IN1010 - Seminar 12

- Rekursjon

Laget av: Marlen Jarholt (marleja@ifi.uio.no) og Johanna Haarseth (johannph@ifi.uio.no)

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!

Oppmøteregistrering:

https://nettskjema.no/a/188717

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 meg en direktemelding i chatten.

Løkke vs rekursjon

```java
class EnkelIterasjon{
    public static void main(String[] args) {
        System.out.println("Iterasjon:");
        skrivTallIterasjon(5);
    }
    public static void skrivTallIterasjon(int n){
        for(int i = n; i >= 0; i--){
            System.out.println(i);
        }
    }
```

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

```java
class EnkelRekursjon {
    public static void main(String[] args) {
        System.out.println("Rekursjon:");
        skrivTallRekursjon(5);
    }
    public static void skrivTallRekursjon(int n){
        if (n < 0) {
            return;
        }
        System.out.println(n);
        skrivTallRekursjon(n-1);
```

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

```java
class EnkelIterasjon{
    public static void main(String[] args) {
        System.out.println("Iterasjon:");
        skrivTallIterasjon(5);
    }
    public static void skrivTallIterasjon(int n){
        for(int i = n; i >= 0; i--){
            System.out.println(i);
        }
    }
```

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

```java
class EnkelRekursjon {
    public static void main(String[] args) {
        System.out.println("Rekursjon:");
        skrivTallRekursjon(5);
    }
    public static void skrivTallRekursjon(int n){
        if (n < 0) {
            return;
        }
        System.out.println(n);
        skrivTallRekursjon(n-1);
```

Basis case

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

```java
class EnkelIterasjon{
    public static void main(String[] args) {
        System.out.println("Iterasjon:");
        skrivTallIterasjon(5);
    }
    public static void skrivTallIterasjon(int n){
        for(int i = n; i >= 0; i--){
            System.out.println(i);
        }
    }
```

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

```java
class EnkelRekursjon {
    public static void main(String[] args) {
        System.out.println("Rekursjon:");
        skrivTallRekursjon(5);
    }
    public static void skrivTallRekursjon(int n){
        if (n < 0) {
            return;
        }
        System.out.println(n);
        skrivTallRekursjon(n-1);
```

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

```java
class EnkelIterasjon{
    public static void main(String[] args) {
        System.out.println("Iterasjon:");
        skrivTallIterasjon(5);
    }
    public static void skrivTallIterasjon(int n){
        for(int i = n; i >= 0; i--){
            System.out.println(i);
        }
    }
```

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

```java
class EnkelRekursjon {
    public static void main(String[] args) {
        System.out.println("Rekursjon:");
        skrivTallRekursjon(5);
    }
    public static void skrivTallRekursjon(int n){
        if (n < 0) {
            return;
        }
        System.out.println(n);
        skrivTallRekursjon(n-1);
```

Hva printes her?
Send meg en direktemelding i chatten.

Løkke vs rekursjon

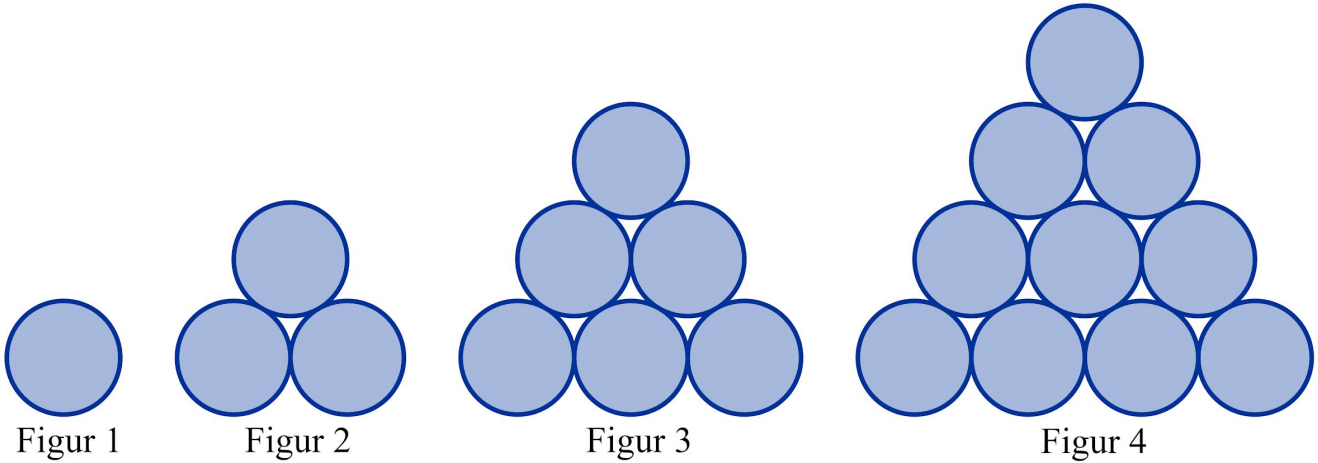
```java
class EnkelIterasjon{
    public static void main(String[] args) {
        System.out.println("Iterasjon:");
        skrivTallIterasjon(5);
    }
    public static void skrivTallIterasjon(int n){
        for(int i = n; i >= 0; i--){
            System.out.println(i);
        }
    }
```

