# DAT102 – Våren 2026

Øvingsoppgaver uke6 (2. – 6. februar)

## Introduksjon

**Oppgavene i dette settet er i utgangspunktet frivillige, men deler vil muligens inngå i Oblig2! Dette er ikke helt bestemt ennå.**

## Oppgave 1 – Parentessjekker (Anvendelse av Stabel)

Bruk en av implementasjonene av StabelADT vi gjorde i forelesningen (LenketStabel eller TabellStabel) som en del av løsningen på denne oppgaven.

a)

Lag en klasse `ParentesSjekker` med metoden boolean `sjekkParenteser(String s)`, som sjekker om en String s bare innholder korrekte parentespar, altså at alle startparenteser har matchende sluttparenteser (og omvendt), og at rekkefølgen er riktig.

Eks:

- `"{ [ ( ) ] }"` er korrekt
- `"{ [ ( ) }"` er ikke korrekt. Mangler sluttparentes ]
- `"[ ( ) ] }"` er ikke korrekt. Mangler startparentes {
- `"{ [ ( ] ) }"` er ikke korrekt. Sluttparentes ] kommer for tidlig.

Eks:

```
String javaprogram = """
      class HelloWorld {
          public static void main(String[] args) {
              System.out.println("Hello World!");
          }
      }
      """;
```

er korrekt. Her inneholder stringen mer enn parenteser, men parentesene er korrekt.

**TIPS!** Se forelesning «F05 Stabel» for en grov beskrivelse av hva som må gjøres.

**TIPS!** En String kan gjøres om til en char[] ved å bruke s.toCharArray().

**TIPS!** Det er sikkert lurt å lage hjelpemetoder for å sjekke detaljer, f.eks.:

- boolean erStartParentes(char c)
- boolean erSluttParentes(char c)
- boolean erParentesPar(char start, char slutt)

b)

Lag JUnit enhetstester i en klasse `ParentesSjekkerTest` som verifiserer at `ParentesSjekker.sjekkParenteser()` virker på de 5 eksemplene over.

## Oppgave 2 – Rekursjon

a) Summen av de n første naturlige tall er gitt ved: $S_n = 1+2+3+\ldots+n$. En formel for å finne $S_n$ er gitt ved:

$S_n = S_{n-1} + n$, $S_1 = 1$

Lag en rekursiv Java-metode som beregner $S_n$, og skriv et enkelt hovedprogram som bruker denne metoden for å finne $S_{100}$.

Hva er kjøretidsutviklingen for denne metoden, uttrykt i O-notasjon?

b) Et palindrom er en tekst som er lik forlengs og baklengs. Eksempler på palindromer:

- «abba»
- «regninger»

Lag en rekursiv metode `boolean isPalindrome(String aString)` som sjekker om en String er et palindrom. Lag JUnit-tester som tester ut metoden på en del ulike strenger, både palindromer og ikke-palindromer.

Hva er kjøretidsutviklingen for denne metoden, uttrykt i O-notasjon (n er lengden på strengen)?

c) Fibonacci-tallene dukker opp både i naturen og innenfor økonomi. De kan defineres på ulike måter. En rekursiv definisjon er:

$f_n = f_{n-1} + f_{n-2,}$ og $f_0 = 0, f_1$=1

Lag en rekursiv Java-metode som beregner $f_n$. Lag et hovedprogram der du prøver litt ulike verdier av n. Etter hvert som n øker, vil du oppdage at det tar lang tid å utføre metoden.

Hva er kjøretidsutviklingen for denne metoden, uttrykt i O-notasjon?

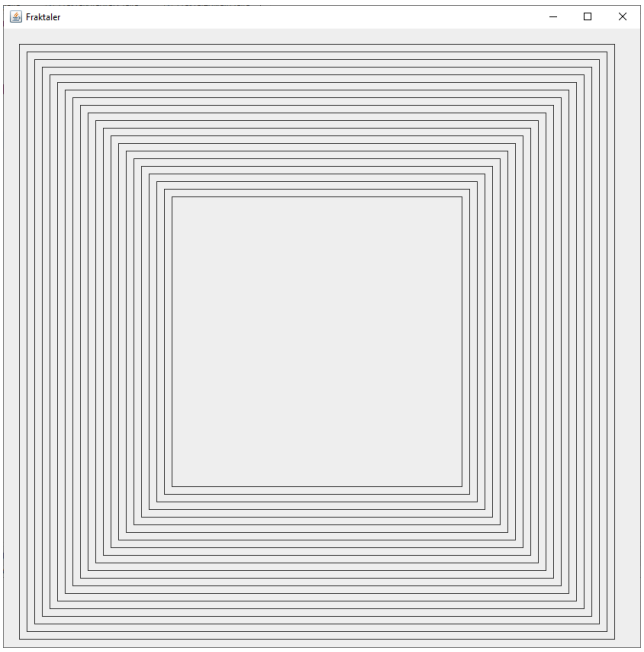
d) Det er lett å lage en ikke-rekursiv metode for å beregne Fibonacci-tallene. Lag en slik metode og observer at den vil være rask å utføre for verdier av n der den rekursive metoden bruker lang tid. Tips: Beregn $f_2, f_3, \ldots, f_n$.

