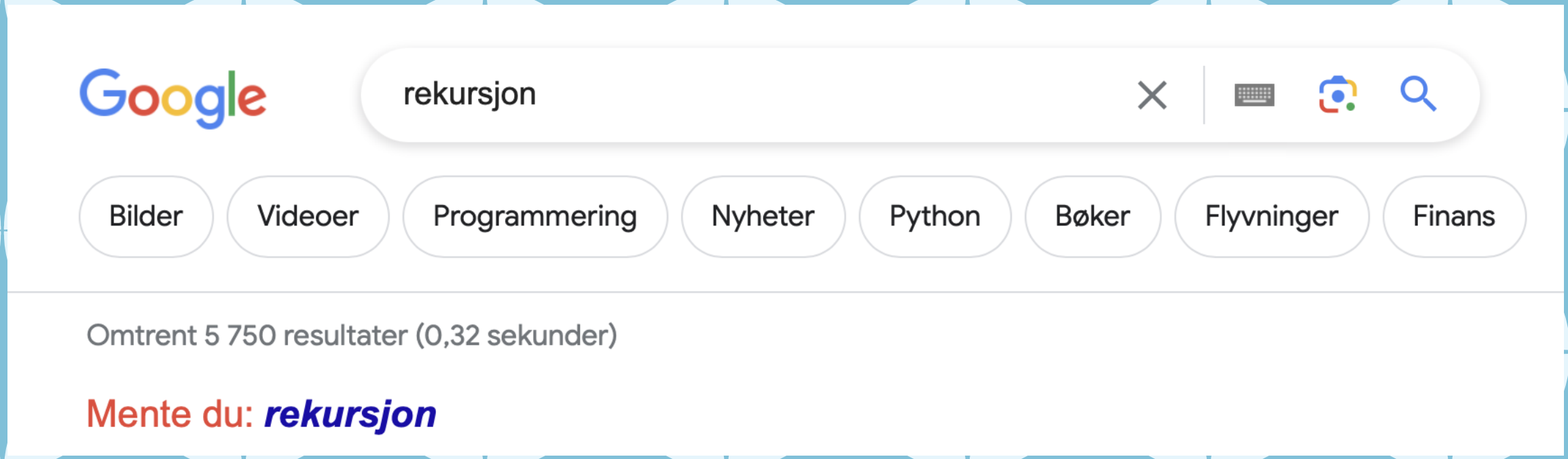


REKURSJON

Uke 12 IN1010
Agneshb@uio.no

Snublegruppe og grublegruppe

Etter påske vil vi «omdøpe» to av gruppetimene til «snublegruppe» og «grublegruppe». Snublegruppen er litt ekstra tilrettelagt for deg som synes IN1010 er krevende. Grublegruppen er tilrettelagt for deg som synes at IN1010 gjerne kunne dekket mer og gått fortere frem.

- Snublegruppen vil holdes på tirsdager klokken 10–12 på Logo.
- Grublegruppen vil holdes på onsdager klokken 14–16 på Caml.

I DAG

Rekursive metoder

Call stack i rekursjon

Livekoding av en metode som teller antall hunder eller katter i en lenkeliste av Dyr.

Livekoding av fibonacci implementert rekursivt

Oppgaver



REKURSIVE METODER

Enkelt forklart: rekursive metoder er metoder som kaller på seg selv.

- Som igjen kaller seg selv
  - Som igjen kaller seg selv
    - (..)

Kan fortsette evig, som en evig while-løkke

Ønsker å stoppe en gang

Iterative løsninger kan omformuleres til rekursive løsninger.



