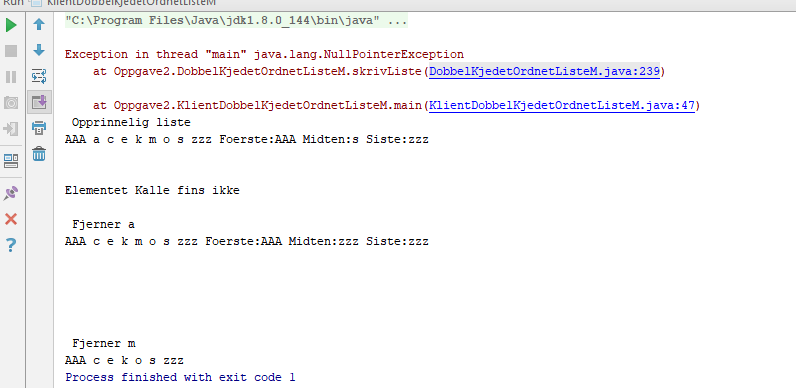
Metoden leggtil:



Metoden fjern:



Utskrift med metoden fjern og leggtil:

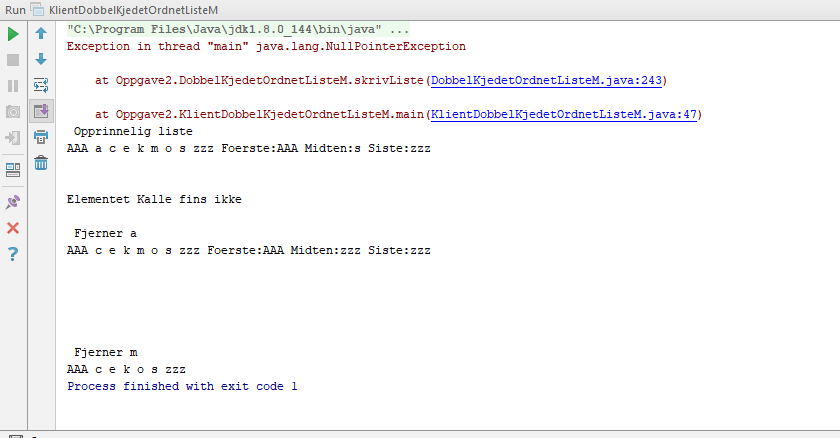


## b)

Modifisert metoden fjern:



Kjøring med fjern som bruker finn-metoden:



## c)

Grunnen til at metoden nyMidten er i O(n) er:

Siden metoden bruker en for løkke vil den være på O(n) og være avhengig av listen.

For å få metoden i O(1) kan en fjerne for løkken og heller bruke if og else if.

## d)

I) Når vi ikke har noen midtpeker:

Verste: n-1  
gj.snitt: n/2

II) Når vi har midtpeker og søking kun en vei:  
verste: (n-1)/2  
gj.snitt: n/4

III)

1. 2)  
   3)  
   4)

# Oppgave 3

## a)

Binærsøking er en algoritme der en deler en sortert liste i to for hvert steg og sammenligner med det elementet den skal finne fram, er elementet i mindre enn det vi leter etter vil den ta den midtre delen som er større.

Public static int Binærsøk(int verdi, int[] tabell, int venstre, int hoyre){

Int midten = (venstre + hoyre)/2;

If(hoyre<venstre){

Return FINNES\_IKKE;

}

Else if(verdi<tabell[midten]){

Return Binærsøk(verdi, tabell, venstre, midten-1);

}

Else if(verdi>tabell[midten]){

Return Binærsøk(verdi, tabell, midten-1, hoyre);

}

Else if (verdi==tabell[midten]){

Return midten;

}

Return FINNES\_IKKE; }

Første if er hvis en skrev inn en ugyldig verdi. Andre if er hvis søkeverdien er mindre enn midten skal en søke i midten av venstre siden. Tredje if er hvis søkeverdien er større en midten skal en søke i høyre siden av midten. Fjerde if er hvis søkeverdien er lik verdien til midten. Siste setning er om en ikke fant verdien så finnes ikke den.

## b)

Gitt en sortert liste av 15 verdier som vi ønsker å bruke binærsøking på:

2 4 5 7 8 10 12 15 18 21 23 27 29 30 31

Vi skal finne 8.

Da vil metoden først finne midten i hele listen som blir verdien 15. Metoden ser at verdien er større enn det vi søker og vil da finne midten på venstre siden, den nye midten vil da være 7.

