



IN1010 - Seminar 12

- Rekursjon
- 

Praktisk

Husk å sjekke emnesiden regelmessig

Undervisningstilbud:

- <https://www.uio.no/studier/emner/matnat/ifi/IN1010/v21/undervisningstilbud/>
- Jeg har konkrete spørsmål/problemer med min kode -> Labtime!
- Jeg vil ha mer liveprogrammering -> Plenumstime!
- Jeg vil jobbe med andre (og kanskje en kjapp recap av forelesning) -> Gruppetime!
- Jeg vil ha en recap av de vanskeligste konseptene fra forelesning -> Repetisjonsgruppe!

Oblig 5 er nå publisert (ca. 3 uker på dere). HUSK at denne skal inn rett etter påsken, som betyr at det ikke vil være noen labtimer/gruppetimer og at gruppelærere/faglærere ikke er tilgjengelig den siste uken på mattermost. Kan derfor være lurt å prøve å bli ferdig før ferien stater.

Send Marlen en direkte melding i chatten

Vil du jobbe sammen med noen andre ? (ja /nei)

Hvis du har noen ønsker på hvem du vil jobbe med, så send det i samme melding

Svar gjerne også om svaret skulle være nei

Repetisjon denne uka

Hva er rekursjon?

Vi bryter opp komplekse oppgaver i mindre/enklere oppgaver

Rekursjon: samme operasjon flere ganger

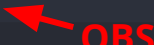
Hvordan implementere rekursjon?

1. Basis case
 - a. Vi må passe på at vi ikke får uendelig rekursjon!
 - b. Basis caset gjør ikke et rekursivt kall!
2. Hvert rekursive kall må gjøre beregningen litt enklere
 - a. Slik at vi til slutt treffer basis caset!

TIPS: Hvis du synes rekursjon er litt vanskelig så **LES BOKA KAP 13**, det gjelder å få litt feelingen for rekursjon. Men som regel er det mye enklere enn man kanskje tror! Et annet tips er å tegne hva vi vil at skal skje!

Løkke vs rekursjon

```
15 class EnkelIterasjon{  
16     public static void main(String[] args) {  
17         System.out.println("Iterasjon:");  
18         skrivTallIterasjon(5);  
19     }  
20     public static void skrivTallIterasjon(int n){  
21         for(int i = n; i >= 0; i--){  
22             System.out.println(i);  
23         }  
24     }  
}
```



Hva printes her?
Send Johanna en
direktemelding i chatten.

Løkke vs rekursjon

```
15 class EnkelIterasjon{  
16   public static void main(String[] args) {  
17     System.out.println("Iterasjon:");  
18     skrivTallIterasjon(5);  
19   }  
20   public static void skrivTallIterasjon(int n){  
21     for(int i = n; i >= 0; i--){  
22       System.out.println(i);  
23     }  
24   }  
}
```

```
jonbon@jons-macbook-pro uke12 % java EnkelIterasjon  
Iterasjon:
```

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

```
1 class EnkelRekursjon {  
2   public static void main(String[] args) {  
3     System.out.println("Rekursjon:");  
4     skrivTallRekursjon(5);  
5   }  
6   public static void skrivTallRekursjon(int n){  
7     if (n < 0) {  
8       return;  
9     }  
10    System.out.println(n);  
11    skrivTallRekursjon(n-1);  
12  }
```

1. Basis case
 - a. Vi må passe på at vi ikke får uendelig rekursjon!
 - b. Basis caset gjør ikke et rekursivt kall!
2. Hvert rekursive kall må gjøre beregningen litt enklere
 - a. Slik at vi til slutt treffer basis caset!

Løkke vs rekursjon

```
15 class EnkelIterasjon{  
16   public static void main(String[] args) {  
17     System.out.println("Iterasjon:");  
18     skrivTallIterasjon(5);  
19   }  
20   public static void skrivTallIterasjon(int n){  
21     for(int i = n; i >= 0; i--){  
22       System.out.println(i);  
23     }  
24   }  
}
```

```
jonbon@jons-macbook-pro uke12 % java EnkelIterasjon  
Iterasjon:  
5  
4  
3  
2  
1  
0
```

```
1 class EnkelRekursjon {  
2   public static void main(String[] args) {  
3     System.out.println("Rekursjon:");  
4     skrivTallRekursjon(5);  
5   }  
6   public static void skrivTallRekursjon(int n){  
7     if (n < 0) {  
8       return; Basis case  
9     }  
10    System.out.println(n);  
11    skrivTallRekursjon(n-1);  
12  }
```

1. Basis case
 - a. Vi må passe på at vi ikke får uendelig rekursjon!
 - b. Basis caset gjør ikke et rekursivt kall!
2. Hvert rekursive kall må gjøre beregningen litt enklere
 - a. Slik at vi til slutt treffer basis caset!

Løkke vs rekursjon

```
15 class EnkelIterasjon{  
16   public static void main(String[] args) {  
17     System.out.println("Iterasjon:");  
18     skrivTallIterasjon(5);  
19   }  
20   public static void skrivTallIterasjon(int n){  
21     for(int i = n; i >= 0; i--){  
22       System.out.println(i);  
23     }  
24   }
```

```
jonbon@jons-macbook-pro uke12 % java EnkelIterasjon  
Iterasjon:  
5  
4  
3  
2  
1  
0
```

```
1 class EnkelRekursjon {  
2   public static void main(String[] args) {  
3     System.out.println("Rekursjon:");  
4     skrivTallRekursjon(5);  
5   }  
6   public static void skrivTallRekursjon(int n){  
7     if (n < 0) {  
8       return;  
9     }  
10    System.out.println(n);  
11    skrivTallRekursjon(n-1);  
12  }
```

Enklere kall hver gang

1. Basis case
 - a. Vi må passe på at vi ikke får uendelig rekursjon!
 - b. Basis caset gjør ikke et rekursivt kall!
2. Hvert rekursive kall må gjøre beregningen litt enklere
 - a. Slik at vi til slutt treffer basis caset!