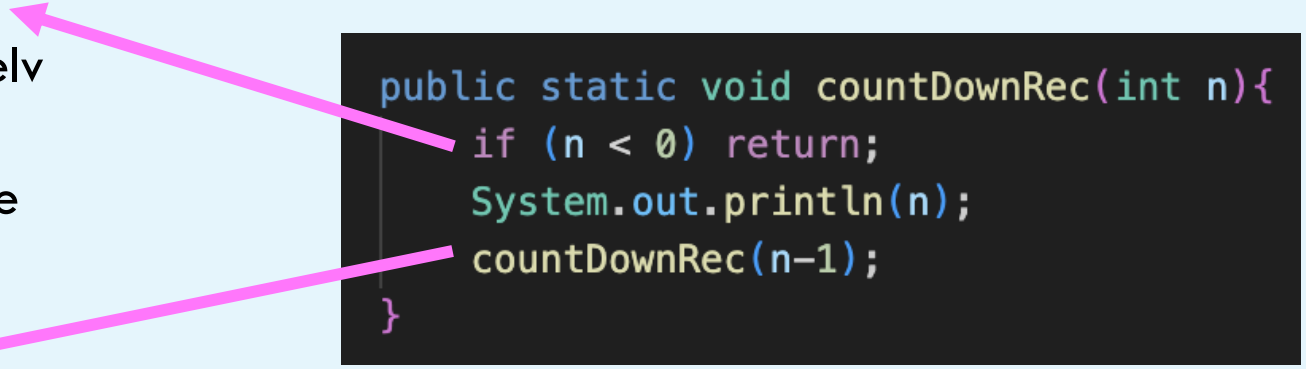
HVA INNEHOLDER EN REKURSIV METODE?

Basistilfelle (“base case”)

- Rekursjonen stopper, kaller ikke seg selv igjen → «stop case».
- Kallene «nøstes opp» fra siste til første kall.

Rekursjonssteg (“recursive step”)

- Endrer noe slik at man til slutt når basistilfellet.
- I dette tilfelle er det ett rekursivt kall, men det kan være flere rekursive kall (feks Fibonacci).



```
public static void countDownRec(int n){
    if (n < 0) return;
    System.out.println(n);
    countDownRec(n-1);
}
```

REKURSIV VS. ITERATIV

Rekursiv

```java
public static void countDownRec(int n){
    if (n < 0) return;
    System.out.println(n);
    countDownRec(n-1);
}
```

Iterativ

```java
public static void countDownIt(int n) {
    while(n >= 0) {
        System.out.println(n);
        n--;
    }
}
```

countDownRec(5)

countDownIt(5)



```
5
4
3
2
1
0
```

HVA SKJER I DEN REKURSIVE METODEN?

```java
public static void countDownRec(int n){
    if (n < 0) return;
    System.out.println(n);
    countDownRec(n-1);
}
```

```java
countDownRec(n:5);
```

n = 5 → printer **5**

→ kaller countDownRec(**4**)

n = 4 → printer **4**

→ kaller countDownRec(**3**)

n = 3 → printer **3**

→ kaller countDownRec(**2**)

n = 2 → printer **2**

→ kaller countDownRec(**1**)

n = 1 → printer **1**

→ kaller countDownRec(**0**)

n = 0 → printer **0**

→ kaller countDownRec(**-1**)

n = -1 → n er mindre enn 0 og returnerer uten rekursivt kall

CALL STACK I FORBINDELSE MED REKURSJON

Stack (Stabel):

