IN1010 uke 7

Gruppe 2

Agenda

- Rekursjon
  - Hva er det?
  - Rekursiv vs iterativ
  - Stacken og rekursjon
  - Kodeeksempel/oppgave
  - Tegne rekursjon

- Jobbe med oppgaver/trix/oblig!

Hva er rekursjon?

Rekursjon

= Metode som kaller seg selv

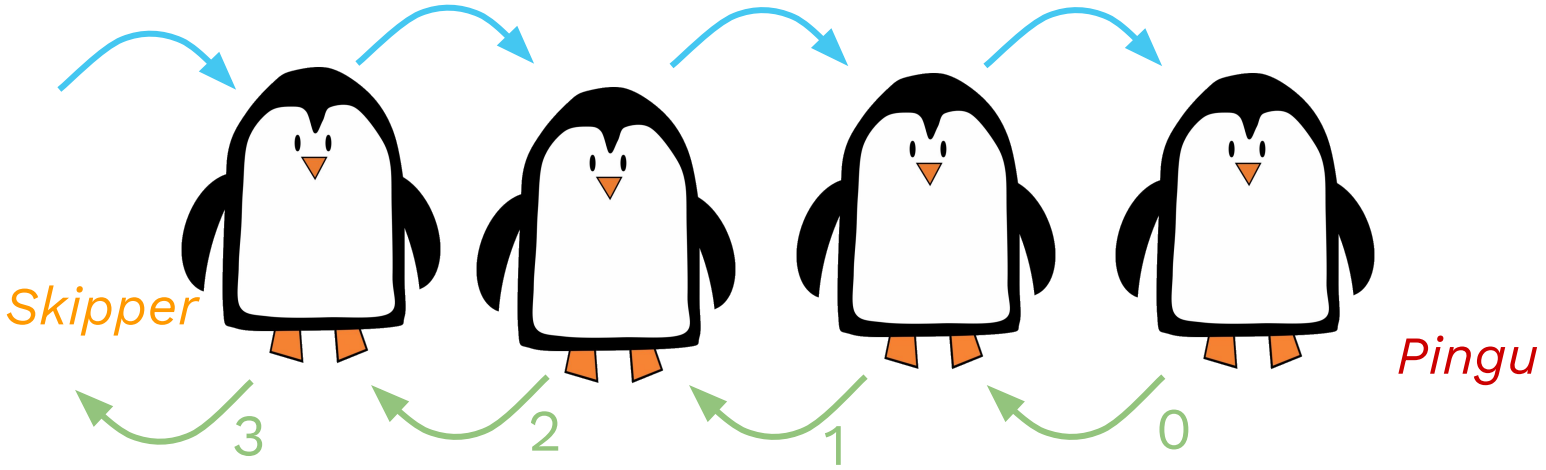
- Alternativ til iterasjon (løkker)
- Istedenfor å prøve å prøve å løse hele problemet i metoden → løser kun “neste steg”, til kommer til *basistilfellet*



Eksempel

- Skipper vil telle antall pingviner bak han
  - → 1 mer enn hva den rett bak har
  - → hver pingvin spør neste hvor mange som er bak

- Pingu: Basistilfelle, 0 bak seg



Hva printes?

```
public static void main(String[] args) {
        skrivTall(5);
}

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

Hva printes?

```
public static void main(String[] args) {
        skrivTall(5);
}

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

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

Rekursjon

- Består av 2 deler:
  - Basistilfelle
    - enkleste tilfellet
  - Rekursivt kall
    - ikke “fremme”, vil fortsette for å komme til basistilfellet
- Tips
  - Tenk på basistilfellet, skriv det først,
  - Rekursive delen: tenk at metoden allerede fungerer, slik at du kan bruke den

Rekursjon vs iterasjon

Hvordan hadde du gjort dette til en rekursiv metode? Hva er basistilfellet?

```
static int oekTil5Iterativ(int tall) {
    while (tall < 5) {
        tall ++;
    }
    return tall;
}
```

```java
static int oekTil5Rekursiv(int tall) {
    if (tall >= 5) {
        // Basistilfelle
        return tall;
    }

    // Rekursivt kall
    return oekTil5Rekursiv(tall + 1);
}
```

Hvorfor bruke rekursjon i det hele tatt?

