Velkommen!

•••

Johanna
johannph på mattermost
johannph@uio.no på mail!

Kort: Praktisk informasjon

- 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!
- Skriv alt dere lurer på i chatten enten til everybody eller bare til meg 😊
  - Si i fra hvis dere faller av! Enten hvis noe er vanskelig, eller hvis dere glemte å følge med! Bare si i fra!
  - Vi har god tid, så det er masse tid til å gå gjennom ting flere ganger

Oblig 6: et lite tips!

Husk polimorfi!

Send meg en melding i chatten: hva er polimorfi?

Ingen bryr seg om du svarer feil, men gjør et forsøk på å svare.

Vi skal se litt på det sammen!

Polimorfi

```java
class IkkePolimorfi{
  public static void main(String[] args) {
    Frukt frukt1 = new Banan();
    Frukt frukt2 = new Eple();

    System.out.println(frukt1.hentFarge());
    System.out.println(frukt2.hentFarge());
  }
}

class Frukt{
  public String hentFarge(){
    if (this instanceof Banan ){
      return "Gul";
    } else if (this instanceof Eple){
      return "Rød";
    }
    return "Vet ikke"
  }

}
class Eple extends Frukt{
}
class Banan extends Frukt{
}
```

<- Ikke polimorfi

Polimorfi ->

Polimorfi er en bedre løsning

```
jonbon@jons-macbook-pro uke13 % java Polimorfi
Gul
Rød
jonbon@jons-macbook-pro uke13 % java IkkePolimorfi
Gul
Rød
```

```java
class Polimorfi{
  public static void main(String[] args) {
    Frukt frukt1 = new Banan();
    Frukt frukt2 = new Eple();

    System.out.println(frukt1.hentFarge());
    System.out.println(frukt2.hentFarge());
  }
}

class Frukt{
  public String hentFarge(){
    return "Vet ikke";
  }

}
class Eple extends Frukt{
  public string hentFarge(){
    return "Rød";
  }
}
class Banan extends Frukt{
  public string hentFarge(){
    return "Gul"
}
```

Juks – kopiering av kode

Hovedpoenget med obligene er at dere skal øve. Man kan ikke bli utvikler uten å progge.

Er man stuck burde man finne ut nøyaktig hva som er vanskelig og så googel det/tenke på det/spørre noen om akkurat det. I jobb er det ingen som sitter med en fasit dere kan titte litt på. En av de _viktigste_ skillsene for å bli god utvikler er å kunne google.

Ser dere på koden til andre som har løst obligen mister dere masse læringspotensial.

Kopiering av kode er juks. Og det går bare utover den som kopierer, ikke den som deler kode.