Løkke vs rekursjon

```
15 class EnkelIterasjon{  
16   public static void main(String[] args) {  
17     System.out.println("Iterasjon:");  
18     skrivTallIterasjon(5);  
19   }  
20   public static void skrivTallIterasjon(int n){  
21     for(int i = n; i >= 0; i--){  
22       System.out.println(i);  
23     }  
24   }  
}
```

```
jonbon@jons-macbook-pro uke12 % java EnkelIterasjon  
Iterasjon:  
5  
4  
3  
2  
1  
0
```

```
1 class EnkelRekursjon {  
2   public static void main(String[] args) {  
3     System.out.println("Rekursjon:");  
4     skrivTallRekursjon(5);  
5   }  
6   public static void skrivTallRekursjon(int n){  
7     if (n < 0) {  
8       return;  
9     }  
10    System.out.println(n);  
11    skrivTallRekursjon(n-1);  
12  }
```

Hva printes her?
Send Johanna en
direktemelding i chatten.

Løkke vs rekursjon

```
15 class EnkelIterasjon{  
16   public static void main(String[] args) {  
17     System.out.println("Iterasjon:");  
18     skrivTallIterasjon(5);  
19   }  
20   public static void skrivTallIterasjon(int n){  
21     for(int i = n; i >= 0; i--){  
22       System.out.println(i);  
23     }  
24   }  
}
```

```
jonbon@jons-macbook-pro uke12 % java EnkelIterasjon  
Iterasjon:
```

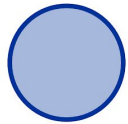
```
5  
4  
3  
2  
1  
0
```

```
1 class EnkelRekursjon {  
2   public static void main(String[] args) {  
3     System.out.println("Rekursjon:");  
4     skrivTallRekursjon(5);  
5   }  
6   public static void skrivTallRekursjon(int n){  
7     if (n < 0) {  
8       return;  
9     }  
10    System.out.println(n);  
11    skrivTallRekursjon(n-1);  
12  }
```

```
jonbon@jons-macbook-pro uke12 % java EnkelRekursjon  
Rekursjon:
```

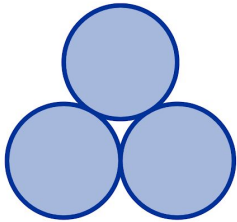
```
5  
4  
3  
2  
1  
0
```

Eksempel: rekursjon trekantall



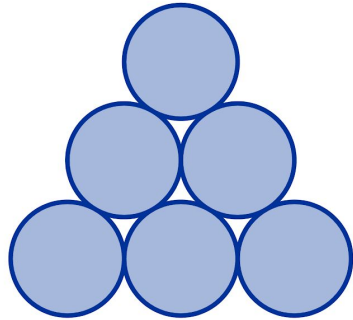
Figur 1

1



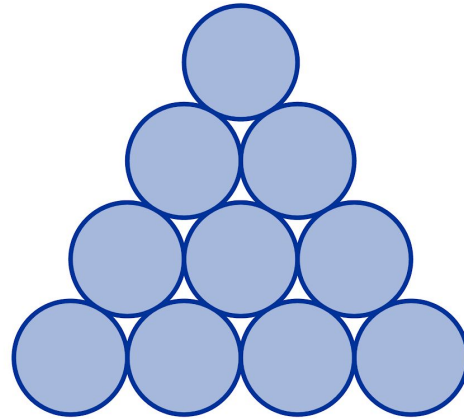
Figur 2

$1+2=3$



Figur 3

$3+3=6$



Figur 4

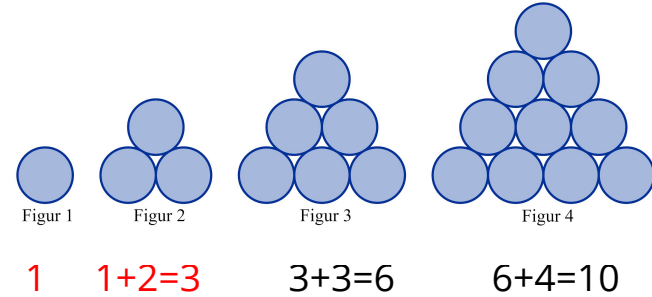
$6+4=10$

Eksempel: rekursjon trekantttall

For å regne ut trekantttallet til trekant **2** trenger vi bare vite trekantttallet til trekant 1.

(trekantttallet til trekant 1) + **2** = x

$$1 + 2 = 3$$

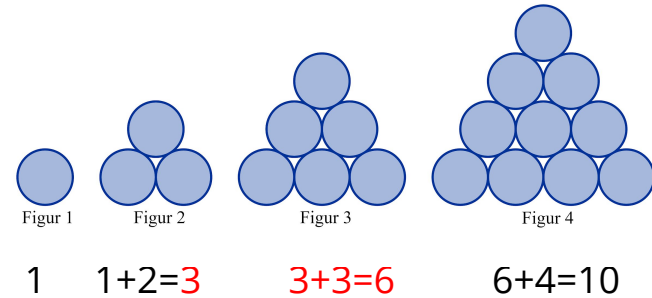


Eksempel: rekursjon trekantttall

For å regne ut trekantttallet til trekant **3**
trenger vi bare vite trekantttallet til trekant 2.