- For noen problemer mer naturlig å bruke enn iterasjon (løkker)
- Kan gjøre koden mye mer leselig / kortere / klarere i mange tilfeller
  - Tips: Prøv å skrive om lenkeliste med rekursjon!

Fler eksempler

Hva printes?

```java
public static void main(String[] args) {
    siHei(antGanger:3);
}


static void siHei(int antGanger) {
    if (antGanger <= 0) {
        return;
    }
    System.out.println(x:"Hei!");
    siHei(antGanger);
}
```

Hva printes?

```
public static
    siHei(antG
}


static void si
    if (antGar
        return
    }
    System.out
    siHei(antG
}
```

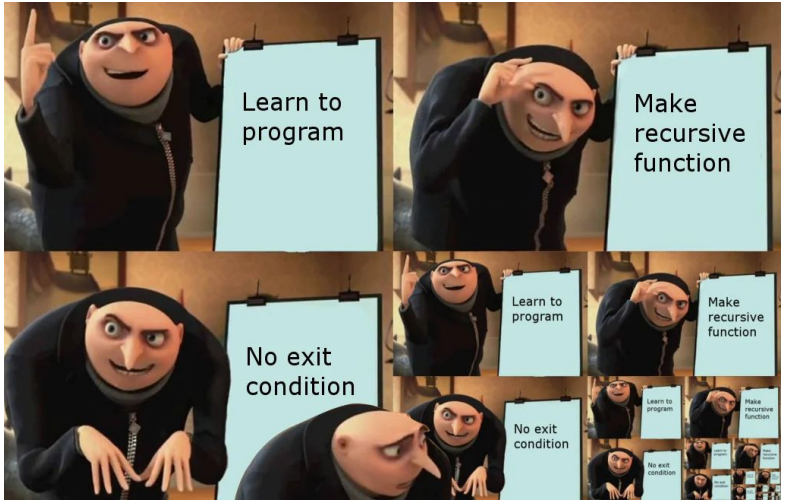
TERMINAL

```
Hei!
Hei!
Hei!
Hei!
Exception in thread "main" java.lang.StackOverflowError
        at java.base/sun.nio.cs.StreamEncoder.flushBuffer(StreamEncoder.java:112)
        at java.base/java.io.OutputStreamWriter.flushBuffer(OutputStreamWriter.java:192)
        at java.base/java.io.PrintStream.implWriteln(PrintStream.java:849)
        at java.base/java.io.PrintStream.writeln(PrintStream.java:826)
        at java.base/java.io.PrintStream.println(PrintStream.java:1168)
        at TestRekursjon.siHei(TestRekursjon.java:11)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
        at TestRekursjon.siHei(TestRekursjon.java:12)
```

StackOverflow Error!

Viktig - unngå “evige” metodekall (*stack overflow*)

- Problemet må bli “lettere” / nærmere basistilfellet for hvert rekursivt kall!
- Hvis aldri når basistilfellet → metoden kalle på seg selv i evighet (til stacken er full)




```java
static void siHei(int antGanger) {
    if (antGanger <= 0) {
        return;
    }
    System.out.println(x:"Hei!");
    siHei(antGanger);
}
```

```java
static void siHei(int antGanger) {
    if (antGanger <= 0) {
        return;
    }
    System.out.println(x:"Hei!");
    siHei(antGanger-1);
}
```

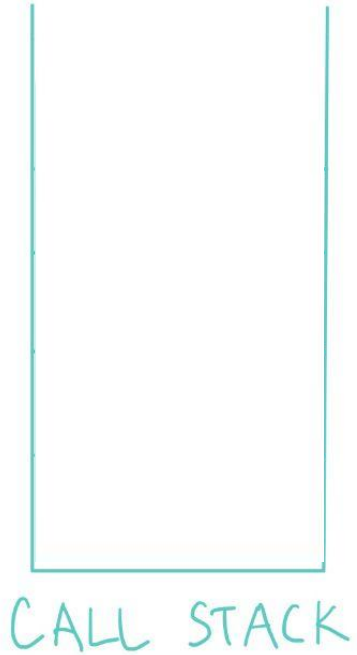
Stacken og rekursjon

Hva skjer når vi kaller på en rekursiv metode?