2 4 5 7 8 10 12 15

Metoden ser at verdien er mindre enn det vi søker etter og vil da finne midten på høyre siden og vil da finne verdien 10.

7 8 10

Siden 10 er større enn søkeverdien 8 vil metoden finne midten på venstre siden og vil da ende opp med 8.

8

Metoden gikk gjennom 4 midtpunkt før den fant verdien. Hadde vi hatt en vanlig for-løkke ville den brukt 5 i dette tilfellet.

## C)

Vi skal finne 16 i den samme tabellen, noe som ikke finns. Prosessen vil være slik:

Metoden finner først midten i hele listen som blir verdien 15. Metoden ser at verdien er mindre enn det vi søker og vil da finne midten på høyre siden, den nye midten vil da være 23.

15 18 21 23 27 29 30 31

Metoden ser at verdien er større enn det vi søker etter og vil finne midten på venstre siden, den nye verdien blir 18.

15 18 21 23

Metode ser igjen at verdien er større enn det vi søker og prøver å lage ny midten på venstre siden, men finner ingen verdi, da vil metoden returnere FINNES\_IKKE som er en final verdi som er lik -1.

Metoden gikk gjennom 4 ganger.

Verste tilfelle med binærsøk er [log\_2 n = ln n/ ln2]

Skriver vi inn lengden på denne tabellen får vi ln 15/ln2=3.9 noe vi kan runde opp til 4 som stemmer med det vi fikk.

# Oppgave 4

## a)

Quicksort

|  |  |  |  |
| --- | --- | --- | --- |
| n | Antall målinger | Målt tid(gj.snitt) | Teoretisk tid c\*f(n)  C= tid/(n\*log2\*n) |
| 32000 | 5 |  | 305ms/(32000\*(ln32000/ln2))= 6.37\*10^-6 |
| 64000 | 5 |  | 550ms/(64000\*(ln64000/ln2))= 5.38\*10^-4 |
| 128000 | 5 |  | 920ms/(128000\*(ln128000/ln2))= 4.24\*10^-4 |

Flettesortering:

|  |  |  |  |
| --- | --- | --- | --- |
| n | Antall målinger | Målt tid(gj.snitt) | Teoretisk tid c\*f(n)  C=tid/n\*log2\*n |
| 32000 | 5 |  | 363ms/(32000\*(ln32000/ln2))= 7.58\*10^-4 |
| 64000 | 5 |  | 725ms/(64000\*(ln64000/ln2))= 7.10\*10^-4 |
| 128000 | 5 |  | 1176ms/(128000\*(ln128000/ln2))= 5.42\*10^-4 |

Boblesortering:

|  |  |  |  |
| --- | --- | --- | --- |
| n | Antall målinger | Målt tid(gj.snitt) | Teoretisk tid c\*f(n) C=tid/n^2 |
| 32000 | 5 | 3060ms | 8579ms/32000^2= 8.3\*10^-6 |
| 64000 | 5 | 12778ms | 44693ms/64000^2= 1.09\*10^-5 |
| 128000 | 5 | 55245ms | 189479ms/128000^2= 1.15\*10^-5 |

Radix-sortering:

|  |  |  |  |
| --- | --- | --- | --- |
| n | Antall målinger | Målt tid(gj.snitt) | Teoretisk tid c\*f(n) |
| 32000 | 5 |  |  |
| 64000 | 5 |  |  |
| 128000 | 5 |  |  |

Sortering ved innsetting:

|  |  |  |  |
| --- | --- | --- | --- |
| n | Antall målinger | Målt tid(gj.snitt) | Teoretisk tid c\*f(n)  C=tid/n^2 |
| 32000 | 5 | 846ms | 3790ms/32000^2= 3.70\*10^-6 |
| 64000 | 5 | 4366ms | 16190ms/64000^2= 3.95\*10^-6 |
| 128000 | 5 | 18668ms | 76846ms/128000^2= 4.69\*10^-6 |

Sortering ved utvalg:

|  |  |  |  |
| --- | --- | --- | --- |
| n | Antall målinger | Målt tid(gj.snitt) | Teoretisk tid c\*f(n)  C=tid/n^2 DONT |
| 32000 | 5 | 930ms |  |
| 64000 | 5 | 4124ms |  |
| 128000 | 5 | 17175ms |  |

## b)

## c)

## d)