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

```java
class EnkelRekursjon {
    public static void main(String[] args) {
        System.out.println("Rekursjon:");
        skrivTallRekursjon(5);
    }
    public static void skrivTallRekursjon(int n){
        if (n < 0) {
            return;
        }
        System.out.println(n);
        skrivTallRekursjon(n-1);
```

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

Eksempel: rekursjon trekanttall

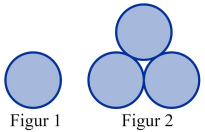


1 1+2=3 3+3=6 6+4=10

Eksempel: rekursjon trekanttall

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

(trekanttallet til trekant 1) + 2 = x
1 + 2 = 3

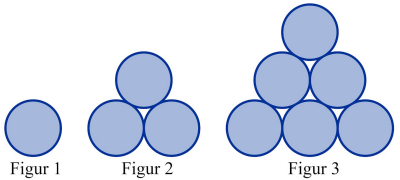


1 1+2=3

Eksempel: rekursjon trekanttall

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

(trekanttallet til trekant 2) + 3 = x
3 + 3 = 6

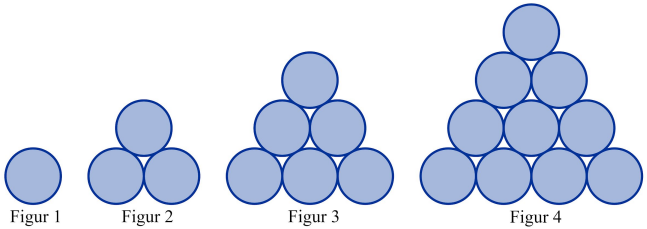


1 1+2=3 3+3=6

Eksempel: rekursjon trekanttall

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

(trekanttallet til trekant 3) + 4 = x
6 + 4 = 10



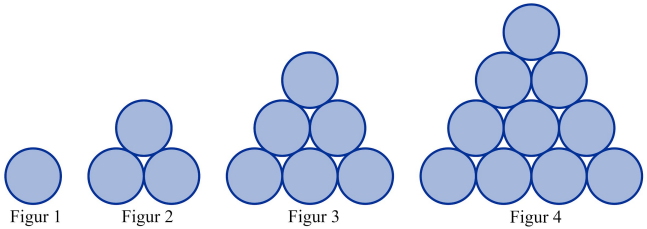
1 1+2=3 3+3=6 6+4=10

Eksempel: rekursjon trekanttall

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

Siden vi vet at trekanttallet til trekant 1 = 1 så kan vi regne oss fram til hvilket som helst trekanttall.

Send meg en melding i chatten: hva er trekanttallet til trekant 6?



1 1+2=3 3+3=6 6+4=10

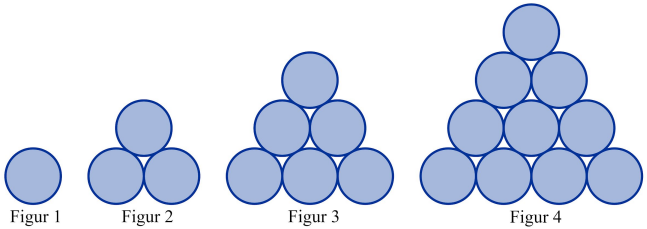
Eksempel: rekursjon trekanttall

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

Siden vi vet at trekanttallet til trekant 1 = 1 så kan vi regne oss fram til hvilket som helst trekanttall.