- Alle metodekall legges på en “call stack” (stabel i innlevering 2)

- Kaller på metode → *push* på stacken
- Returnerer fra metode → *pop* av stacken

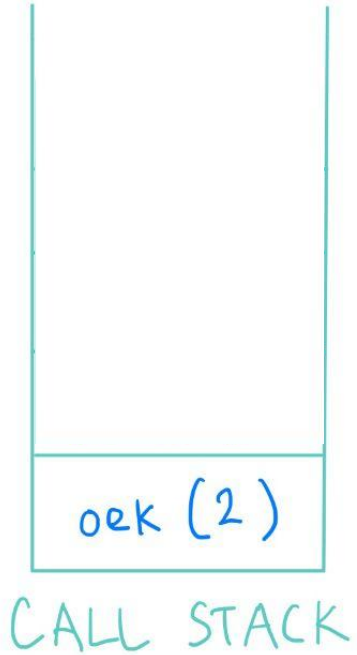
Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

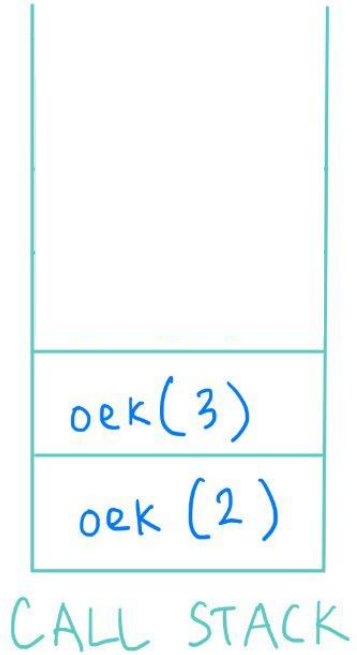
Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

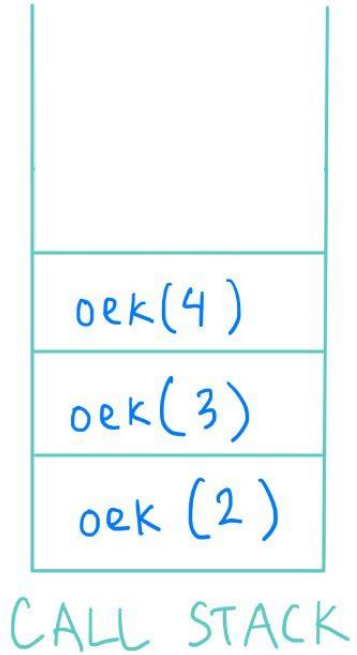
Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

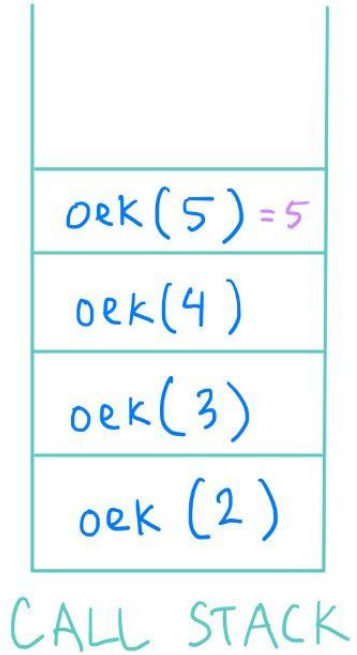
Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

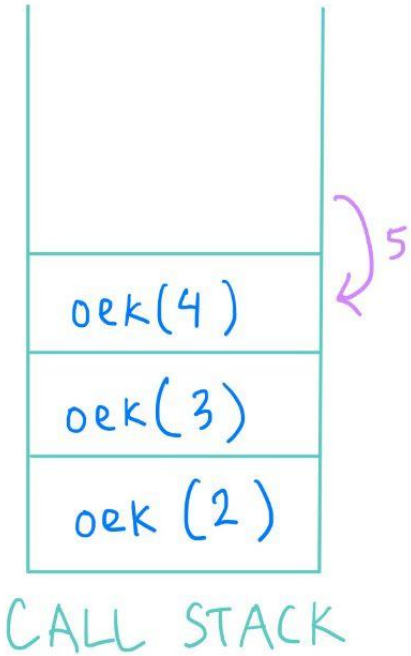
Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