Repetisjon forrige uke

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

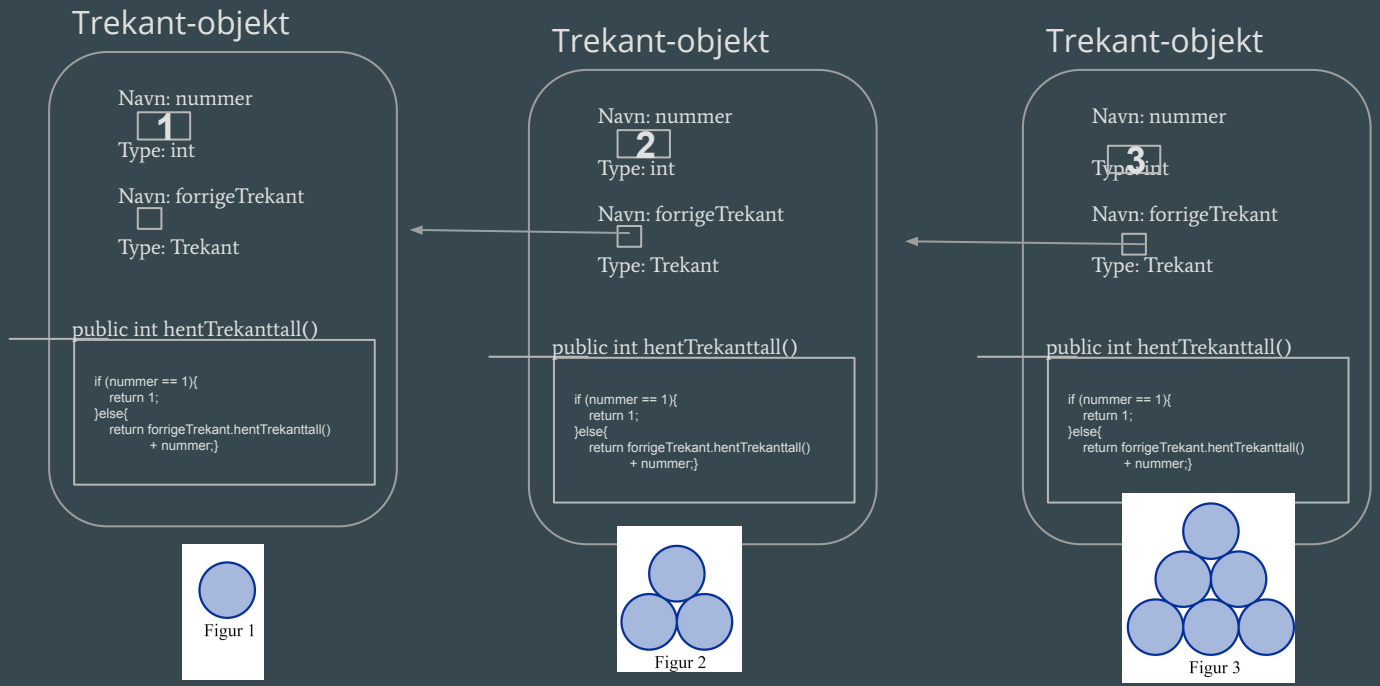
```java
class EnkelIterasjon{
  public static void main(String[] args) {
    System.out.println("Iterasjon:");
    skrivTallIterasjon(5);
  }
  public static void skrivTallIterasjon(int n){
    for(int i = n; i >= 0; i--){
      System.out.println(i);
    }
  }
```