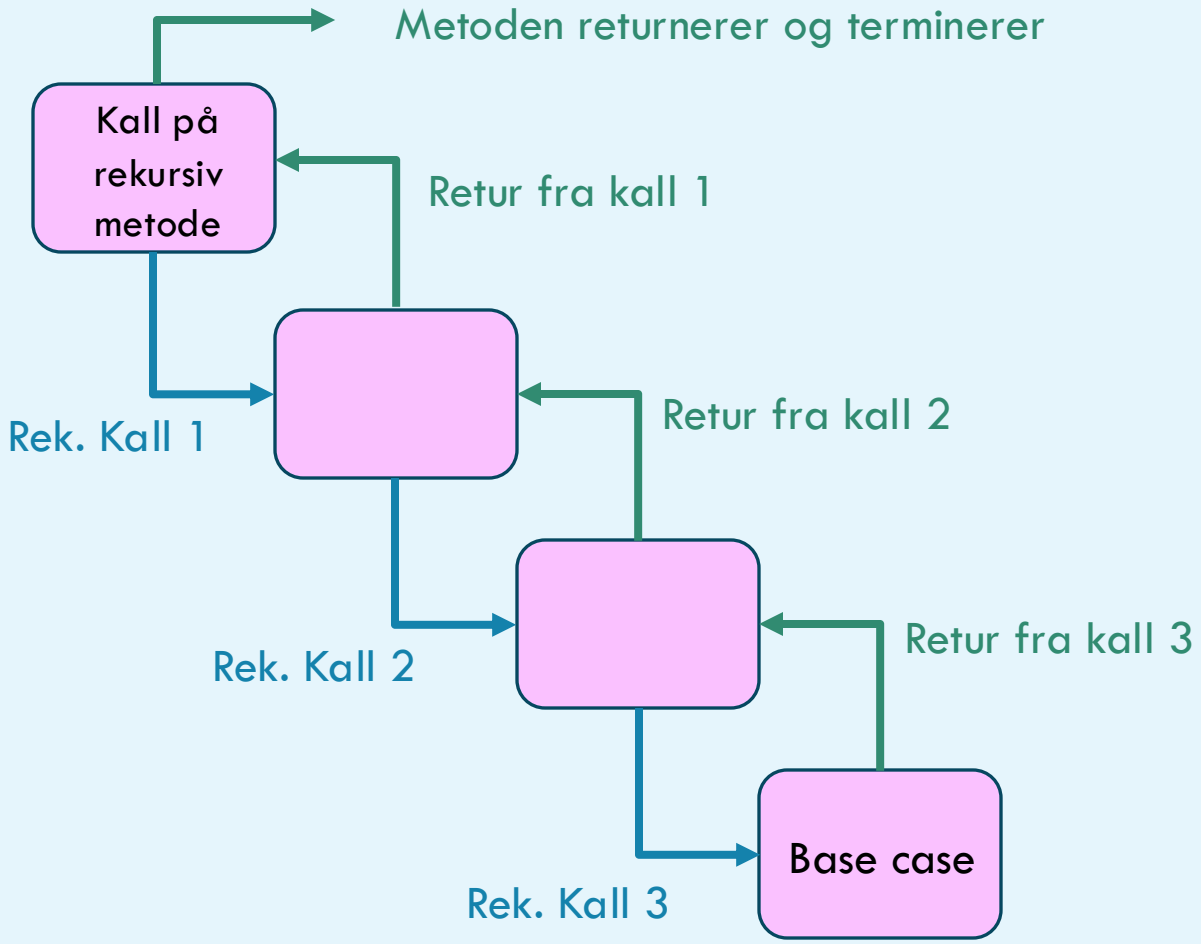
- Push() – legger til foran
- Pop() – henter ut foran

Rekursive metodekall legger seg på en **call stack**.

Når vi når basistilfellet stopper rekursjonen, og alle elementene poppes fra stacken.

- Det siste metodekallet poppes først og det første metodekallet poppes sist

Return: kan enten returnere en verdi eller terminere metodekallet.

