



IN1010 - Seminar 4

- Arv og Polymorfi
- 

Praktisk

Oblig 2 er nå lagt ut, frist 15. februar kl 23.59

Husk å sjekke emnesiden regelmessig

Meld deg opp på lab hvis du trenger hjelp med obligen :D



Oppmøteregistrering:
<https://nettskjema.no/a/184340>



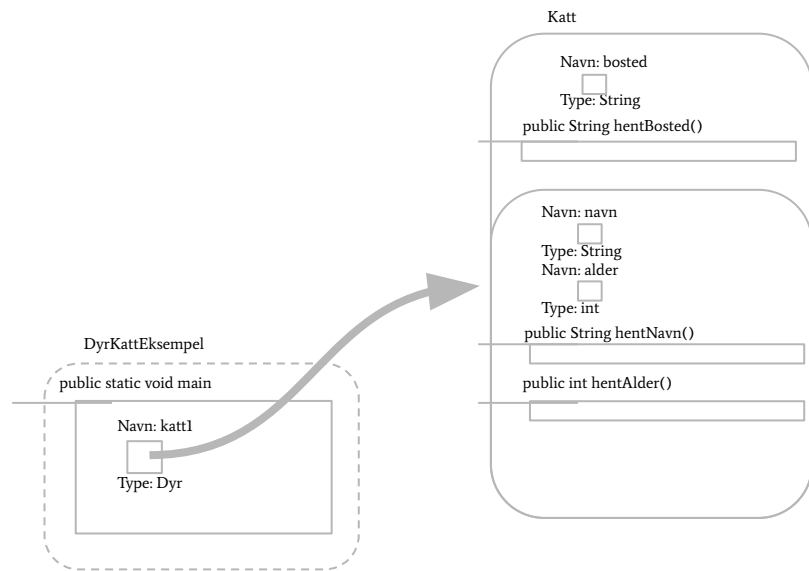
Casting eksempel

```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted){  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted(){  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder){  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn(){  
        return navn;  
    }  
  
    public int hentAlder(){  
        return alder;  
    }  
}
```

```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted){  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted(){  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder){  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn(){  
        return navn;  
    }  
  
    public int hentAlder(){  
        return alder;  
    }  
}
```



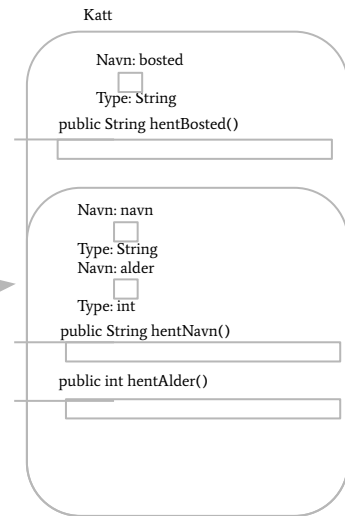
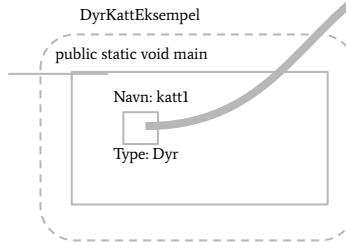
```
class DyrKattEksempel {  
    public static void main(String[] args) {  
        Dyr katt1 = new Katt("Pus", 1, "Oslo");  
        System.out.println(katt1.hentNavn());  
        System.out.println(katt1.hentAlder());  
        System.out.println(katt1.hentBosted());  
    }  
}
```

```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted){  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted(){  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder){  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn(){  
        return navn;  
    }  
  
    public int hentAlder(){  
        return alder;  
    }  
}
```

Siden variabelen er av type dyr må vi ta på oss dyre brillene

```
class DyrKattEksempel {  
    public static void main(String[] args) {  
        Dyr katt1 = new Katt("Pus", 1, "Oslo");  
        System.out.println(katt1.hentNavn());  
        System.out.println(katt1.hentAlder());  
        System.out.println(katt1.hentBosted());  
    }  
}
```