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

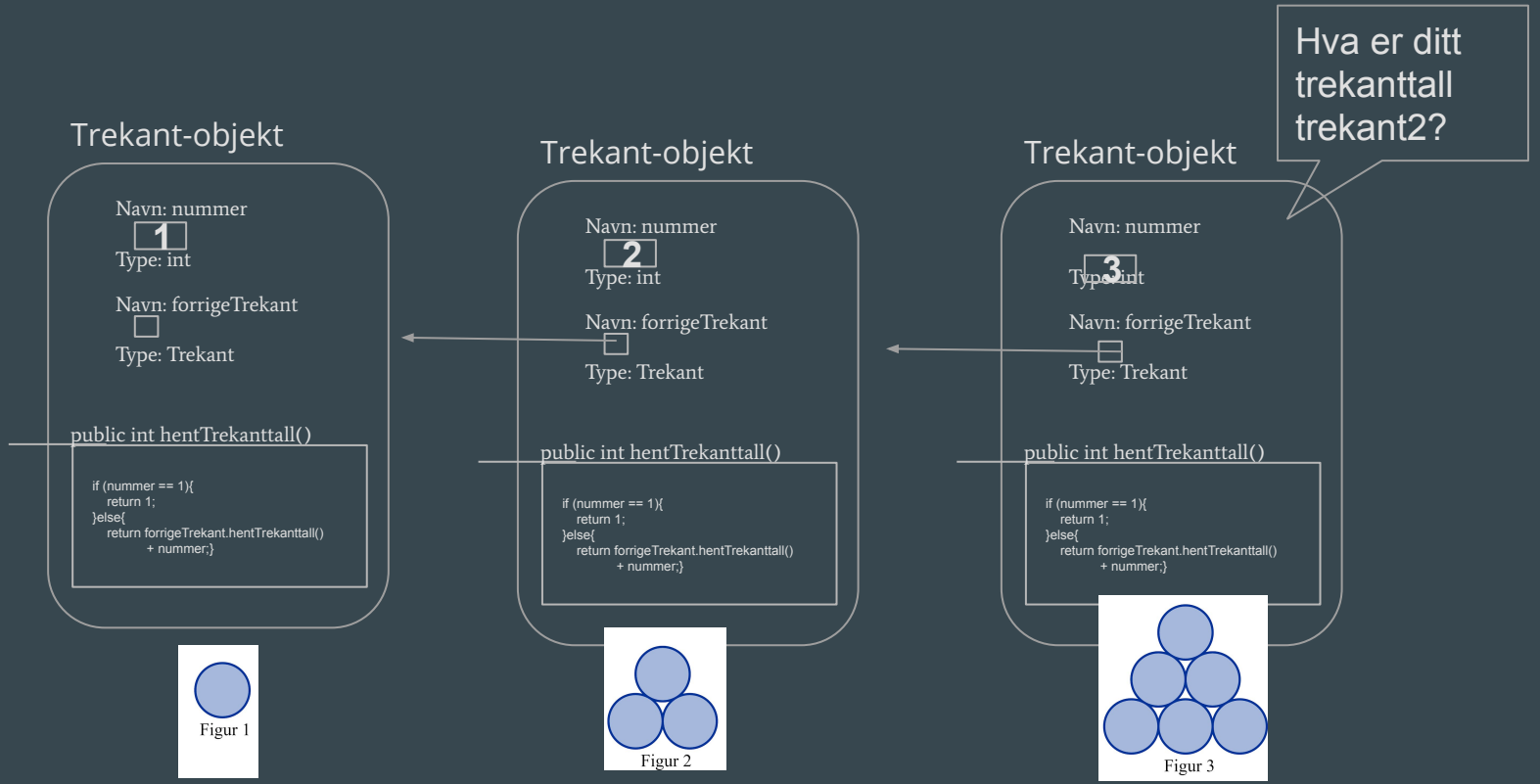
```java
class EnkelRekursjon {
  public static void main(String[] args) {
    System.out.println("Rekursjon:");
    skrivTallRekursjon(5);
  }
  public static void skrivTallRekursjon(int n){
    if (n < 0) {
      return;
    }
    System.out.println(n);
    skrivTallRekursjon(n-1);
```