(trekantttallet til trekant 2) + **3** = x

$$3 + 3 = 6$$

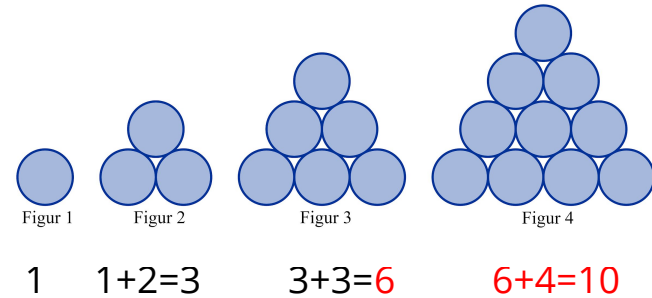


Eksempel: rekursjon trekantttall

For å regne ut trekantttallet til trekant **4** trenger vi bare vite trekantttallet til trekant 3.

(trekantttallet til trekant 3) + **4** = x

$6 + 4 = 10$

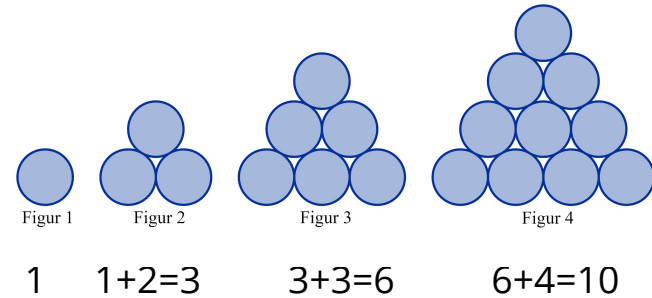


Eksempel: rekursjon trekantall

For å regne ut trekantallet til trekant y trenger vi bare vite trekantallet til trekanten før y .

Siden vi vet at trekantallet til trekant $1 = 1$ så kan vi regne oss fram til hvilket som helst trekantall.

Send Johanna melding i chatten: hva er trekantallet til trekant 6?



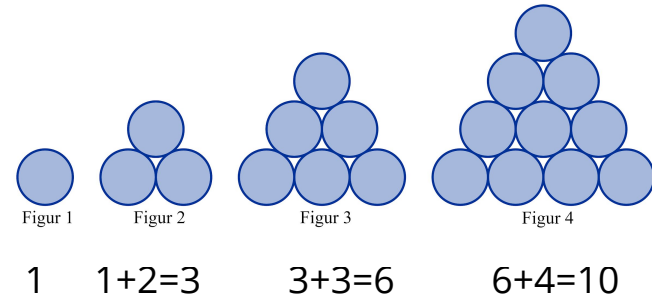
Eksempel: rekursjon trekantall

For å regne ut trekantallet til trekant y trenger vi bare vite trekantallet til trekanten før y .

Siden vi vet at trekantallet til trekant $1 = 1$ så kan vi regne oss fram til hvilket som helst trekantall.

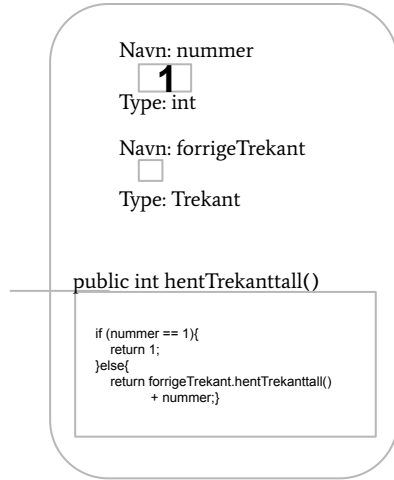
Send Johanna melding i chatten: hva er trekantallet til trekant 6?

$1 + 2 = 3$, $3 + 3 = 6$, $6 + 4 = 10$, $10 + 5 = 15$,
 $15 + 6 = 21$



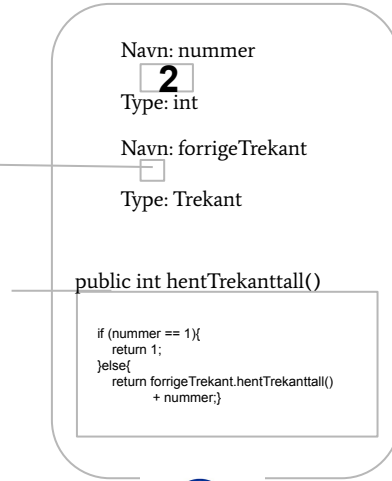
Rekursjon trekantall

Trekant-objekt



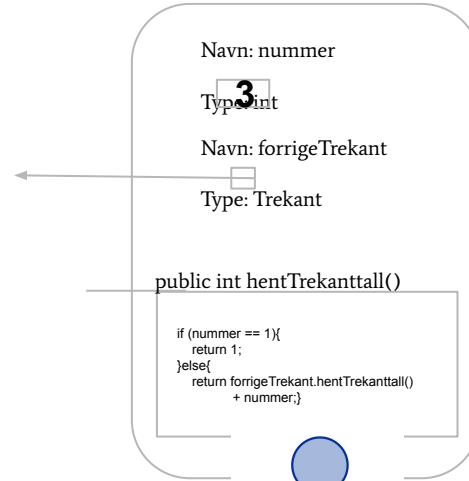
Figur 1

Trekant-objekt



Figur 2

Trekant-objekt



Figur 3

Rekursjon trekantall

Trekant-objekt

Navn: nummer

1

Type: int

Navn: forrigeTrekant

Type: Trekant

public int hentTrekantall()

```
if (nummer == 1){  
    return 1;  
}else{  
    return forrigeTrekant.hentTrekantall()  
        + nummer;}  
}
```