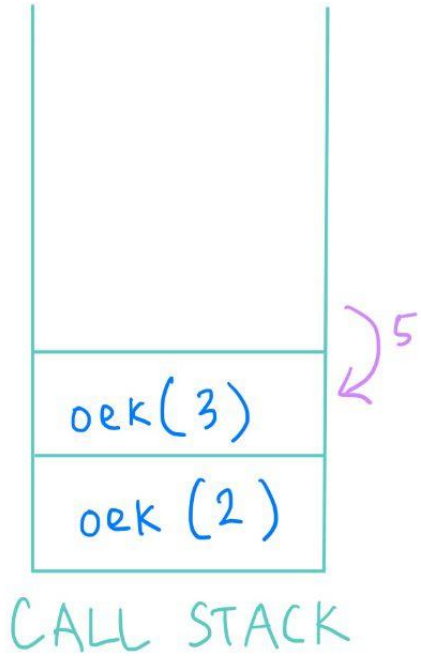
Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

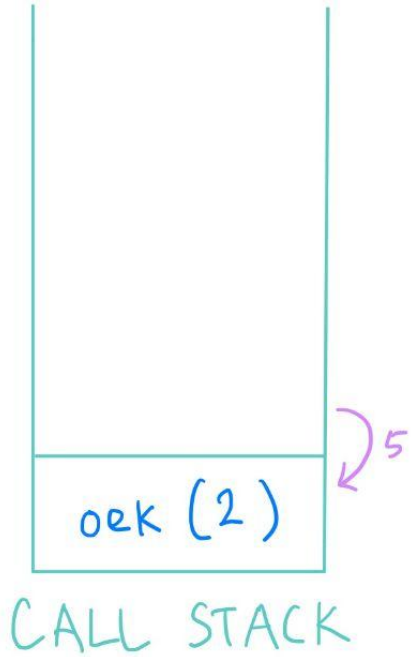
Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

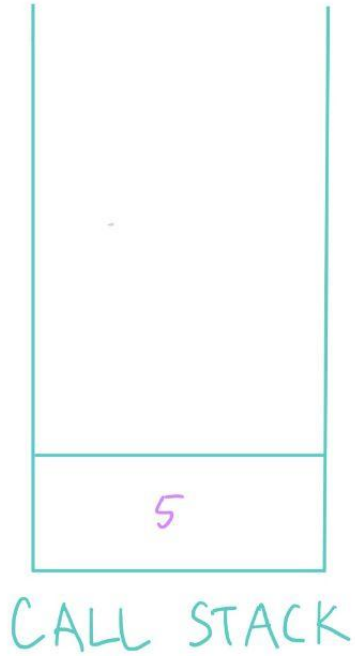
Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

Hva skjer når vi kaller på en rekursiv metode?




```java
public static void main(String[] args) {
    oek(tall:2);
}


static int oek(int tall) {
    if (tall >= 5) {
    return tall;
    }
    return oek(tall + 1);
}
```

Kodeeksempel / oppgave

Fakultets-funksjon med rekursjon

Kodeeksempel / oppgave

- Hvordan kan vi skrive en rekursiv funksjon som regner ut fakultetet av et tall (n!)
  e.g. 5! = 5*4*3*2*1

$$3! = 1 \cdot 2 \cdot 3 = 6$$

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

$$7! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 = 5\,040$$

Spørsmål som kan hjelpe:

- Hva er basistilfellet (når skal vi “stoppe”)?
- Hva skal vi gjøre i hvert rekursive kall for å komme nærmere basistilfellet?
- Hvordan kan vi bruke resultatet fra andre kall til å komme nærmere svaret?

Tegne hva som skjer i rekursjon

```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```

Hva returnerer fakultet(5)?

Steg 1: Tegn opp helt til du kommer til basistilfellet

fakultet(5)

```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```

fakultet(5)

↳ tall = 5

```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```

fakultet(5)

↳ tall = 5

return 5 · fakultet(4)

```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```

```
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```

fakultet(5)

↳ tall = 5

return 5 · fakultet(4)

↳ tall = 4

return 4 · fakultet(3)

```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```

fakultet(5)

↳ tall = 5

return 5 · fakultet(4)

↳ tall = 4

return 4 · fakultet(3)

↳ tall = 3

return 3 · fakultet(2)

```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```

fakultet(5)
↳ tall = 5
return 5 · fakultet(4)
↳ tall = 4
return 4 · fakultet(3)
↳ tall = 3
return 3 · fakultet(2)
↳ tall = 2
return 2 · fakultet(1)