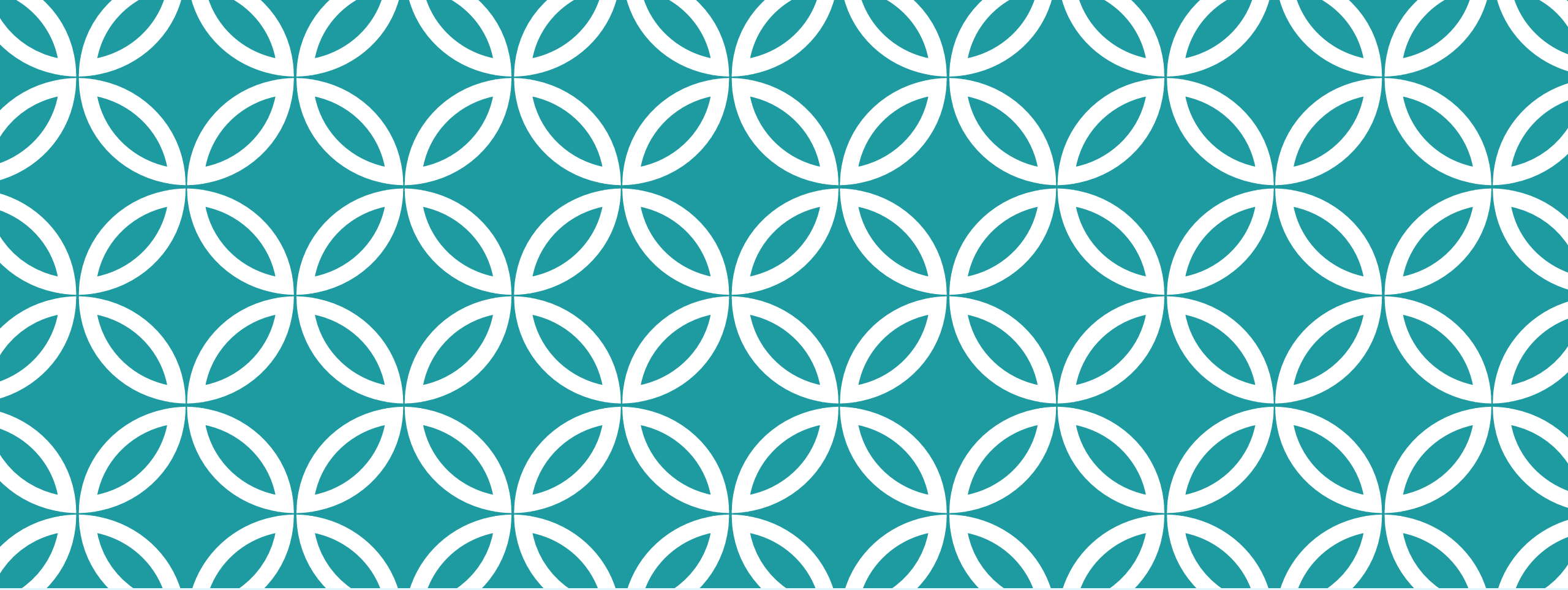


EKSEMPEL: REKURSIV METODE FOR Å SUMMERE TALL FRA N NED TIL 0

```java
public static int recursiveSum(int n) {
    // Base case: når n = 0 er det ikke flere tall å summere
    if (n == 0) {
        return 0;
    }
    // Recursive step: summerer n med resultatet av et
    //rekursivt kall på recursiveSum med n-1 som argument
    return n + recursiveSum(n-1);
}
```

Her returnerer vi resultatet av det rekursive kallet.