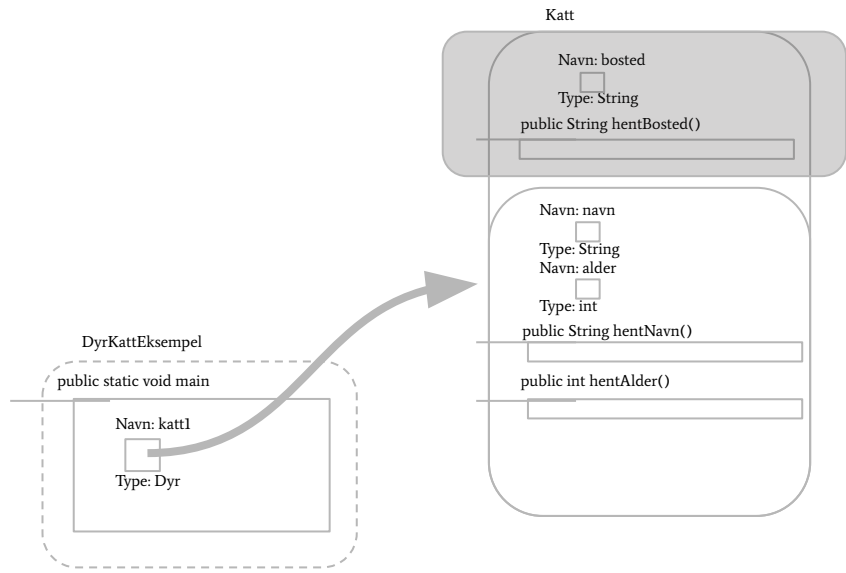


```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted){  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted(){  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder){  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn(){  
        return navn;  
    }  
  
    public int hentAlder(){  
        return alder;  
    }  
}
```

Katt1 ser nå kun
egenskapen til
klassen Dyr

```
class DyrKattEksempel {  
    public static void main(String[] args) {  
        Dyr katt1 = new Katt("Pus", 1, "Oslo");  
        System.out.println(katt1.hentNavn());  
        System.out.println(katt1.hentAlder());  
        System.out.println(katt1.hentBosted());  
    }  
}
```

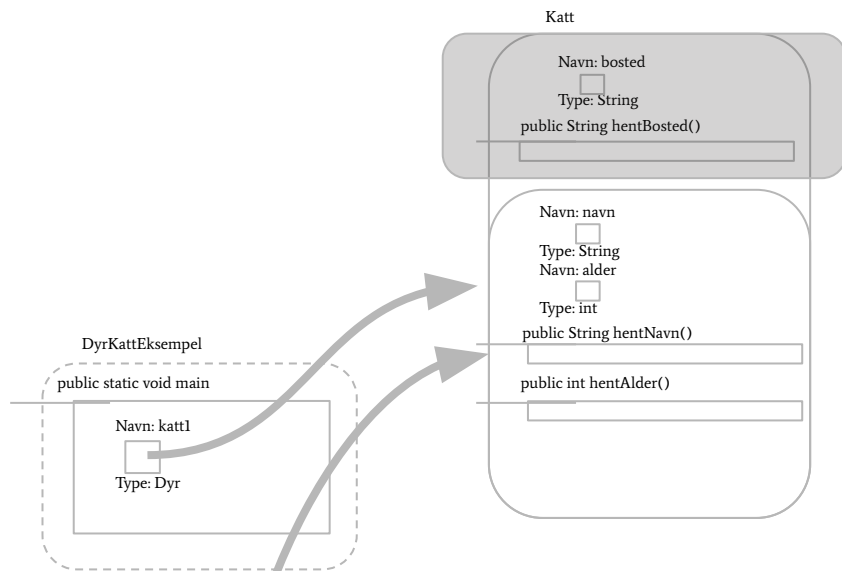


```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted){  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted(){  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder){  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn(){  
        return navn;  
    }  
  
    public int hentAlder(){  
        return alder;  
    }  
}
```

Output: Pus

```
class DyrKattEksempel {  
    public static void main(String[] args) {  
        Dyr katt1 = new Katt("Pus", 1, "Oslo");  
        System.out.println(katt1.hentNavn());  
        System.out.println(katt1.hentAlder());  
        System.out.println(katt1.hentBosted());  
    }  
}
```