```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

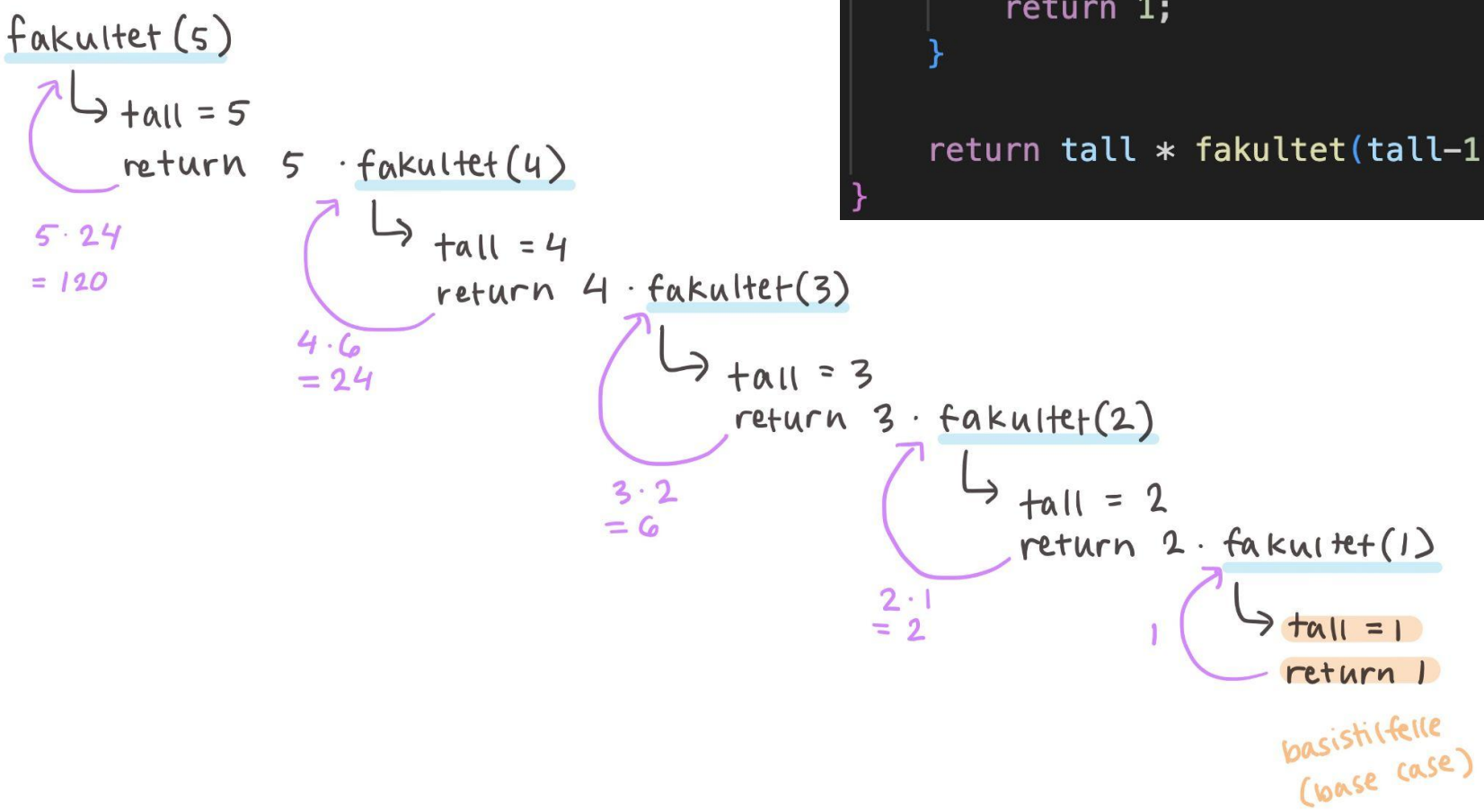
    return tall * fakultet(tall-1);
}
```

fakultet(5)
↳ tall = 5
return 5 · fakultet(4)
↳ tall = 4
return 4 · fakultet(3)
↳ tall = 3
return 3 · fakultet(2)
↳ tall = 2
return 2 · fakultet(1)
↳ tall = 1
return 1

basistilfelle