Figur 1

Trekant-objekt

Navn: nummer

2

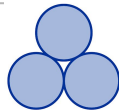
Type: int

Navn: forrigeTrekant

Type: Trekant

public int hentTrekantall()

```
if (nummer == 1){  
    return 1;  
}else{  
    return forrigeTrekant.hentTrekantall()  
        + nummer;}  
}
```



Figur 2

Trekant-objekt

Navn: nummer

3

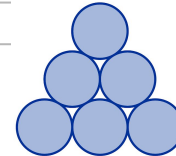
Type: int

Navn: forrigeTrekant

Type: Trekant

public int hentTrekantall()

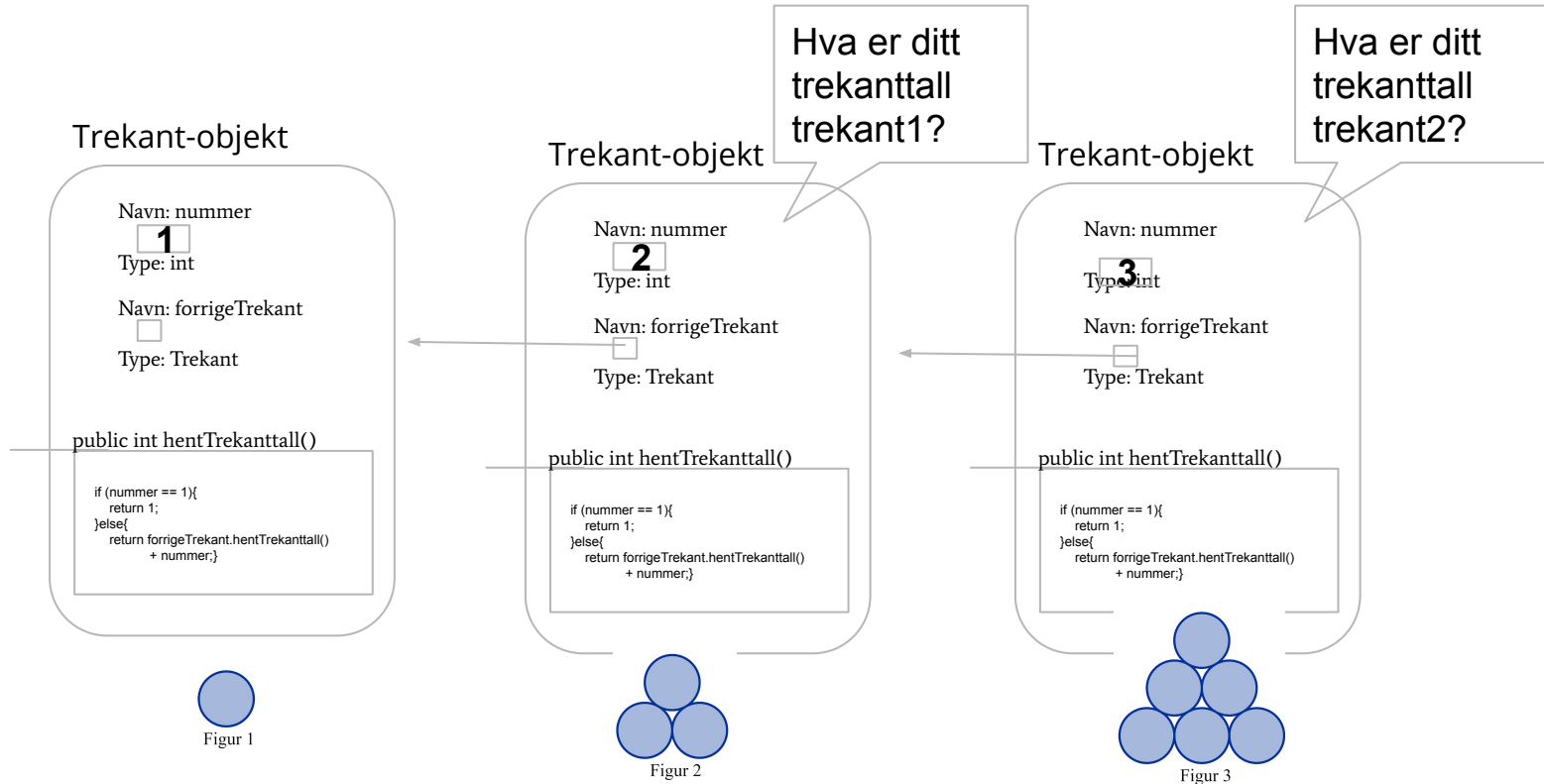
```
if (nummer == 1){  
    return 1;  
}else{  
    return forrigeTrekant.hentTrekantall()  
        + nummer;}  
}
```