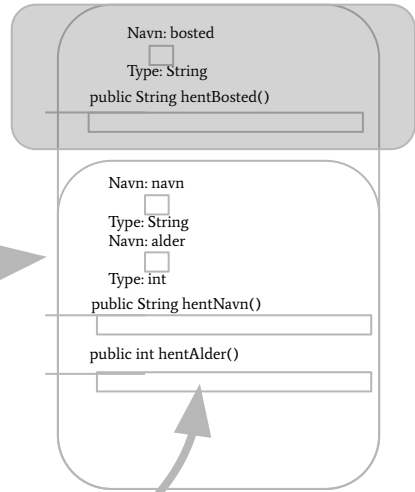
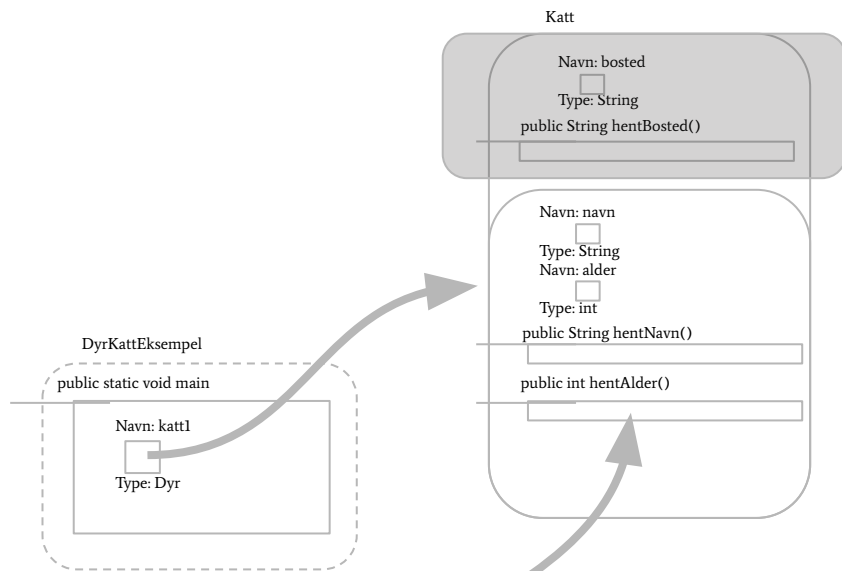



```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted){  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted(){  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder){  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn(){  
        return navn;  
    }  
  
    public int hentAlder(){  
        return alder;  
    }  
}
```

Output: 1

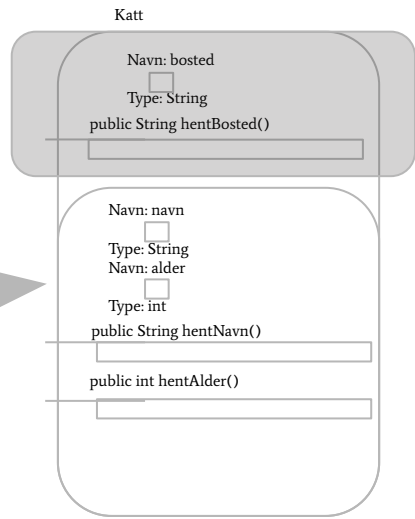
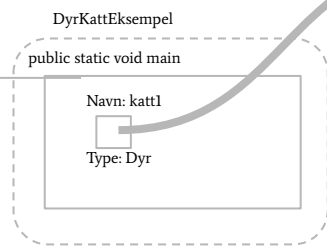
```
class DyrKattEksempel {  
    public static void main(String[] args) {  
        Dyr katt1 = new Katt("Pus", 1, "Oslo");  
  
        System.out.println(katt1.hentNavn());  
        System.out.println(katt1.hentAlder());  
        System.out.println(katt1.hentBosted());  
    }  
}
```



```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted) {  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted() {  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder) {  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn() {  
        return navn;  
    }  
  
    public int hentAlder() {  
        return alder;  
    }  
}
```

Feilmelding. Siden katt1 ikke ser egenskapene til Katt objektet og dermed heller ikke metoden hentBosted. Derfor må vi caste!



```
class DyrKattEksempel {  
    public static void main(String[] args) {  
        Dyr katt1 = new Katt("Pus", 1, "Oslo");  
  
        System.out.println(katt1.hentNavn())  
        System.out.println(katt1.hentAlder())  
        System.out.println(katt1.hentBosted())  
    }  
}
```