(base case)

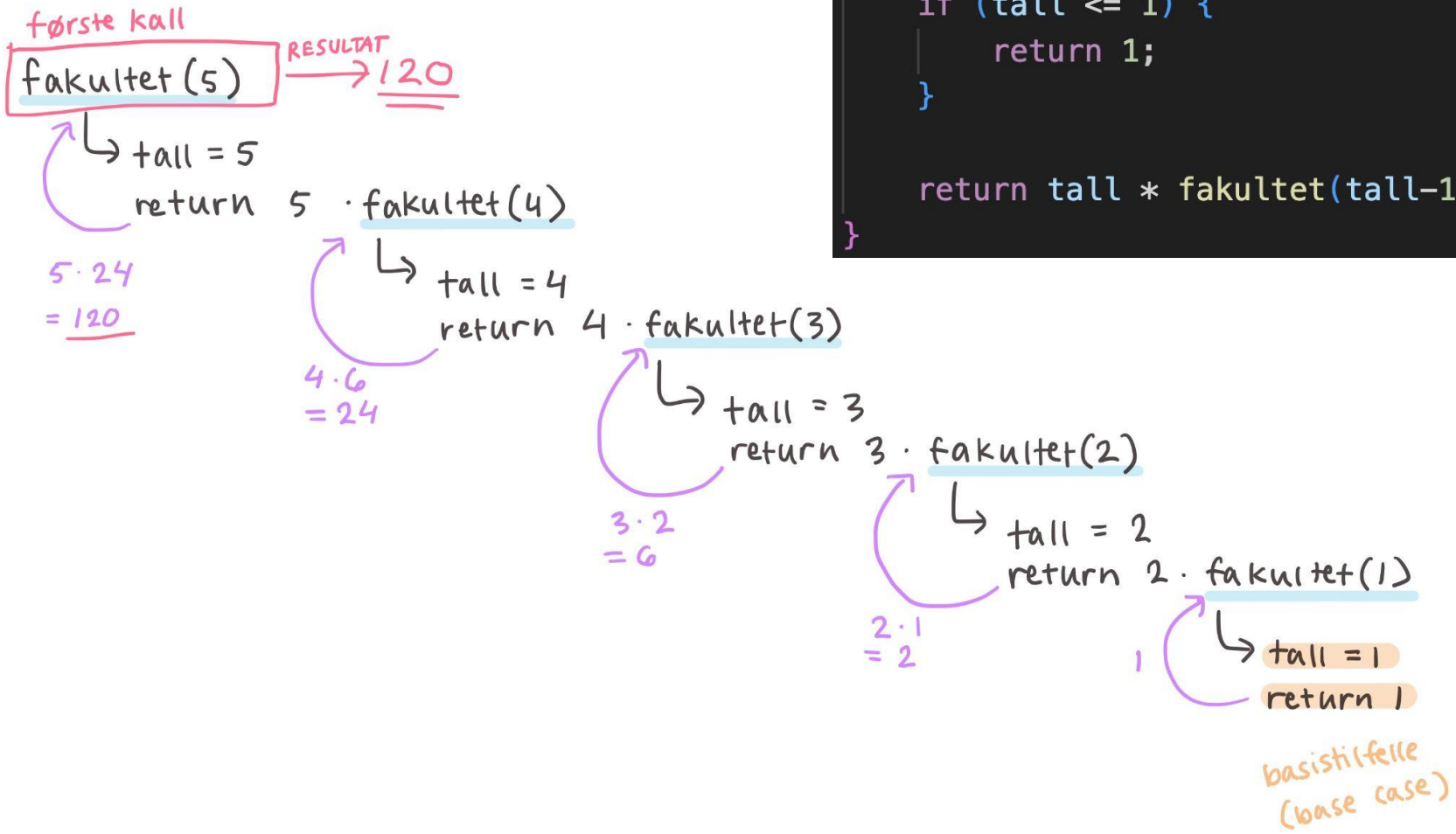
```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```



```java
public static int fakultet(int tall) {
    if (tall <= 1) {
        return 1;
    }

    return tall * fakultet(tall-1);
}
```



Jobb med oppgaver!

IN1010 Emnesiden → Grupper →
Gruppe 2 → Uke7