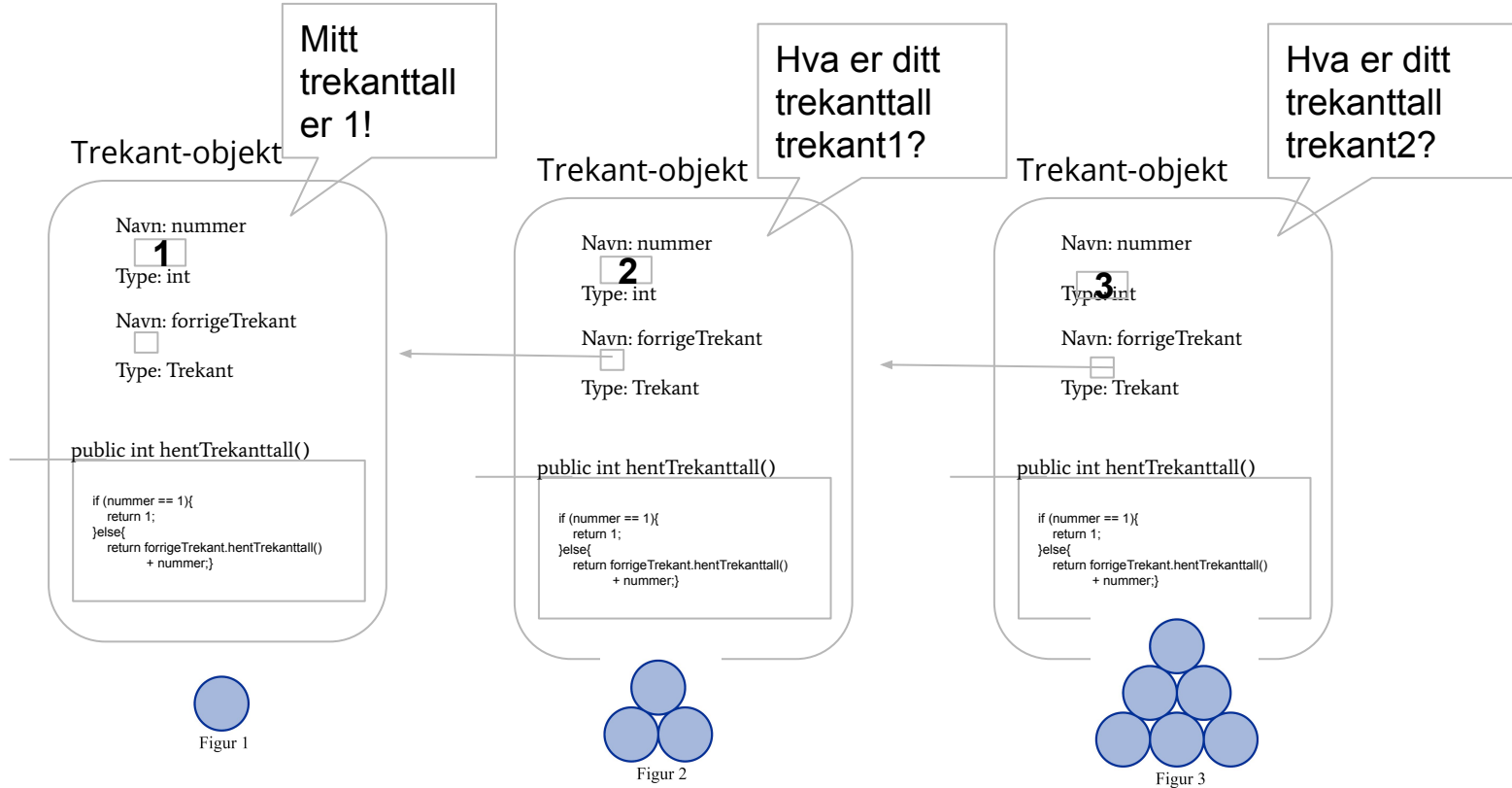
Figur 3

Hva er ditt
trekantall
trekant2?