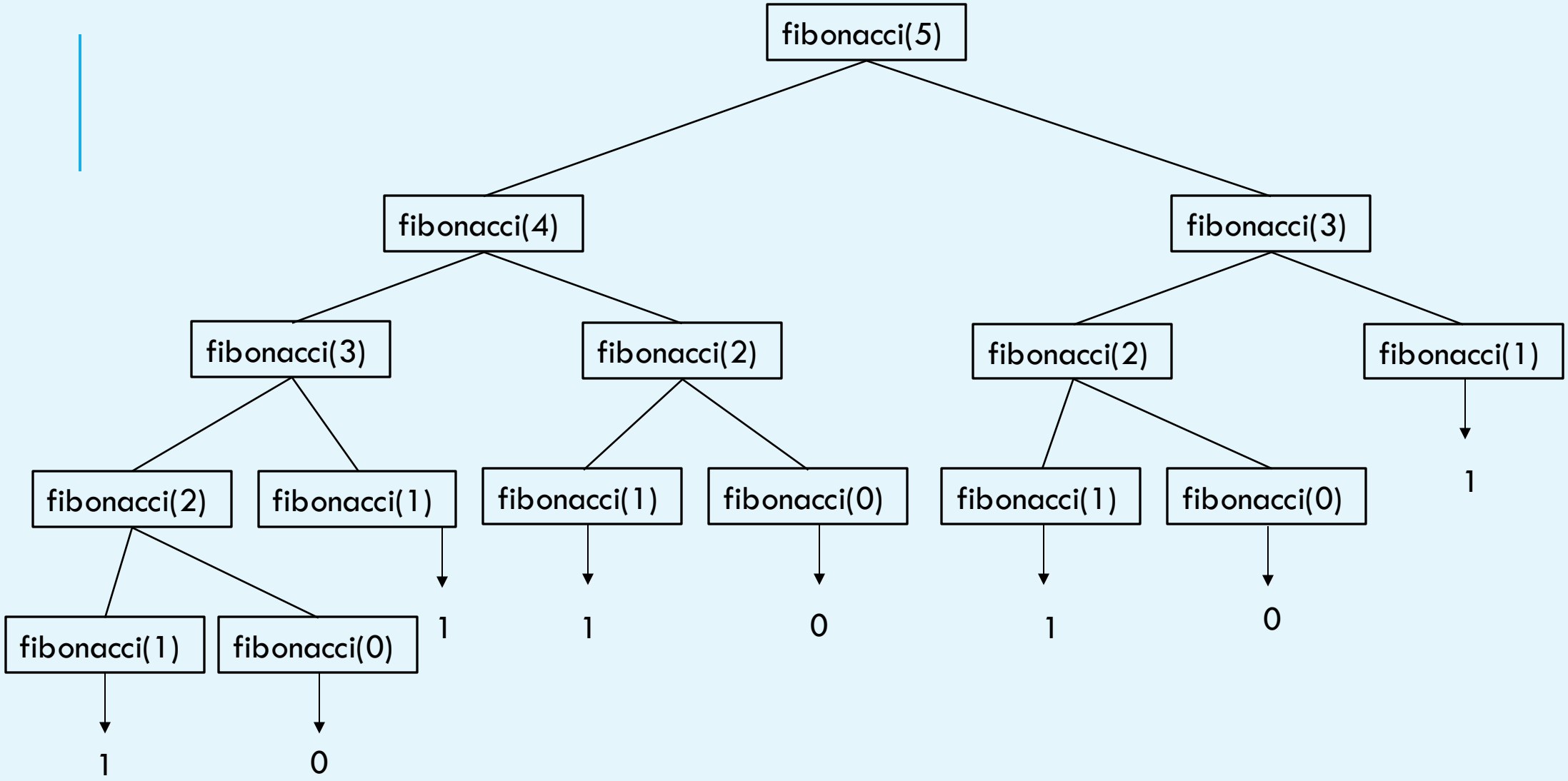
recursiveSum(5)
→ 5 + recursiveSum(4)
→ 5 + 4 + recursiveSum(3)
→ 5 + 4 + 3 + recursiveSum(2)
→ 5 + 4 + 3 + 2 + recursiveSum(1)
→ 5 + 4 + 3 + 2 + 1 + recursiveSum(0)
→ 5 + 4 + 3 + 2 + 1 + 0
→ 5 + 4 + 3 + 2 + 1
→ 5 + 4 + 3 + 3
→ 5 + 4 + 6
→ 5 + 10
→ 15