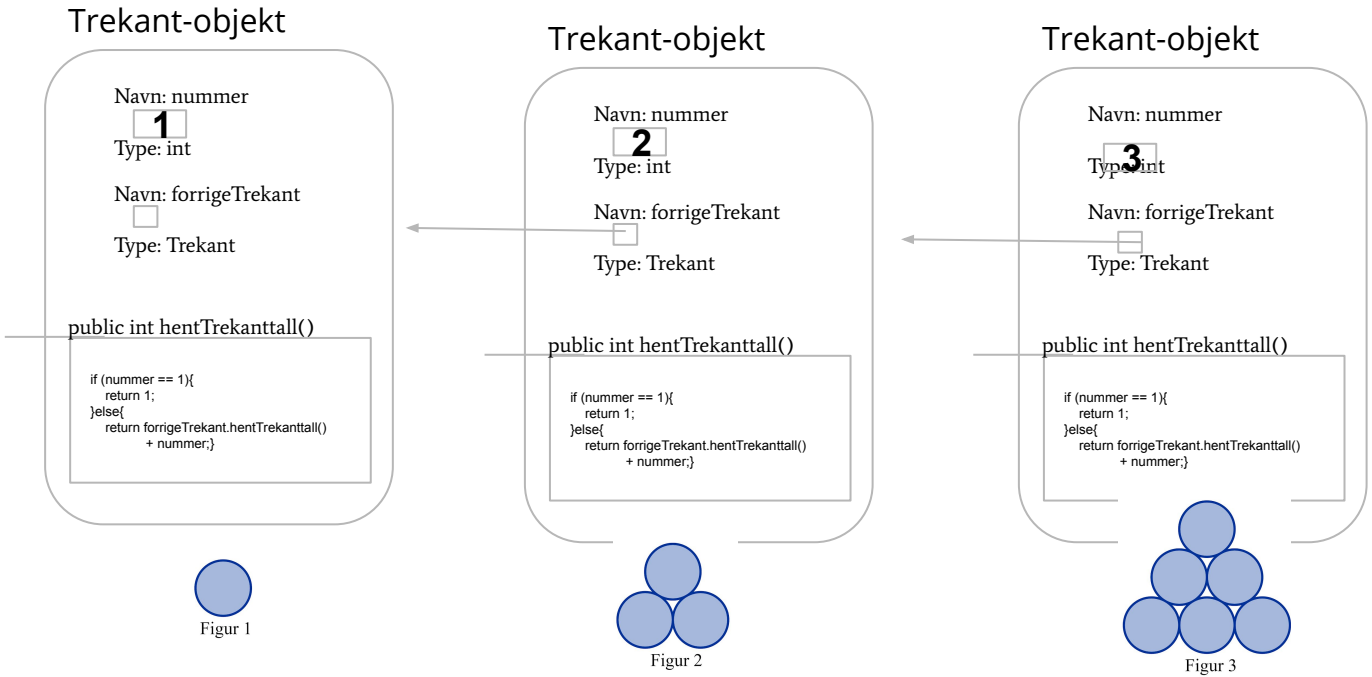
Send meg melding i chatten: hva er trekanttallet til trekant 6?