Hva er kjøretidsutviklingen for denne metoden, uttrykt i O-notasjon?

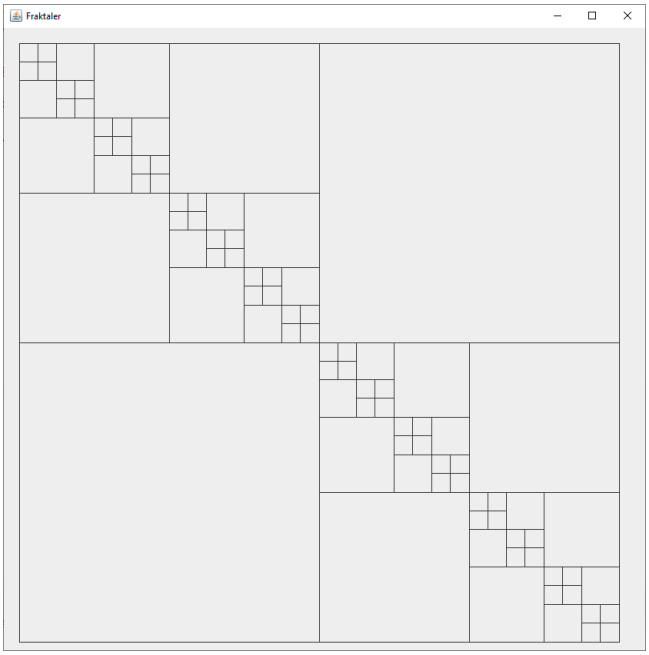
## Oppgave 3 – Rekursjon

Ta utgangspunkt i fraktal-eksemplene som ligger i https://github.com/lars-petter-helland/dat102-v2025 i prosjektet **f07-...** og pakken **...fraktaler**.

Kjør FraktalerMain og studér klassen **FraktalKvadrat1** med den rekursive metoden **drawSquare()** som produserer figuren under (depth = 20). Forstår du hva som skjer?

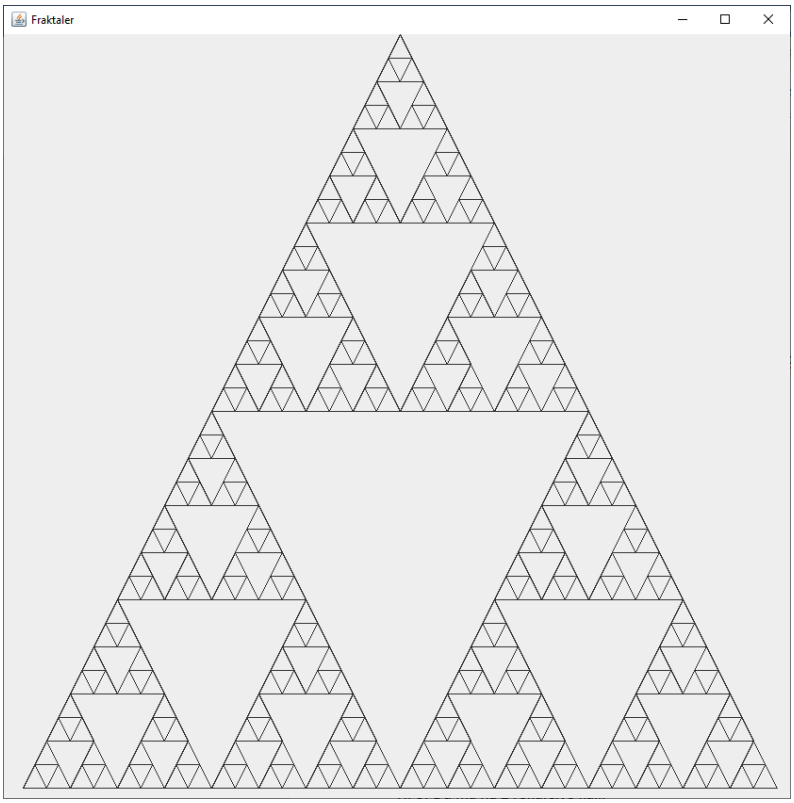



Prøv å lage en annen tilsvarende klasse **FraktalKvadrat2** (copy-paste fra FraktalKvadrat1 ...) som produserer denne figuren (depth = 5):




TIPS! Du må ha 2 rekursive kall.

Eller hva med denne (depth = 5)?




TIPS! Du må ha 3 rekursive kall.