?? X

```
class Katt extends Dyr {
    private String bosted;

    public Katt(String navn, int alder, String bosted) {
        super(navn, alder);
        this.bosted = bosted;
    }

    public String hentBosted() {
        return bosted;
    }
}
```

```
abstract class Dyr {
    protected String navn;
    protected int alder;

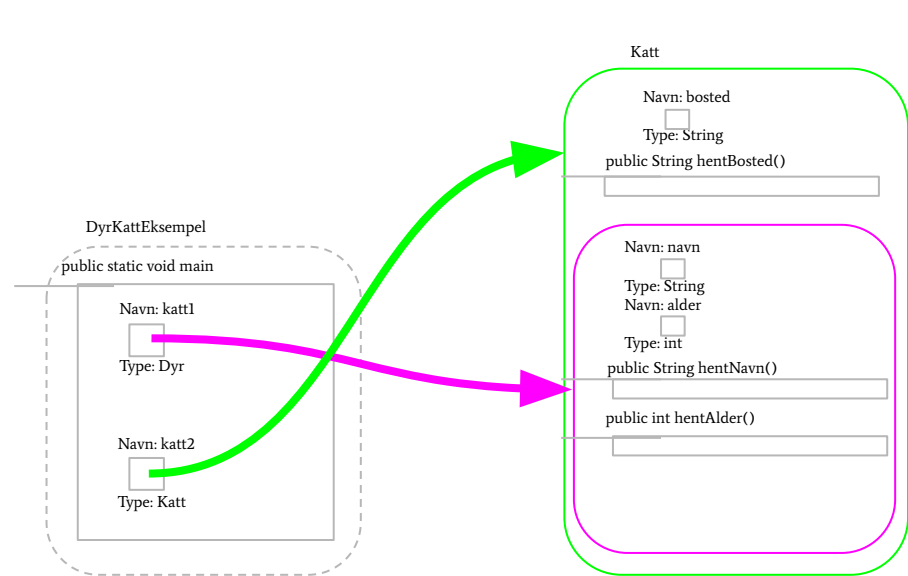
    public Dyr(String navn, int alder) {
        this.navn = navn;
        this.alder = alder;
    }

    public String hentNavn() {
        return navn;
    }

    public int hentAlder() {
        return alder;
    }
}
```

```
class DyrKattEksempel {
    public static void main(String[] args) {
        Dyr katt1 = new Katt("Pus", 1, "Oslo");
        System.out.println(katt1.hentNavn());
        System.out.println(katt1.hentAlder());
        //System.out.println(katt1.hentBosted());

        Katt katt2 = (Katt) katt1;
        System.out.println(katt2.hentBosted());
        System.out.println(katt2.hentNavn());
    }
}
```



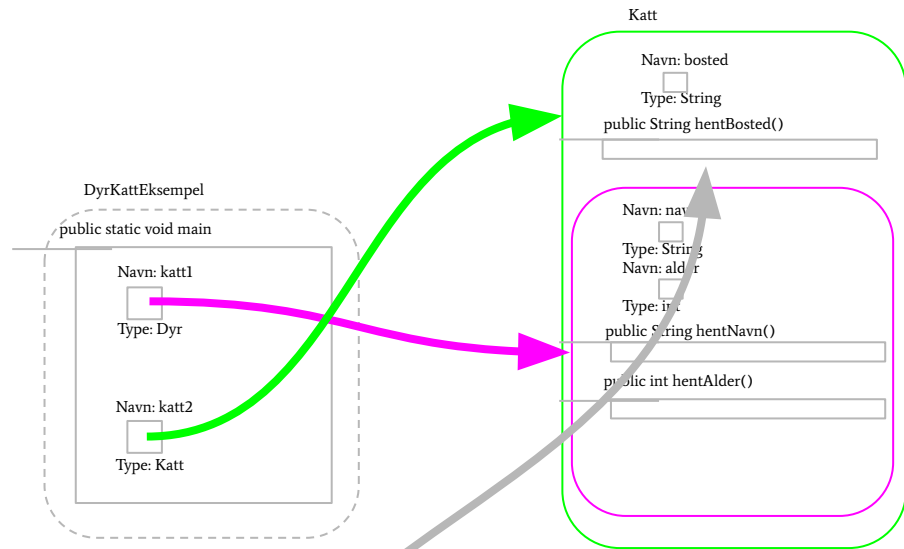
Når vi nå ser på katte objektet men katt2 sine briller får vi også alle egenskapene til en Katt. (Det som er markert i grønt)

PS: katt1 kan fortsatt bare se dyreegenskapene (Det som er markert i rosa)

```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted){  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted(){  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder){  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn(){  
        return navn;  
    }  
  
    public int hentAlder(){  
        return alder;  
    }  
}
```

```
class DyrKattEksempel {  
    public static void main(String[] args) {  
        Dyr katt1 = new Katt("Pus", 1, "Oslo");  
        System.out.println(katt1.hentNavn());  
        System.out.println(katt1.hentAlder());  
        //System.out.println(katt1.hentBosted());  
  
        Katt katt2 = (Katt) katt1;  
        System.out.println(katt2.hentBosted());  
        System.out.println(katt2.hentNavn());  
    }  
}
```