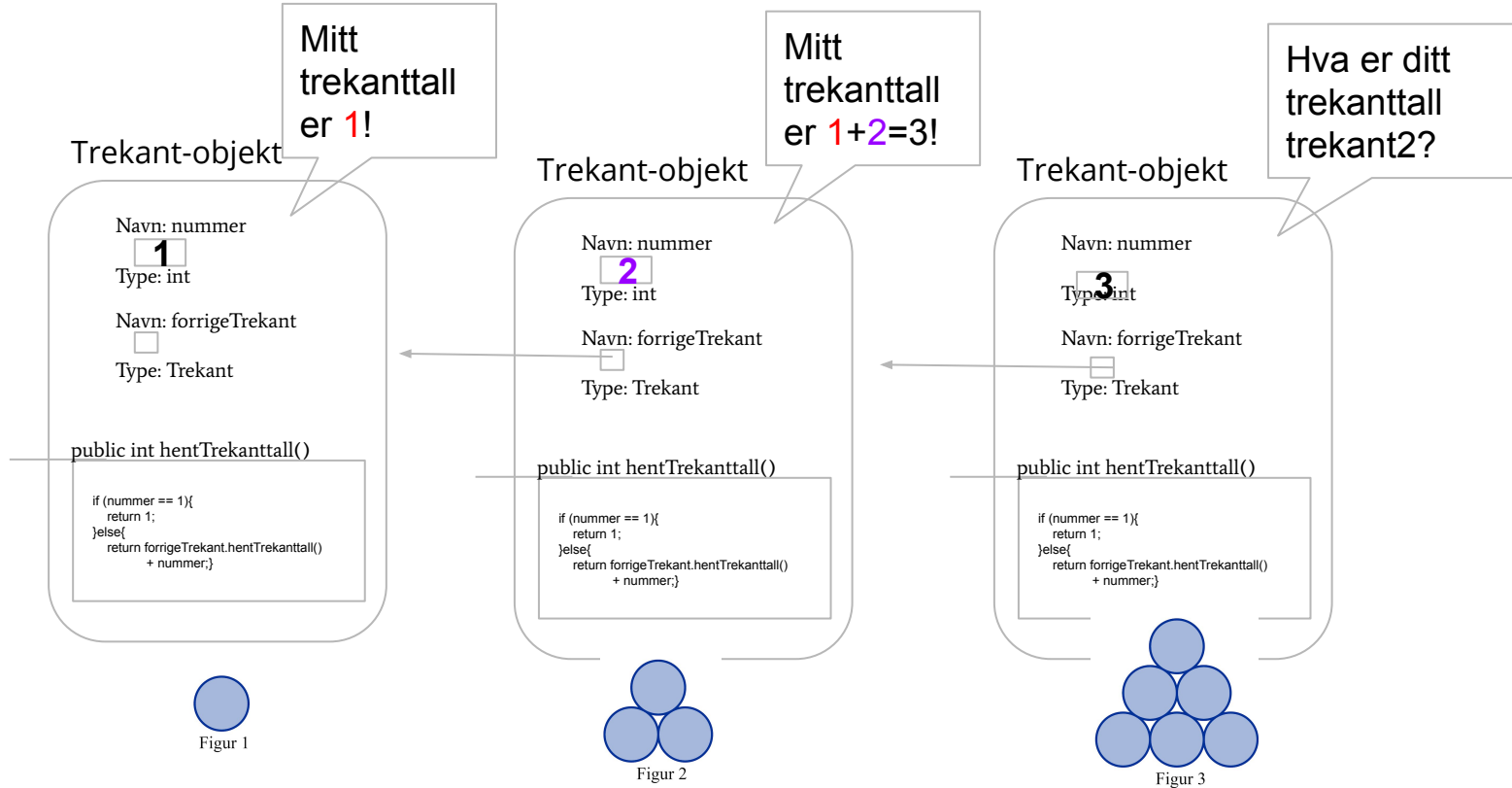
Rekursjon trekantall



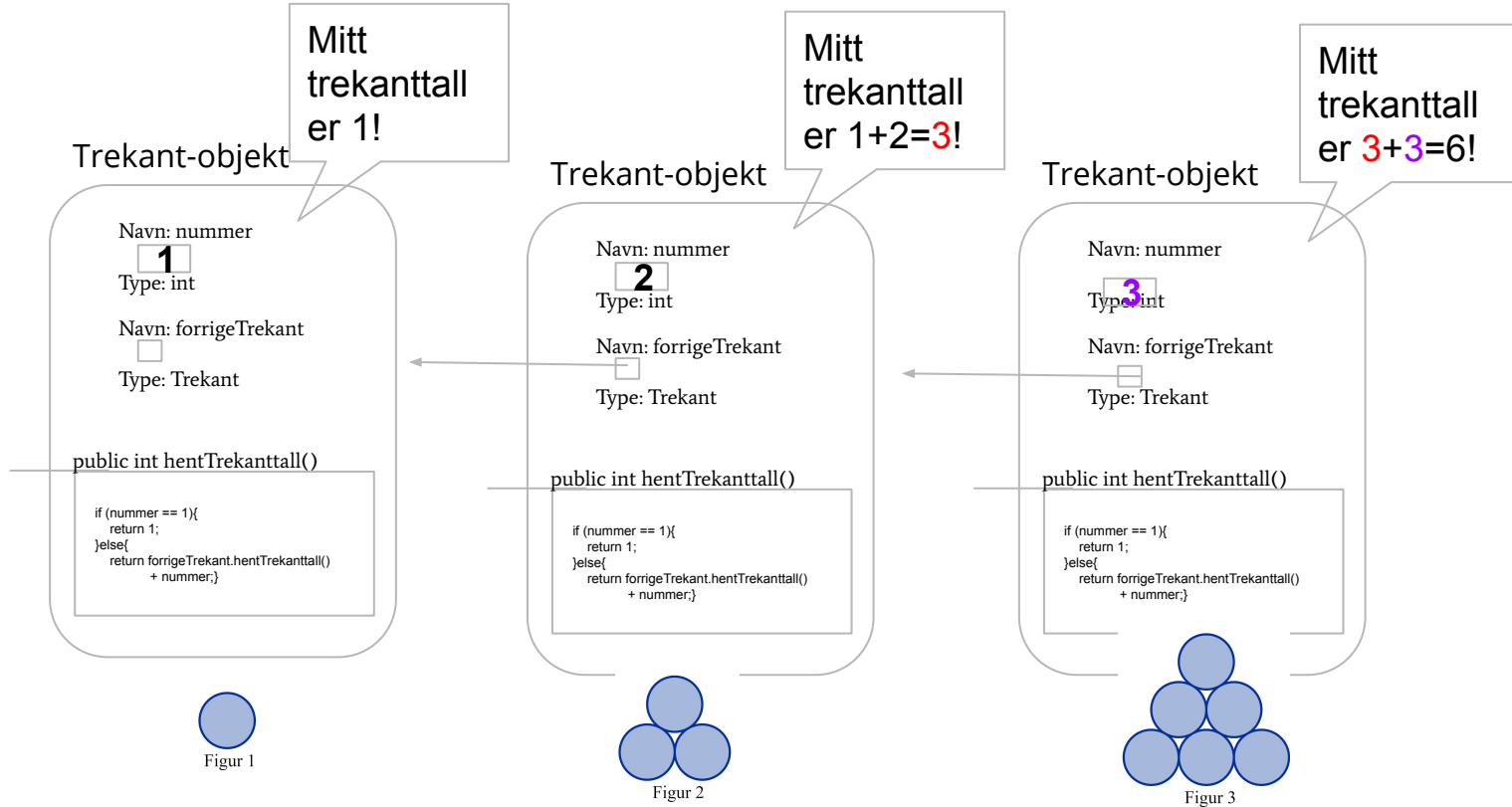
Rekursjon trekantall



Rekursjon trekantall

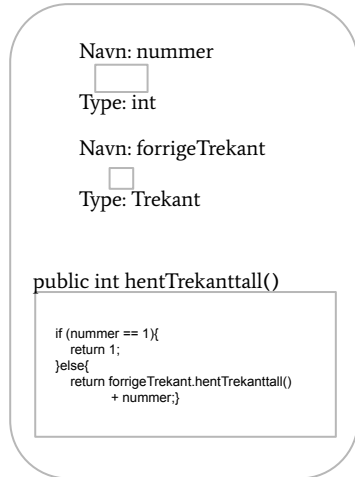


Rekursjon trekantall



Rekursjon trekanttall

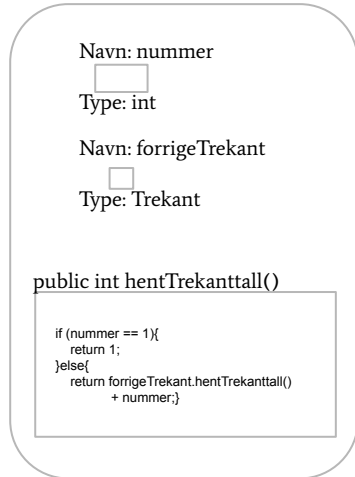
Trekant-objekt



```
1 class Trekant{
2     · private int nummer;
3     · private Trekant forrigeTrekant;
4     ↵
5     · public Trekant(int nummer, Trekant forrigeTrekant){
6     · · · this.nummer = nummer;
7     · · · this.forrigeTrekant = forrigeTrekant;
8     · }
9     · public int hentTrekanttall(){
10    · · · if (nummer == 1){
11    · · · · · return 1;
12    · · · }else{
13    · · · · · return forrigeTrekant.hentTrekanttall() + nummer;
14    · · · }
```

Rekursjon trekantall

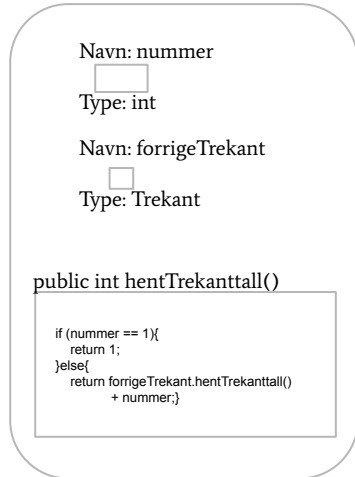
Trekant-objekt



```
1 class Trekant{  
2     · private int nummer;  
3     · private Trekant forrigeTrekant;  