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

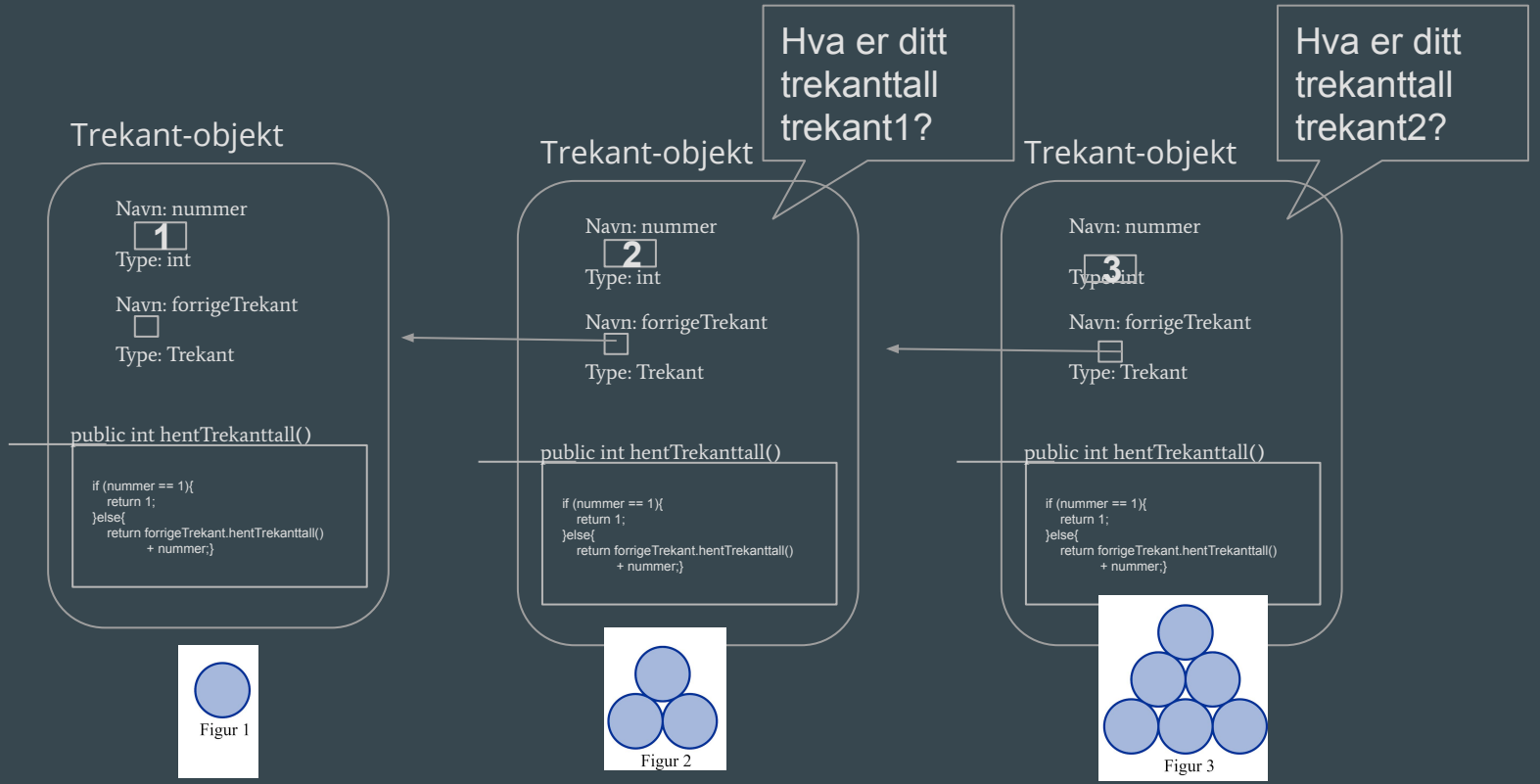
Rekursjon trekanttall



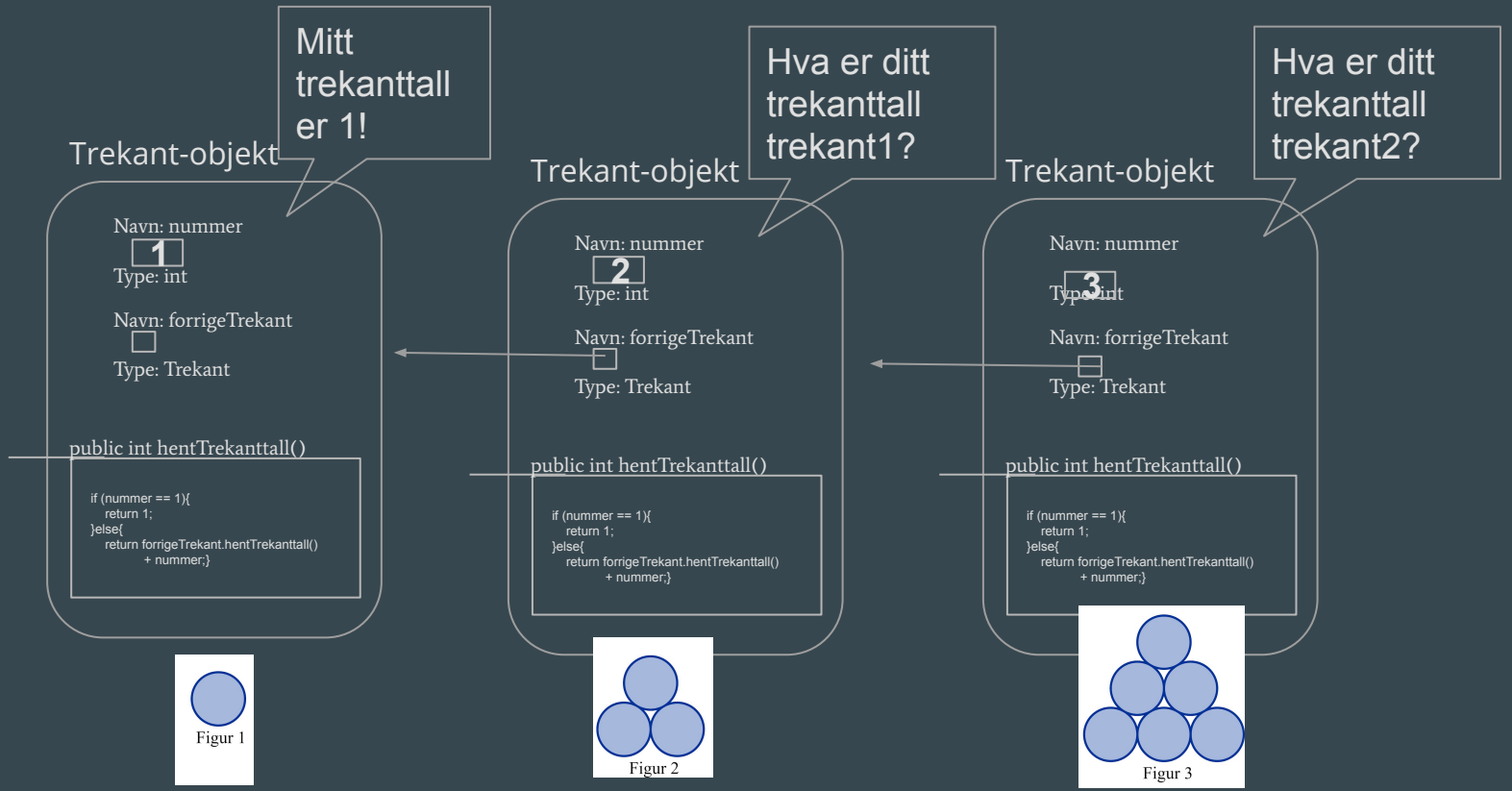
Rekursjon trekanttall



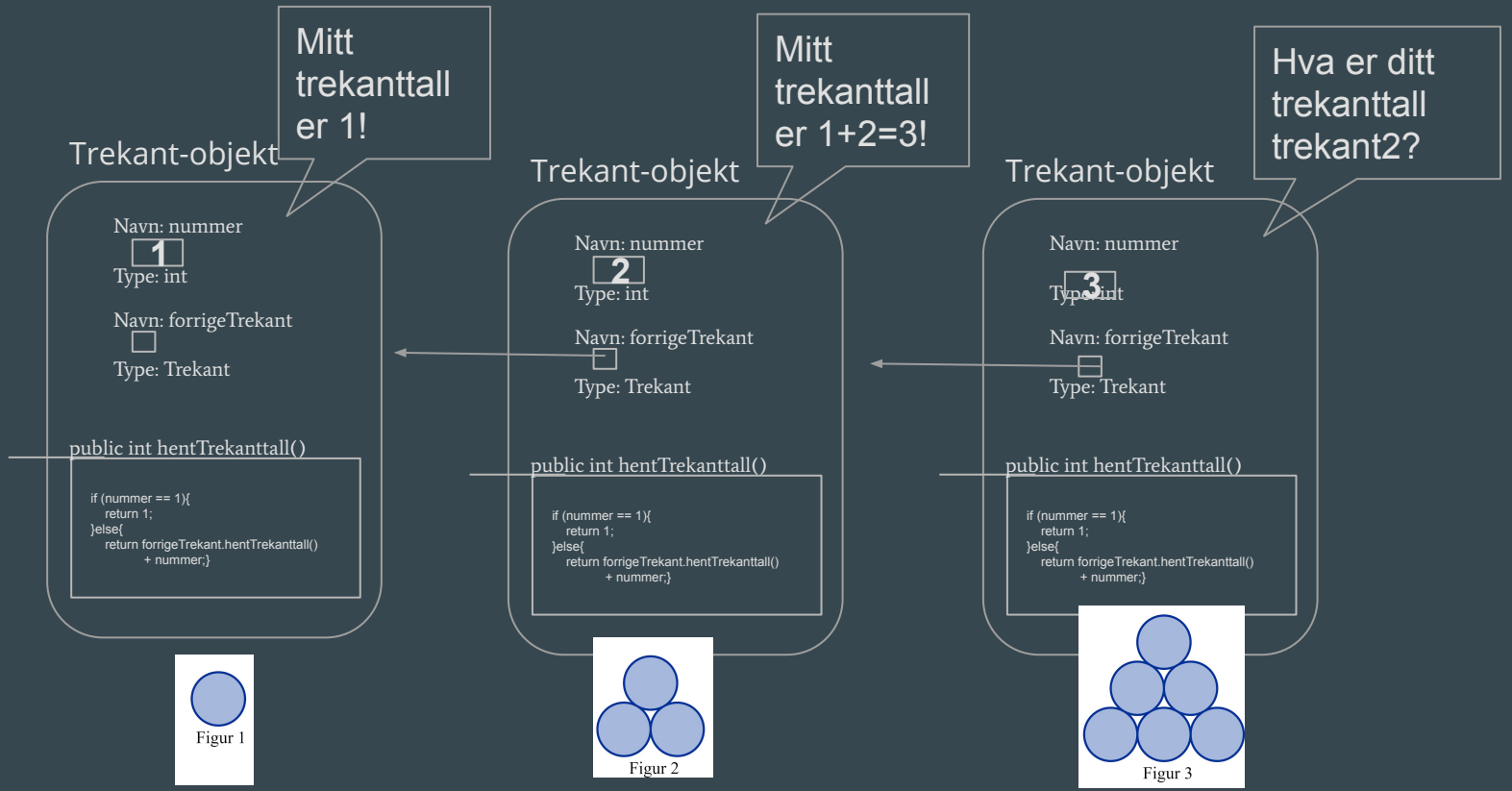