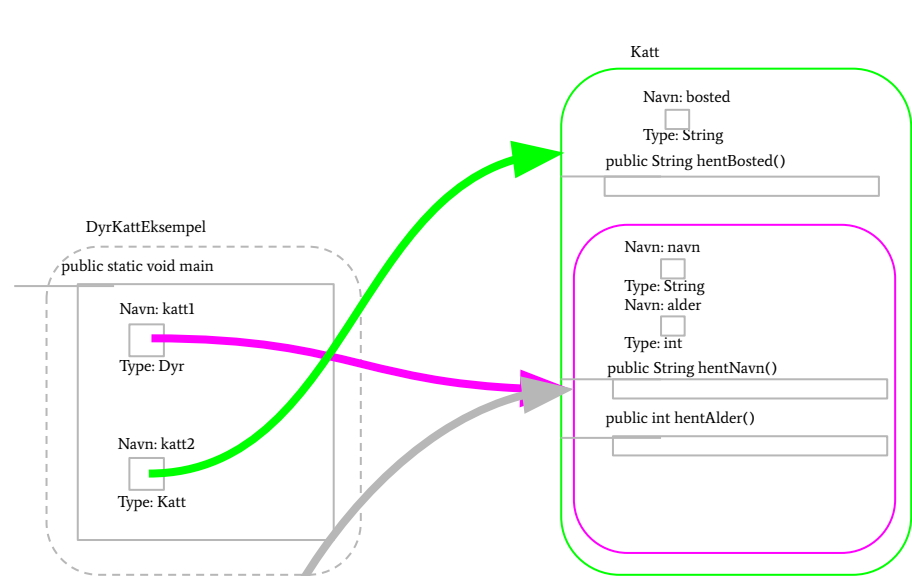


Output: Oslo

```
class Katt extends Dyr {  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted) {  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted() {  
        return bosted;  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder) {  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn() {  
        return navn;  
    }  
  
    public int hentAlder() {  
        return alder;  
    }  
}
```

```
class DyrKattEksempel {  
    public static void main(String[] args) {  
        Dyr katt1 = new Katt("Pus", 1, "Oslo");  
        System.out.println(katt1.hentNavn());  
        System.out.println(katt1.hentAlder());  
        //System.out.println(katt1.hentBosted());  
  
        Katt katt2 = (Katt) katt1;  
        System.out.println(katt2.hentBosted());  
        System.out.println(katt2.hentNavn());  
    }  
}
```



Output: Pus

Repetisjon denne uken

super() konstruktør

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bosted){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12 }
```

super() konstruktør

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bosted){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12 }
```


super() konstruktør

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bosted){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12 }
```

```
1 class TestArv{
2     public static void main(String[] args) {
3         // Lov!
4         Dyr katt1 = new Katt("Bjarne", 15, "Lillehammerveien 5");
5         katt1.skrivUtInfo();
6     }
7 }
```

Samme klasser som forrige slide, lager et hovedprogram for å kjøre koden.

super() konstruktør

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bosted){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12 }
```

```
1 class TestArv{
2     public static void main(String[] args) {
3         // Lov!
4         Dyr katt1 = new Katt("Bjarne", 15, "Lillehammerveien 5");
5         katt1.skrivUtInfo();
6     }
7 }
```

Kaller på katt.skrivUtInfo().

Da er det metoden i Dyr som kjører, siden Katt ikke har noen skrivUtInfo() metode.

super() konstruktør

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bosted){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12 }
```

```
1 class TestArv{
2     public static void main(String[] args) {
3         // Lov!
4         Dyr katt1 = new Katt("Bjarne", 15, "Lillehammerveien 5");
5         katt1.skrivUtInfo();
6     }
7 }
```

Kaller på katt.skrivUtInfo().

Da er det metoden i Dyr som kjører, siden Katt ikke har noen skrivUtInfo() metode.

```
jonbon@jons-macbook-pro uke4 % java TestArv
Navn: Bjarne
Alder: 15
```

.super

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bosted){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12 }
```

```
1 class TestArv{
2     public static void main(String[] args) {
3         // Lov!
4         Dyr katt1 = new Katt("Bjarne", 15, "Lillehammerveien 5");
5         katt1.skrivUtInfo();
6     }
7 }
```

Bostedsadresse blir ikke skrevet ut

```
jonbon@jons-macbook-pro uke4 % java TestArv
Navn: Bjarne
Alder: 15
```

.super

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bostedsadresse){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12    public void skrivUtInfo(){
13        super.skrivUtInfo();
14        System.out.println(bostedsadresse);
15    }
16 }
```

```
1 class TestArv{
2     public static void main(String[] args) {
3         // Lov!
4         Dyr katt1 = new Katt("Bjarne", 15, "Lillehammerveien 5");
5         katt1.skrivUtInfo();
6     }
7 }
```

Legger til metoden skrivUtInfo() i Katt.

Men kaller også på super.skrivUtInfo() slik at navn og alder også blir skrevet ut.

```
jonbon@jons-macbook-pro uke4 % java TestArv
Navn: Bjarne
Alder: 15
Lillehammerveien 5
```

Polimorfy

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bostedsadresse){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12    public void skrivUtInfo(){
13        super.skrivUtInfo();
14        System.out.println(bostedsadresse);
15    }
16 }
```

```
1 class TestArv{
2     public static void main(String[] args) {
3         // Lov!
4         Dyr katt1 = new Katt("Bjarne", 15, "Lillehammerveien 5");
5         katt1.skrivUtInfo();
6     }
7 }
```

Det at det nå finnes en metode skrivUtInfo() i Dyr og en metode skrivUtInfo() i Katt heter polymorfi!