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

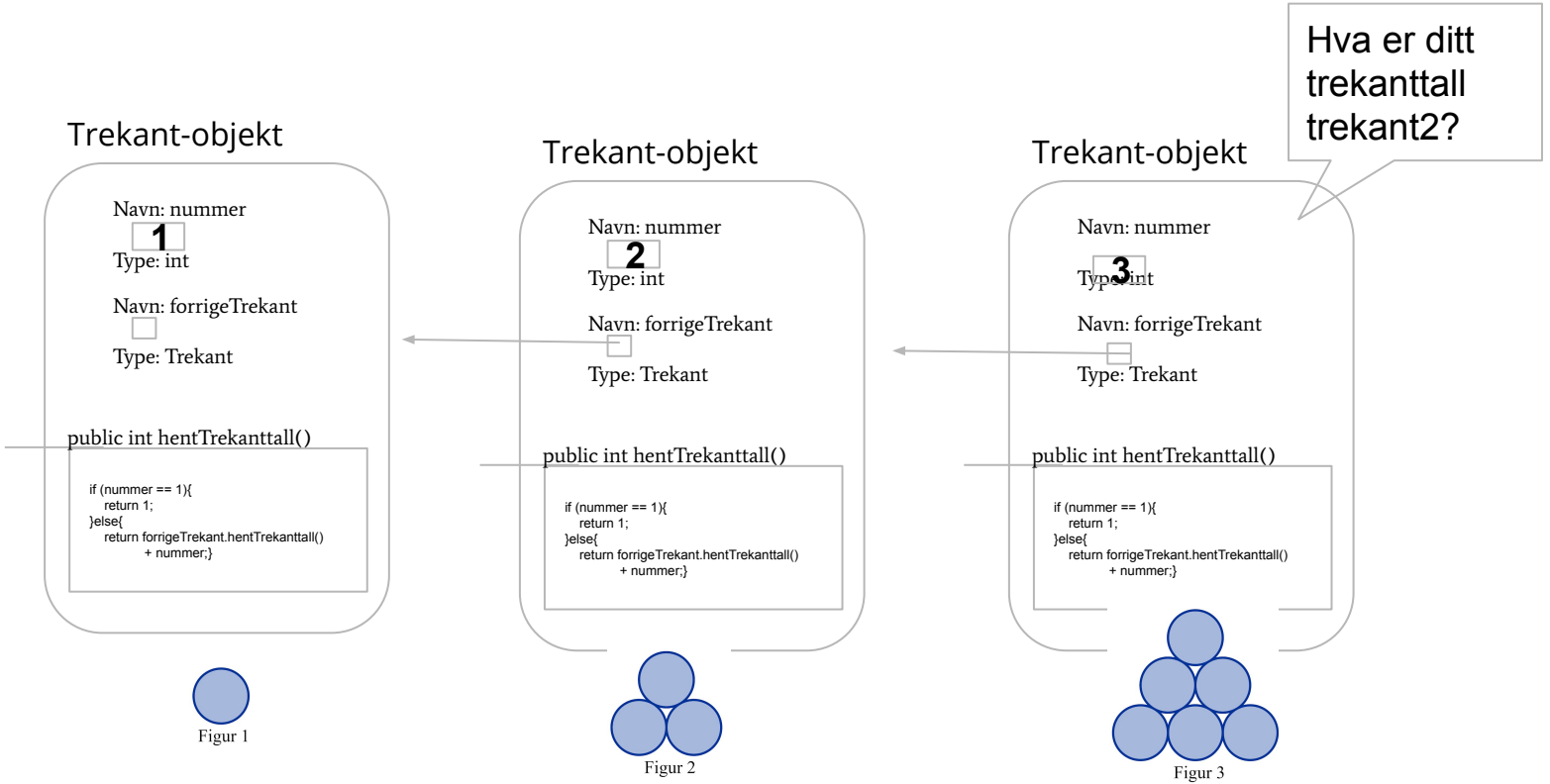


1 1+2=3 3+3=6 6+4=10

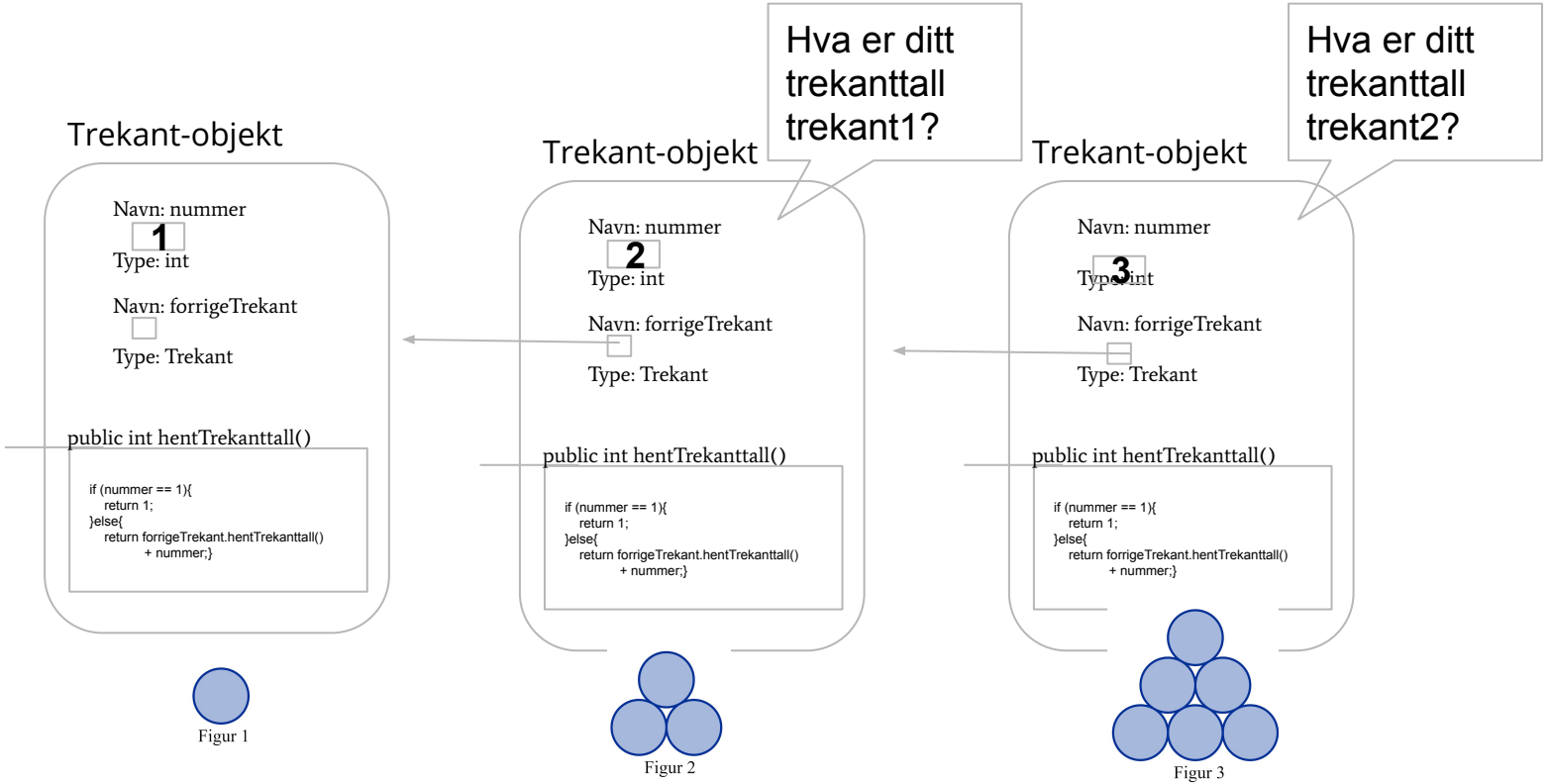
Rekursjon trekanttall



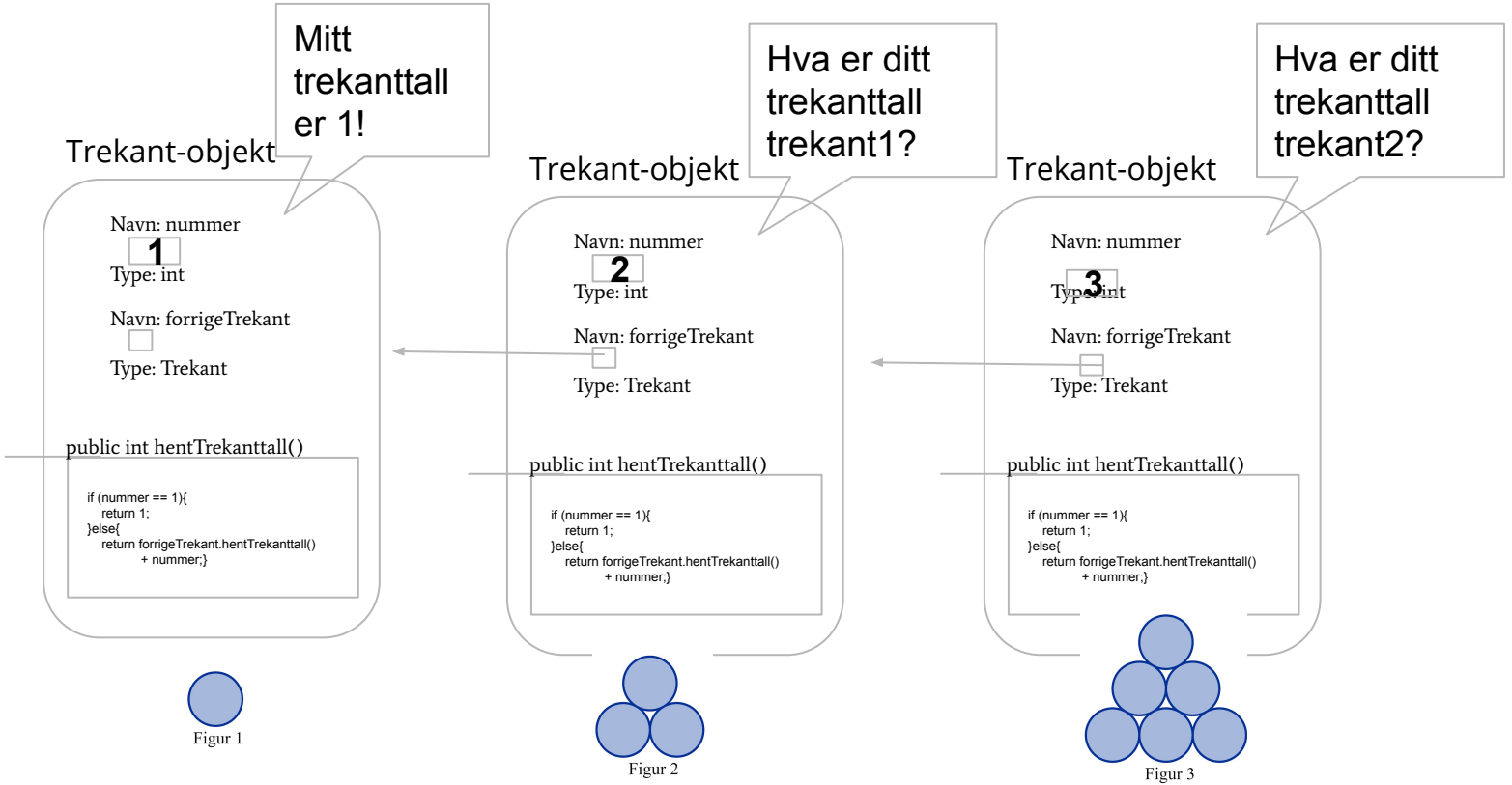
Rekursjon trekanttall



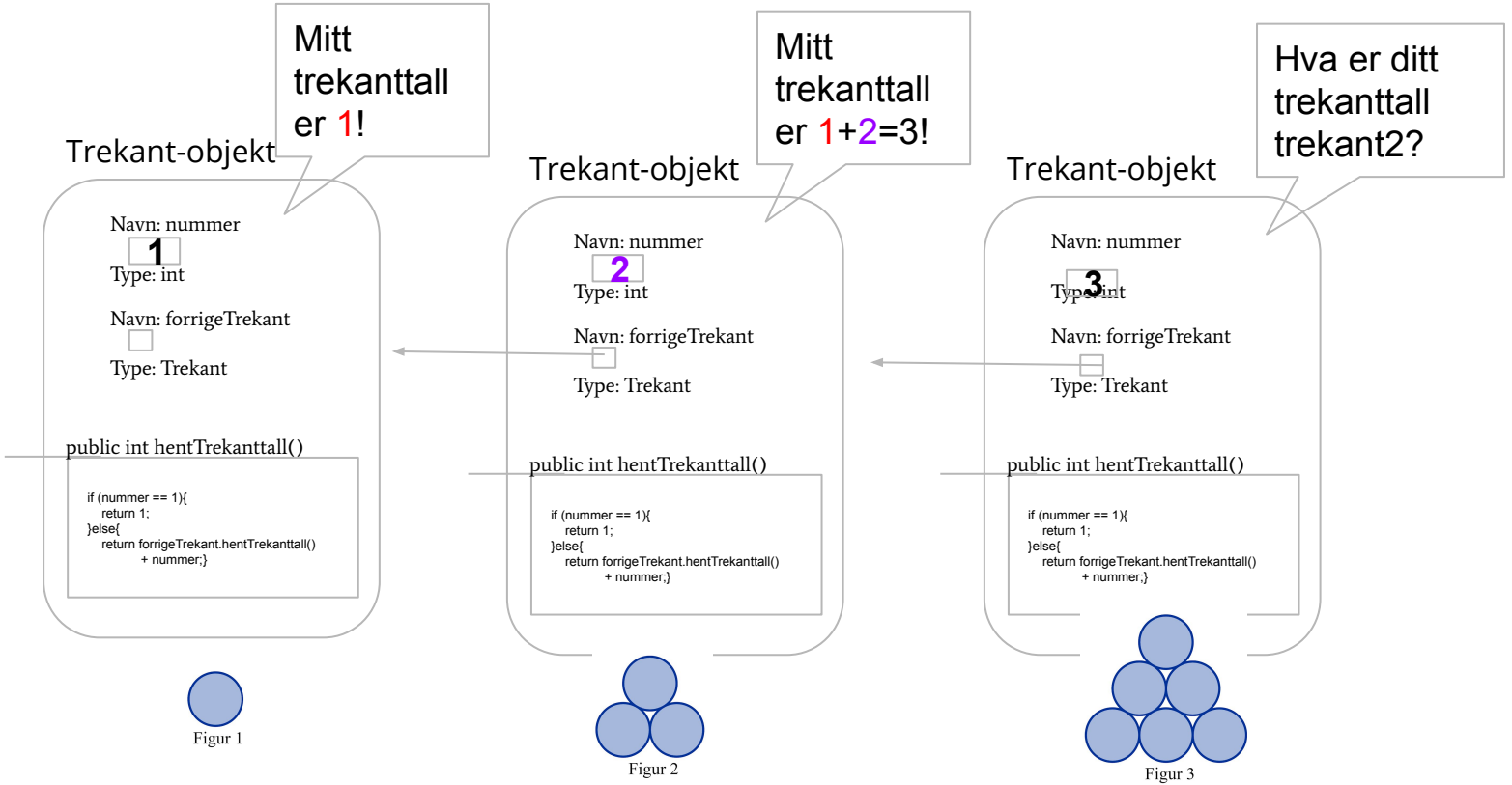
Rekursjon trekanttall



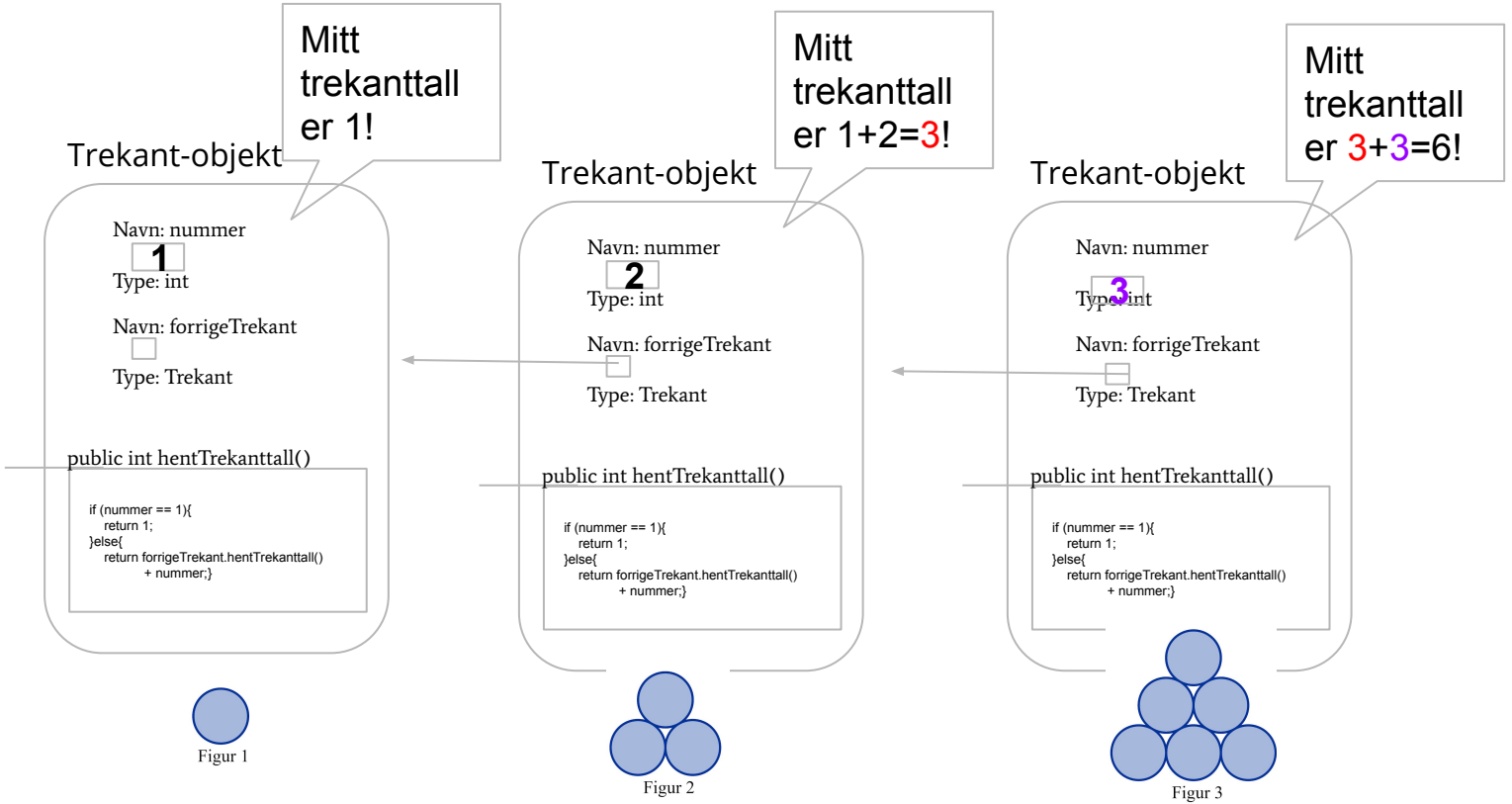
Rekursjon trekanttall



Rekursjon trekanttall



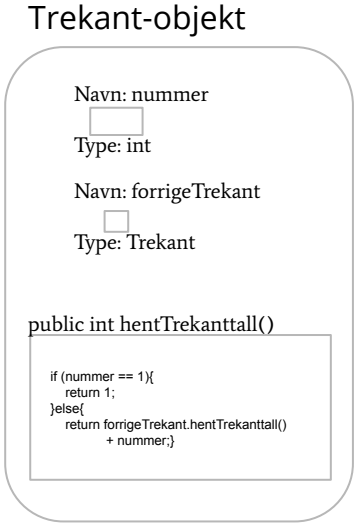