Rekursjon trekanttall



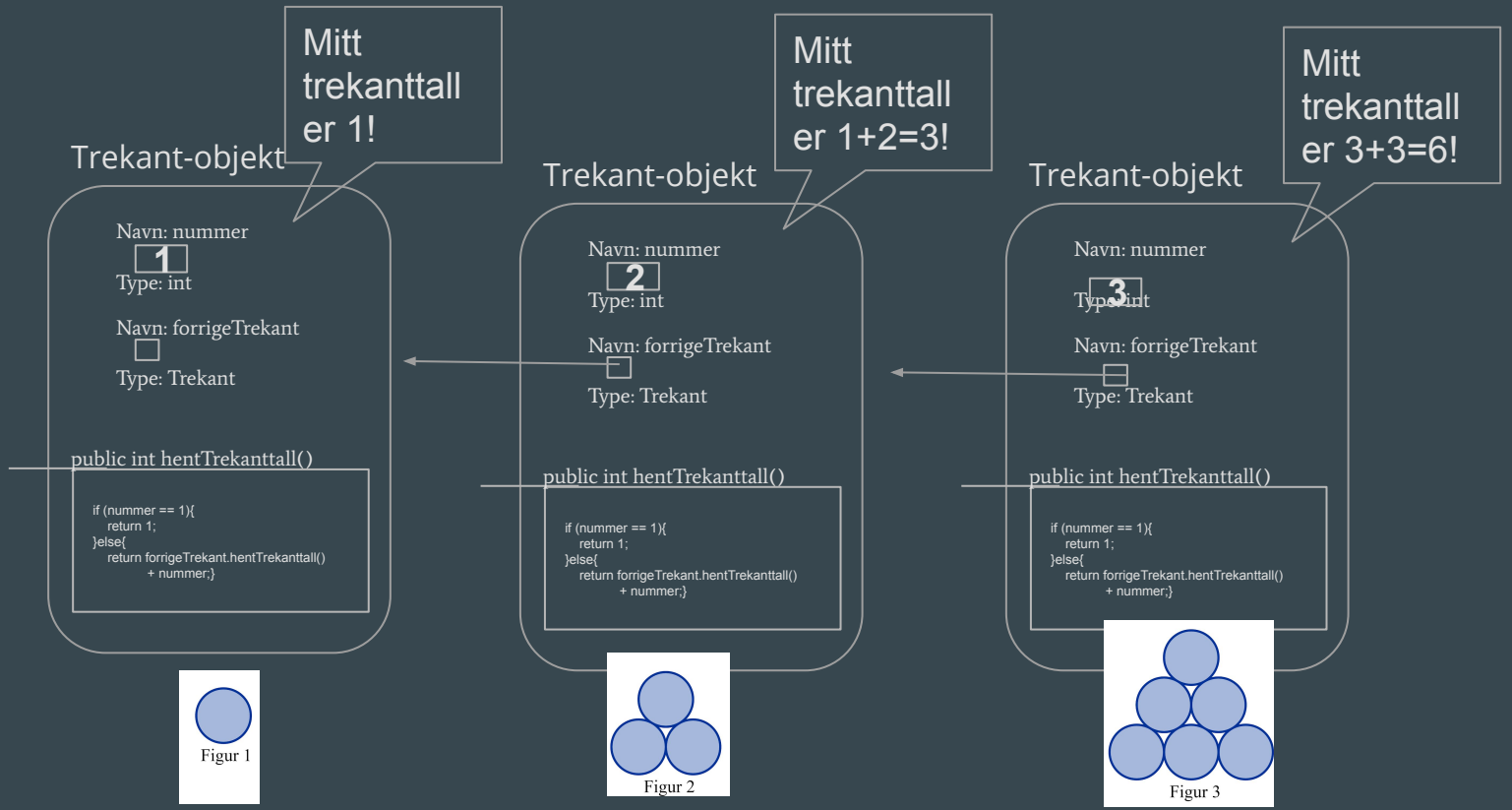
Rekursjon trekanttall



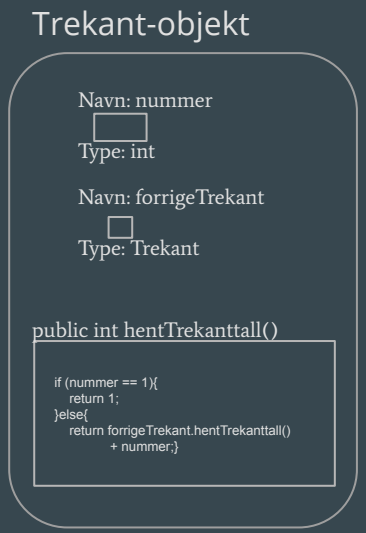
Rekursjon trekanttall



Rekursjon trekanttall



Rekursjon trekanttall