At variabelen katt1 har typer Dyr() er irrelevant, det er objektet som bestemmer hvilken metode som kjører

Polimorfy og override

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bostedsadresse){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12    @Override
13    public void skrivUtInfo(){
14        super.skrivUtInfo();
15        System.out.println(bostedsadresse);
16    }
17 }
```

```
1 class TestArv{
2     public static void main(String[] args) {
3         // Lov!
4         Dyr katt1 = new Katt("Bjarne", 15, "Lillehammerveien 5");
5         katt1.skrivUtInfo();
6     }
7 }
```

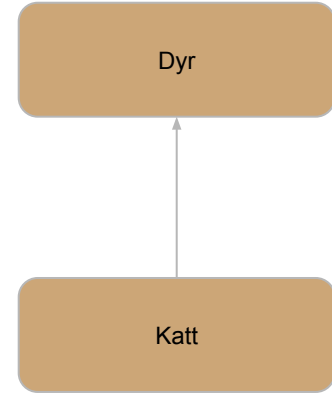
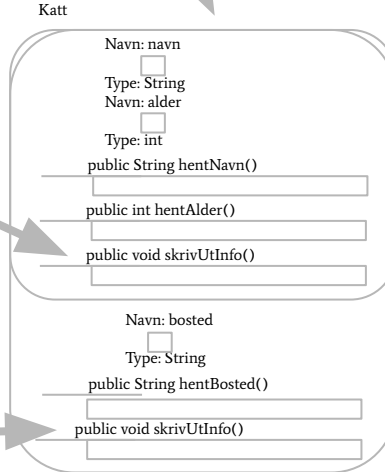
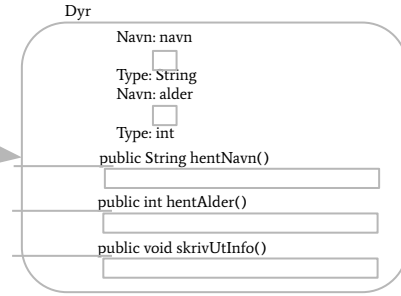
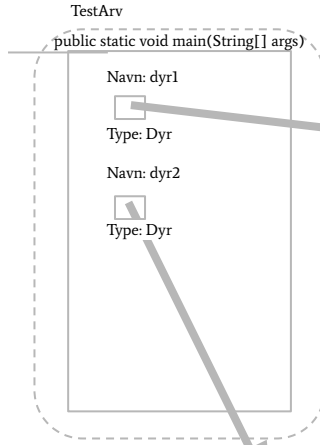
Men vi burde bruke @Override for å være sikker på at programmet vårt skjønner at skrivUtInfo i Katt skal overskrive skrivUtInfo() i Dyr.

Da vil kompilatoren sjekke at vi ikke har skrevet feil, f.eks. SkivutInfo() og skrivUtInfo() (skrivefeil) eller void skrivUtInfo() og String skrivUtInfo() (ulike returtyper)

Polymorfi