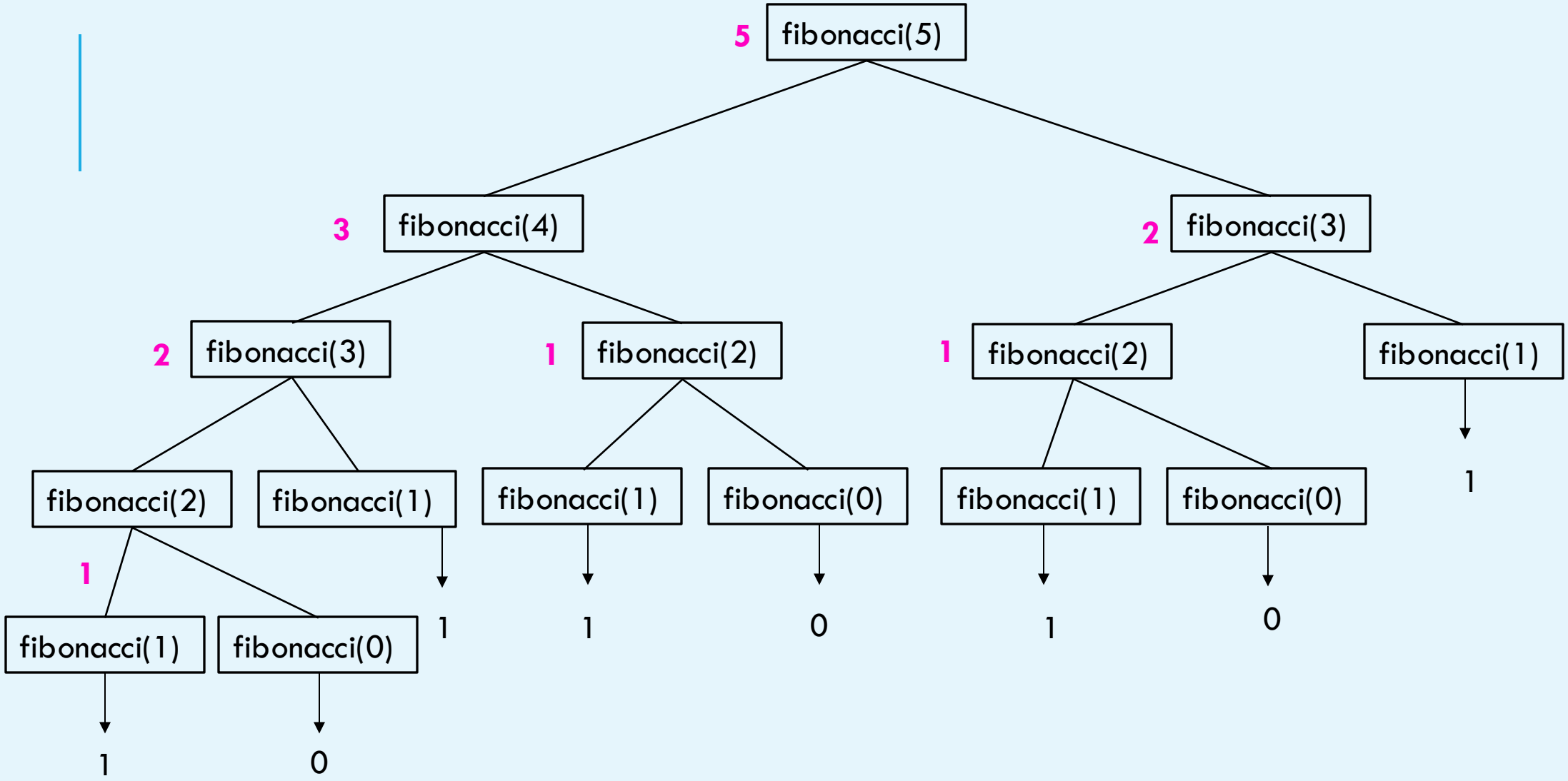


LIVEKODING

Fibonacci

tellHunder/tellKatter


fibonacci(5)
fibonacci(4)
fibonacci(3)
fibonacci(3)
fibonacci(2)
fibonacci(2)
fibonacci(1)
fibonacci(2)
fibonacci(1)
fibonacci(1)
fibonacci(0)
fibonacci(1)
fibonacci(0)
1
fibonacci(1)
fibonacci(0)
1
1
0
1
0
1
1
0


5
fibonacci(5)
3
fibonacci(4)
2
fibonacci(3)
2
fibonacci(3)
1
fibonacci(2)
1
fibonacci(2)
fibonacci(1)
fibonacci(2)
fibonacci(1)
fibonacci(1)
fibonacci(0)
fibonacci(1)
fibonacci(0)
1
1
fibonacci(1)
fibonacci(0)
1
1
0
1
0
1
0

TIL SLUTT

Dette må du vite:

- Rekursive metoder må ha med base case og rekursivt steg

Rekursjon er vanskelig i begynnelsen, men det er lurt å lære seg til eksamen (ref tidligere eksamensoppgaver)

- Løs ukesoppgaver, Trix og innlevering

Positivt: når man først har funnet base case og rekursivt steg, kreves det ofte få linjer kode☺

JOBBE MED OPPGAVER<3

- Seminaroppgaver finnes på gruppesiden til gruppe 3
- Trix
- Innlevering (men gjør gjerne noen mindre oppgaver først!)
- Prøv å tegn rekursive kall, eller gjette hva rekursive kall returnere/printer ut (øvelse til midtveis)