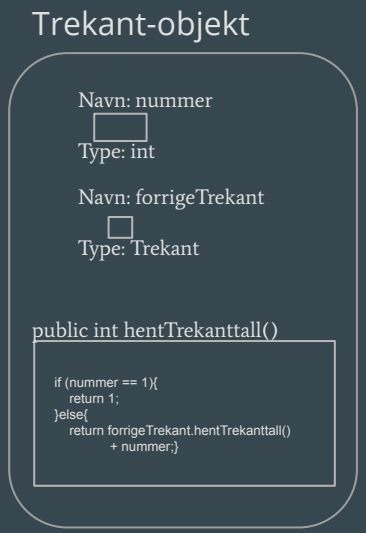


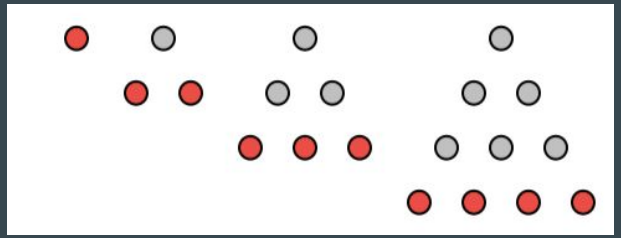

```java
class Trekant{
    private int nummer;
    private Trekant forrigeTrekant;

    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