```
1 class Dyr{
2     protected int alder;
3     protected String navn;
4
5     public Dyr(String navn, int alder){
6         this.navn = navn;
7         this.alder = alder;
8     }
9     public int hentAlder(){
10        return alder;
11    }
12    public String hentNavn(){
13        return navn;
14    }
15    public void skrivUtInfo(){
16        System.out.println("Navn: " + navn);
17        System.out.println("Alder: " + alder);
18    }
19 }
```

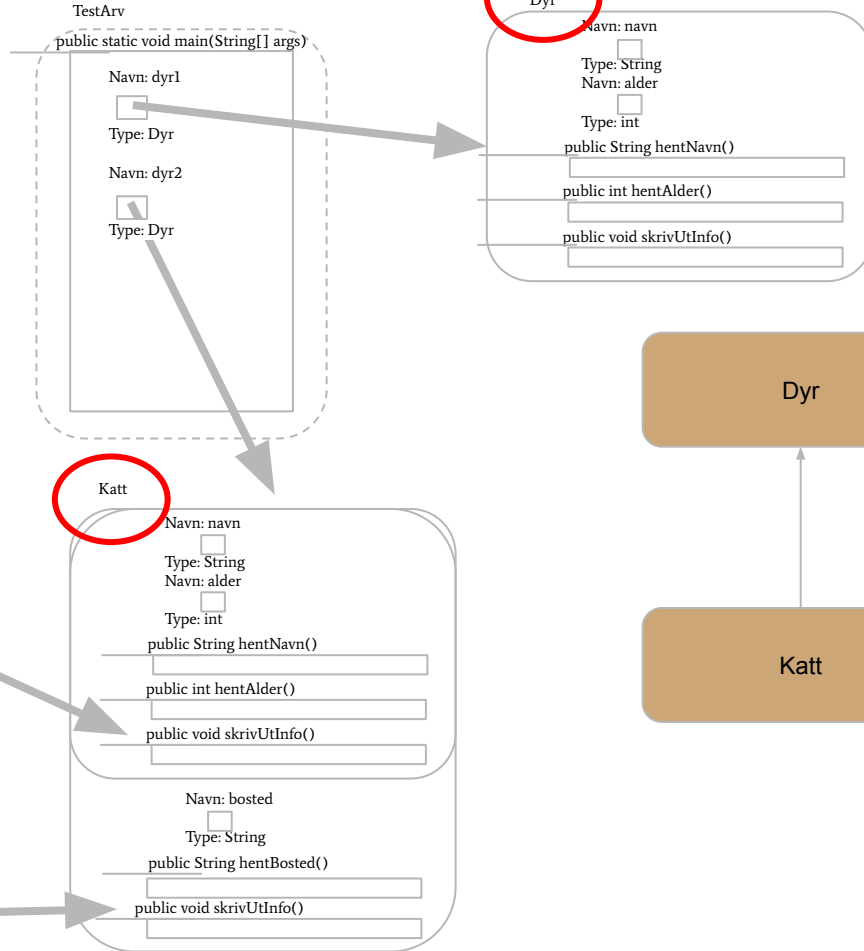
```
1 class Katt extends Dyr{
2
3     private String bostedsadresse;
4
5     public Katt(String navn, int alder, String bostedsadresse){
6         super(navn, alder);
7         this.bostedsadresse = bostedsadresse;
8     }
9     public String hentBosted(){
10        return bostedsadresse;
11    }
12    @Override
13    public void skrivUtInfo(){
14        super.skrivUtInfo();
15        System.out.println(bostedsadresse);
16    }
17 }
```



Polymorfi

```
1 class Dyr{
2   protected int alder;
3   protected String navn;
4
5   public Dyr(String navn, int alder){
6     this.navn = navn;
7     this.alder = alder;
8   }
9   public int hentAlder(){
10    return alder;
11  }
12  public String hentNavn(){
13    return navn;
14  }
15  public void skrivUtInfo(){
16    System.out.println("Navn: " + navn);
17    System.out.println("Alder: " + alder);
18  }
19 }
```

```
1 class Katt extends Dyr{
2
3   private String bostedsadresse;
4
5   public Katt(String navn, int alder, String bostedsadresse){
6     super(navn, alder);
7     this.bostedsadresse = bostedsadresse;
8   }
9   public String hentBosted(){
10    return bostedsadresse;
11  }
12  @Override
13  public void skrivUtInfo(){
14    super.skrivUtInfo();
15    System.out.println(bostedsadresse);
16  }
17 }
```



Tre regler fra forelesning

- Et kall på super **må** legges **helt i begynnelsen av** konstruktøren.
- Kaller man ikke super eksplisitt, vil Java **selv legge inn kall på `super()`** helt først i konstruktøren når programmet kompileres.
- Hvis en klasse ikke har noen konstruktør, legger Java inn en tom konstruktør med kallet `super()`;

Abstract

```
abstract class Dyr {
```

```
protected String navn;  
protected int alder;
```

```
public Dyr(String navn, int alder){  
    this.navn = navn;  
    this.alder = alder;  
}
```

```
public String hentNavn(){  
    return navn;  
}
```

```
public int hentAlder(){  
    return alder;  
}
```

```
public abstract void lagLyd();
```

```
class Fjellrev extends Dyr {
```

```
public Fjellrev(String navn, int alder){  
    super(navn, alder);  
}
```

```
//siden lagLyd er abstract i superklassen maa vi implementer den i  
//sub klassen
```

```
public void lagLyd(){  
    System.out.println("What does the fox say?");  
}
```

```
}
```

Her er all koden som blir
brukt til å forklare hvis
dere vil se alt på samme
slide :))

```
class AbstractEksempel {
```

```
Run | Debug
```

```
public static void main(String[] args) {
```

```
//Det er lov til aa lage instanser av subclassen til en abstract klasse  
Katt katt = new Katt("Pus", 1, "Oslo");  
Fjellrev fjellrev = new Fjellrev("Tom", 13);
```

```
//Ikke lov til aa lage en instans av en abstract klasse  
Dyr dyr = new Dyr("Klara", 12);
```