4     ·  
5     · public Trekant(int nummer, Trekant forrigeTrekant){  
6         ··· this.nummer = nummer;  
7         ··· this.forrigeTrekant = forrigeTrekant;  
8     · }  
9     · public int hentTrekanttall(){  
10        ··· if (nummer == 1){  
11            ···· return 1; ← Basis case  
12        ··· }else{  
13            ···· return forrigeTrekant.hentTrekanttall() + nummer;  
14        ··· }  
}
```

Rekursjon trekantall

Trekant-objekt



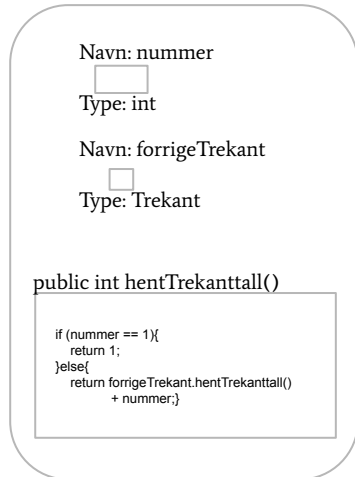
```
1 class Trekant{
2     · private int nummer;
3     · private Trekant forrigeTrekant;
4     ↵
5     · public Trekant(int nummer, Trekant forrigeTrekant){
6     ··· this.nummer = nummer;
7     ··· this.forrigeTrekant = forrigeTrekant;
8     ··}
9     · public int hentTrekantall(){
10    ··· if (nummer == 1){
11    ····· return 1;
12    ···}else{
13    ····· return forrigeTrekant.hentTrekantall() + nummer;
14    ···}
15 }
```