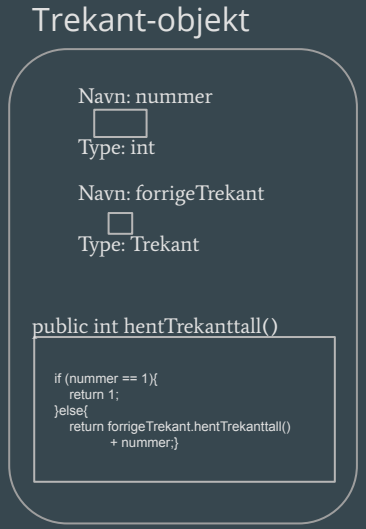


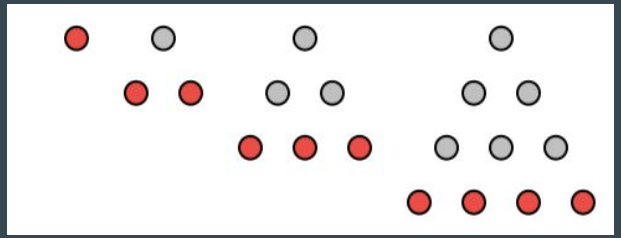


Rekursjon trekanttall

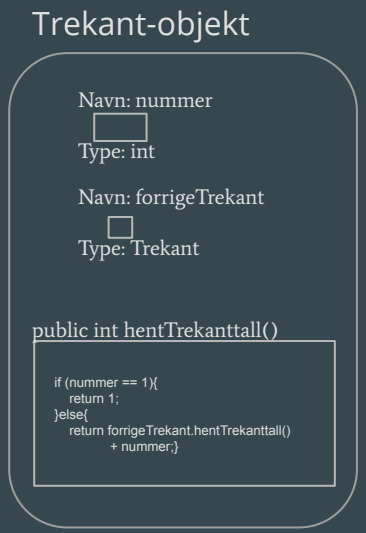
```java
class Trekant{
    private int nummer;
    private Trekant forrigeTrekant;