```
}
```

```
Katt extends Dyr {
```

```
private String bosted;
```

```
public Katt(String navn, int alder, String bosted){  
    super(navn, alder);  
    this.bosted = bosted;  
}
```

```
public String hentBosted(){  
    return bosted;  
}
```

```
//siden lagLyd er abstract i superklassen maa vi implementer den i  
//sub klassen
```

```
public void lagLyd(){  
    System.out.println("Mjauuuuu");  
}
```

```
}
```

Abstract - Hva er det og hvordan lage det?

- **Hva er det?** En abstract klasse er en klasse som man ikke kan opprette en instans av, men som har subklasse(r) man kan opprette instanser av. Her er det f.eks. ikke naturlig å opprette en instans av Dyr fordi det finnes ikke noe som er dyr, kun subklasser av det.
- **Hvordan lage det?**
 - Man skriver abstract helt foran i klasse definisjonen
 - Inne i en abstract klasse kan man ha en abstract metode, en abstract er en metode som klassen ikke implementer selv, men som alle ikke abstracte subklassene må implementere. Dyr lager jo ingen lyd, men alle subklassene (Katt og Fjellrev lagere en lyd)

```
class AbstractEksempel {  
    Run | Debug  
    public static void main(String[] args) {  
        //Det er lov til aa lage instanser av subklassen til en abstract klasse  
        Katt katt = new Katt("Pus", 1, "Oslo");  
        Fjellrev fjellrev = new Fjellrev("Tom", 13); OK  
        //Ikke lov til aa lage en instans av en abstract klasse  
        Dyr dyr = new Dyr("Klara", 12); X  
    }  
}
```

```
abstract class Dyr {  
    protected String navn;  
    protected int alder;  
  
    public Dyr(String navn, int alder){  
        this.navn = navn;  
        this.alder = alder;  
    }  
  
    public String hentNavn(){  
        return navn;  
    }  
  
    public int hentAlder(){  
        return alder;  
    }  
  
    public abstract void lagLyd();  
}
```

Abstract - Ikke abstract subklasser

En ikke abstract subklasse av en abstract klasse MÅ implementere de abstracte metodene til superklassen sin.

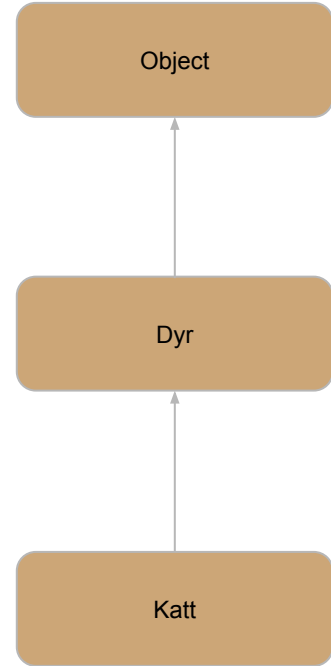
```
class Fjellrev extends Dyr {  
  
    public Fjellrev(String navn, int alder){  
        super(navn, alder);  
    }  
  
    //siden lagLyd er abstract i superklassen maa vi implementer den i  
    //sub klassen  
    public void lagLyd(){  
        System.out.println("What does the fox say?");  
    }  
}
```

```
class Katt extends Dyr {  
  
    private String bosted;  
  
    public Katt(String navn, int alder, String bosted){  
        super(navn, alder);  
        this.bosted = bosted;  
    }  
  
    public String hentBosted(){  
        return bosted;  
    }  
  
    //siden lagLyd er abstract i superklassen maa vi implementer den i  
    //sub klassen  
    public void lagLyd(){  
        System.out.println("Mjauuuuu");  
    }  
}
```

Object, og metodene toString og equals

Object er eller klassers superklasse!

Og den har metoder man kan overskrive, bl.a. toString og equals!





Oppmøteregistrering:
<https://nettskjema.no/a/184340>



Breakoutrooms

1. Slå på kamera og ha en presentasjonsrunde
2. Diskuter:
 - a. Hva betyr Abstract (Google gjerne eller se i forelesning hvis dere ikke husker!)
 - b. Hvorfor er polymorfi lurt ?
3. Jobb sammen med ukesoppgavene, de ligger på emnesiden -> grupper
 - a. Enten ved at én deler skjerm med codecollab.io/
 - b. OBS: codecollab er gratistjenester som UiO ikke har avtale med, sannsynligvis vil de samle data om dere. Dere kan fint løse oppgavene uten å bruke disse tjenestene!
4. Bruk "ask for help"-knappen for å få hjelp 😊
5. Vi møtes her igjen for å gå gjennom oppgavene til slutt (dere bestemmer hvilke)

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

Jobbe med oppgaver

Ris, ros, forslag ?

<https://nettskjema.no/a/181775>