Rekursjon trekanttall



Rekursjon trekanttall

```java
class Trekant{
  private int nummer;
  private Trekant forrigeTrekant;

  public Trekant(int nummer, Trekant forrigeTrekant){
    this.nummer = nummer;
    this.forrigeTrekant = forrigeTrekant;
  }
  public int hentTrekanttall(){
    if (nummer == 1){
      return 1;
    }else{
      return forrigeTrekant.hentTrekanttall() + nummer;
    }
```

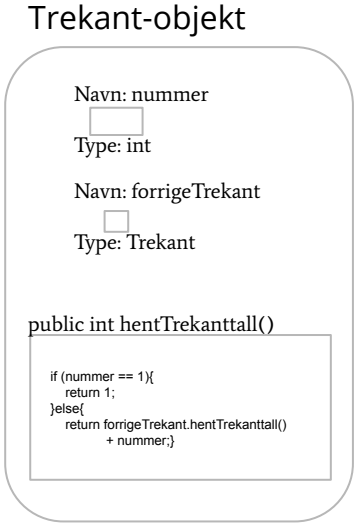


Rekursjon trekanttall

```java
class Trekant{
    private int nummer;
    private Trekant forrigeTrekant;

    public Trekant(int nummer, Trekant forrigeTrekant){
        this.nummer = nummer;
        this.forrigeTrekant = forrigeTrekant;
    }
    public int hentTrekanttall(){
        if (nummer == 1){
            return 1;
        }else{
            return forrigeTrekant.hentTrekanttall() + nummer;
        }
```

Basis case

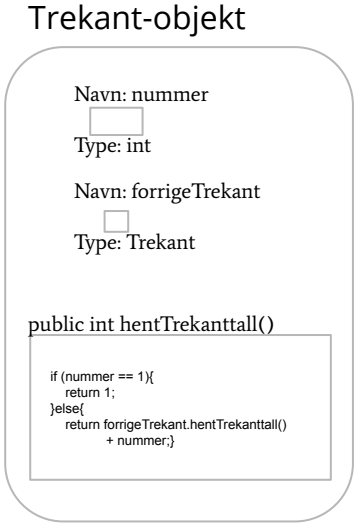


Rekursjon trekanttall

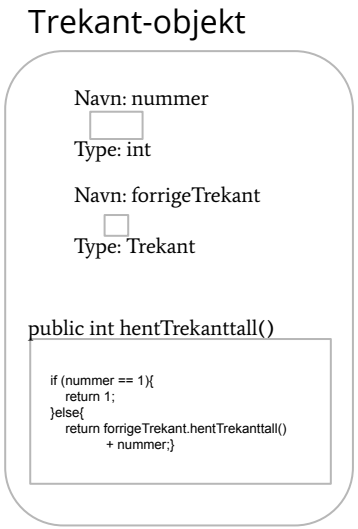
```java
class Trekant{
    private int nummer;
    private Trekant forrigeTrekant;

    public Trekant(int nummer, Trekant forrigeTrekant){
        this.nummer = nummer;
        this.forrigeTrekant = forrigeTrekant;
    }
    public int hentTrekanttall(){
        if (nummer == 1){
            return 1;
        }else{
            return forrigeTrekant.hentTrekanttall() + nummer;
        }
```