    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());
    }
```

Før og etter rekursivt kall

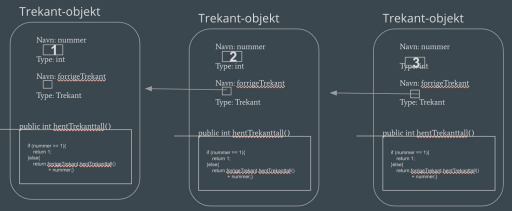
Viktig å henge med på hva som skjer FØR vs. hva som skjer ETTER et rekursivt kall!

Vi skal legge til noen print-lines i koden og printe før og etter de rekursive kallene!

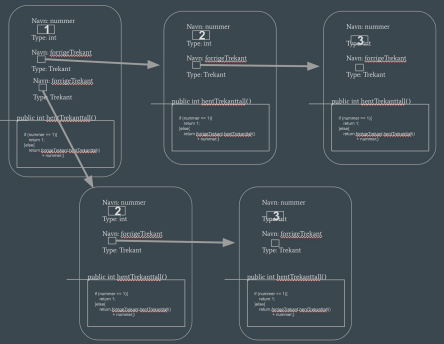
```
jonbon@jons-macbook-pro uke12 % java TestTrekant
Henter trekanttallet til trekant 6:
---------------------------------
Trekant 6 sier: Hva er ditt trekanttall, trekant 5?
Trekant 5 sier: Hva er ditt trekanttall, trekant 4?
Trekant 4 sier: Hva er ditt trekanttall, trekant 3?
Trekant 3 sier: Hva er ditt trekanttall, trekant 2?
Trekant 2 sier: Hva er ditt trekanttall, trekant 1?
Trekant 1 sier: Mitt trekanttall er 1!
Trekant 2 sier: Da er mitt trekanttall: 1+2=3!
Trekant 3 sier: Da er mitt trekanttall: 3+3=6!
Trekant 4 sier: Da er mitt trekanttall: 6+4=10!
Trekant 5 sier: Da er mitt trekanttall: 10+5=15!
Trekant 6 sier: Da er mitt trekanttall: 15+6=21!
---------------------------------
Trekanttallet til trekant 6 er: 21
```

Rekursjon kan grene utover

Vi så på dette eksempelet der strukturen ligner veldig på en lenkeliste og det rekursive kallet går i én retning:



I obligen skal dere implementere et rekursivt kall i noe som egentlig er en graf eller et tre, og det rekursive kallet går i flere retninger, det kan være flere rekursive kall i samme metode:



Repetisjon denne uka

S
O
L
I
D

- **Single responsibility principle**
  En klasse bør bare ha ett ansvarsområde, og dette bør være veldefinert
- **Open/closed principle**
  Komponenter bør være åpne for utvidelser, men lukket for modifikasjoner. Eksempel: klasser kontrollerer hva subklasser kan endre (private, protected, final, etc).
- **Liskov substitution principle**
  Objekter i et program bør kunne byttes ut med instanser av subtyper uten at dette endrer programmets korrekthet.
- **Interface segregation principle**
  Mange små grensesnitt er bedre enn store, monolittiske "super-grensesnitt"
- **Dependency inversion principle**
  Komponenter bør avhenge av abstraksjoner, og ikke konkrete implementasjoner

Enhetstesting

Teste komponenter av koden hver for seg, f.eks. en metode eller en klasse.

Etter enhetstesting kan vi sette komponentene sammen og teste hvordan det fungerer sammen (integrasjontest) og etterhvert hele systemet (systemtesting). Men i in1010 lærer vi bare litt enhetstesting.

Poenget er å finne feil så tidlig som mulig. Dere har sikkert selv erfart at det er vanskeligere å finne feil dess mer kode det er.

Eksempel enhetstesting

Jobbe selv

Jobb med hva dere vil og rekk opp hånda hvis dere trenger hjelp med noe/har spørsmål så møtes vi i breakoutroom! 😊