**Enklere
kall hver
gang**



Rekursjon trekanttall

Trekant-objekt



```
1 class Trekant{  
2     · private int nummer;  
3     · private Trekant forrigeTrekant;  
4     ↵  
5     · public Trekant(int nummer, Trekant forrigeTrekant){  
6         ··· this.nummer = nummer;  
7         ··· this.forrigeTrekant = forrigeTrekant;  
8     }  
9     · public int hentTrekanttall(){  
10        ··· if (nummer == 1){  
11            ····· return 1;  
12        }else{  
13            ····· return forrigeTrekant.hentTrekanttall() + nummer;  
14        }  
}
```

```
1 class TestTrekant{  
2     · public static void main(String[] args) {  
3         ··· Trekant trekant1 = new Trekant(1, null);  
4         ··· Trekant trekant2 = new Trekant(2, trekant1);  
5         ··· Trekant trekant3 = new Trekant(3, trekant2);  
6         ··· Trekant trekant4 = new Trekant(4, trekant3);  
7         ··· Trekant trekant5 = new Trekant(5, trekant4);  
8         ··· Trekant trekant6 = new Trekant(6, trekant5);  
9     }  
10    ··· System.out.println(trekant6.hentTrekanttall());  
11 }
```

Kjøre koden

```
jonbon@jons-macbook-pro uke12 % java TestTrekant  
21
```

```
1 class Trekant{  
2     private int nummer;  
3     private Trekant forrigeTrekant;  
4  
5     public Trekant(int nummer, Trekant forrigeTrekant){  
6         this.nummer = nummer;  
7         this.forrigeTrekant = forrigeTrekant;  
8     }  
9     public int hentTrekanttall(){  
10        if (nummer == 1){  
11            return 1;  
12        }else{  
13            return forrigeTrekant.hentTrekanttall() + nummer;  
14        }  
}
```

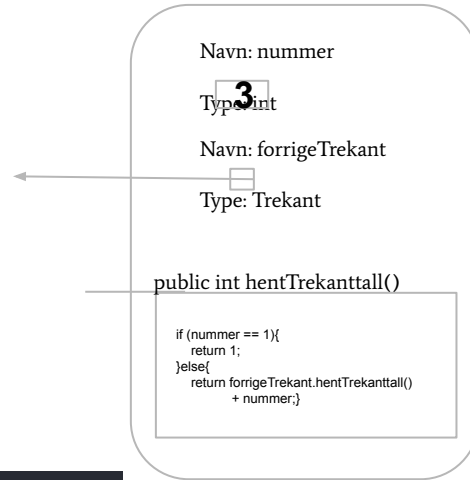
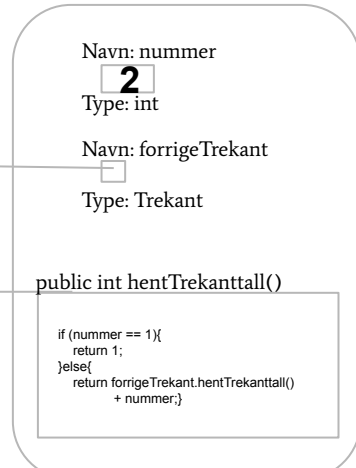
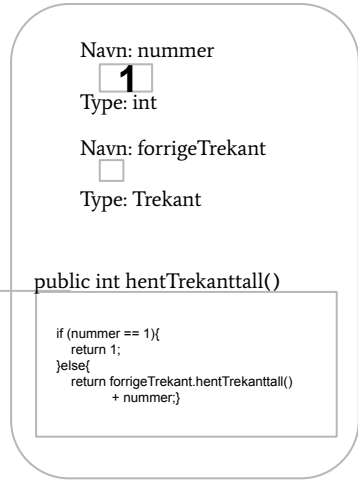
```
1 class TestTrekant{  
2     public static void main(String[] args) {  
3         Trekant trekant1 = new Trekant(1, null);  
4         Trekant trekant2 = new Trekant(2, trekant1);  
5         Trekant trekant3 = new Trekant(3, trekant2);  
6         Trekant trekant4 = new Trekant(4, trekant3);  
7         Trekant trekant5 = new Trekant(5, trekant4);  
8         Trekant trekant6 = new Trekant(6, trekant5);  
9  
10        System.out.println(trekant6.hentTrekanttall());  
11    }  
}
```

Rekursjon trekantall

Trekant-objekt

Trekant-objekt

Trekant-objekt



```
9 public int hentTrekantall(){  
0     if (nummer == 1){  
1         return 1;  
2     }else{  
3         return forrigeTrekant.hentTrekantall() + nummer;  
4     }  
5 }
```

Ris, ros, forslag ?

<https://nettskjema.no/a/180345>

Diskusjon breakout rooms

Repetisjon

- Hvorfor er det lurt med tråder?
- Repeter countdownlatch, cyclic barrier og join join.
- Hva er monitor-design-prinsippet?

Denne uken

- Hva er rekursjon?
- Ukesoppgaver

Breakoutrooms

1. Slå på kamera og ha en presentasjonsrunde
2. Jobb sammen med ukesoppgavene, de ligger på emnesiden -> grupper
 - a. Enten ved at én deler skjerm eller med codecollab.io/
 - b. OBS: codecollab er gratisjeneseter som UiO ikke har avtale med, sannsynligvis vil de samle data om dere. Dere kan fint løse oppgavene uten å bruke den tjenesten!
3. Bruk "ask for help"-knappen for å få hjelp 😊
4. Vi møtes her igjen for å gå gjennom oppgavene til slutt (dere bestemmer hvilke)

Jobbe med oppgaver