Enklere kall hver gang



Rekursjon trekanttall




```java
class Trekant{
  private int nummer;
  private Trekant forrigeTrekant;

  public Trekant(int nummer, Trekant forrigeTrekant){
    this.nummer = nummer;
    this.forrigeTrekant = forrigeTrekant;
  }
  public int hentTrekanttall(){
    if (nummer == 1){
      return 1;
    }else{
      return forrigeTrekant.hentTrekanttall() + nummer;
    }
```

```java
class TestTrekant{
  public static void main(String[] args) {
    Trekant trekant1 = new Trekant(1, null);
    Trekant trekant2 = new Trekant(2, trekant1);
    Trekant trekant3 = new Trekant(3, trekant2);
    Trekant trekant4 = new Trekant(4, trekant3);
    Trekant trekant5 = new Trekant(5, trekant4);
    Trekant trekant6 = new Trekant(6, trekant5);

    System.out.println(trekant6.hentTrekanttall());
  }
```

Kjøre koden

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

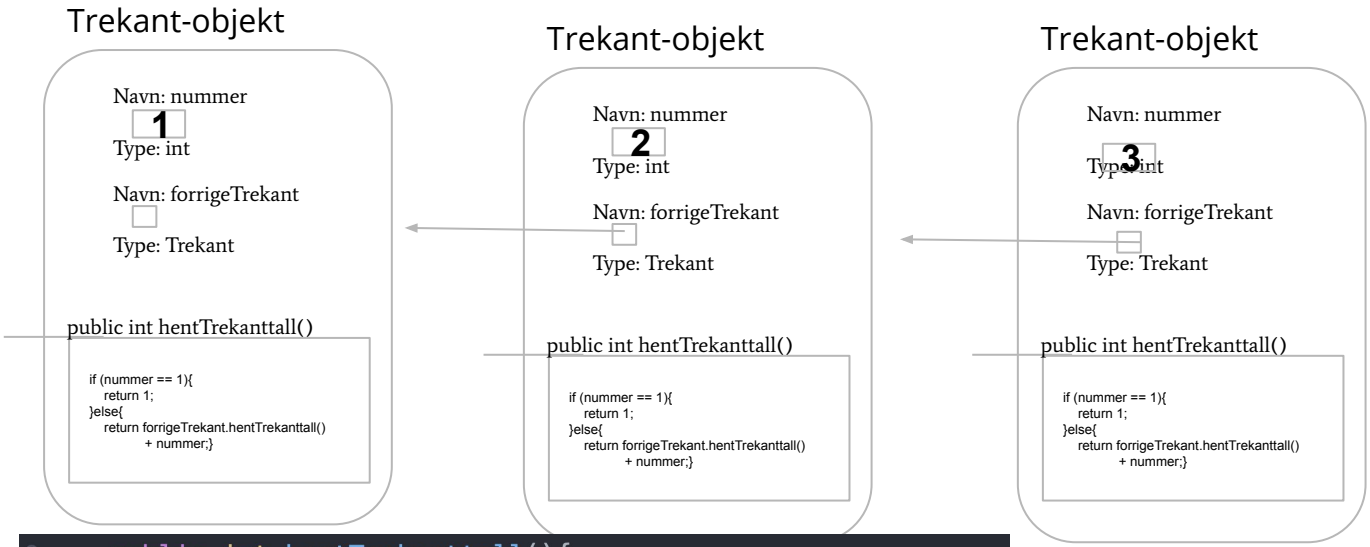
```java
class Trekant{
    private int nummer;
    private Trekant forrigeTrekant;

    public Trekant(int nummer, Trekant forrigeTrekant){
        this.nummer = nummer;
        this.forrigeTrekant = forrigeTrekant;
    }
    public int hentTrekanttall(){
        if (nummer == 1){
            return 1;
        }else{
            return forrigeTrekant.hentTrekanttall() + nummer;
        }
```

```java
class TestTrekant{
    public static void main(String[] args) {
        Trekant trekant1 = new Trekant(1, null);
        Trekant trekant2 = new Trekant(2, trekant1);
        Trekant trekant3 = new Trekant(3, trekant2);
        Trekant trekant4 = new Trekant(4, trekant3);
        Trekant trekant5 = new Trekant(5, trekant4);
        Trekant trekant6 = new Trekant(6, trekant5);

        System.out.println(trekant6.hentTrekanttall());
    }
```

Rekursjon trekanttall




```
9    public int hentTrekanttall(){
10       if (nummer == 1){
11           return 1;
12       }else{
13           return forrigeTrekant.hentTrekanttall() + nummer;
14       }
15   }
```

Oppmøteregistrering:

https://nettskjema.no/a/188717

Ris, ros, forslag ?

https://nettskjema.no/a/180345

Send meg en direktemelding 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

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 gratistjeneseter 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