IN1010 - Seminar 4

- Arv og Polymorfi

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

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

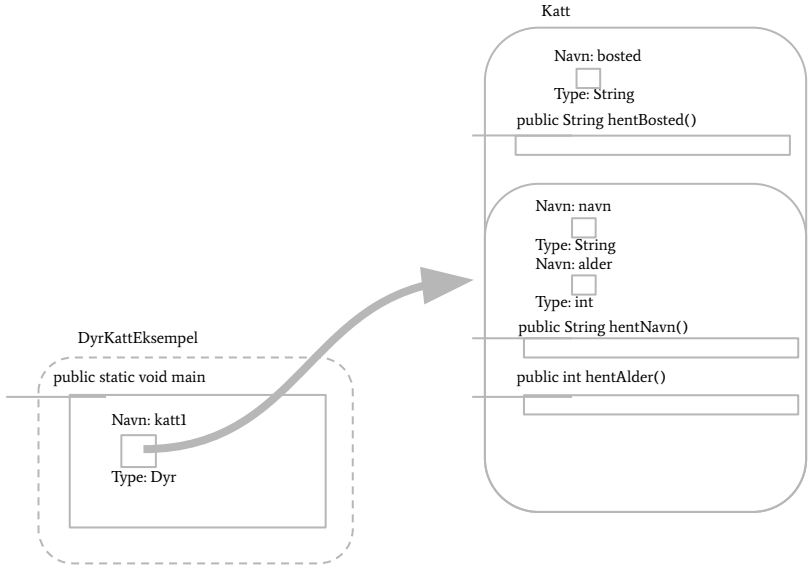
Casting eksempel

```java
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;
    }
}
```

```java
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;
    }
}
```

```java
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;
    }
}
```

```java
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;
    }
}
```

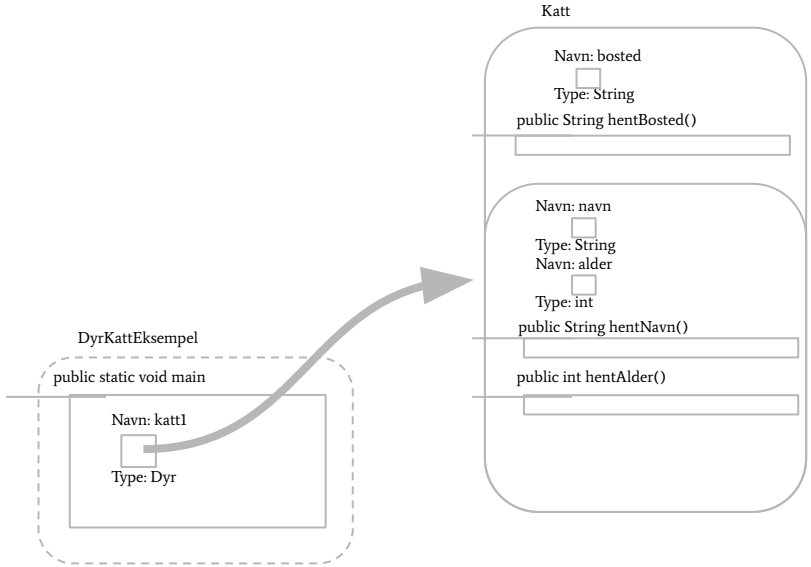



```java
class DyrKattEksempel{

    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())
    }
}
```

```java
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;
    }
}
```

```java
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

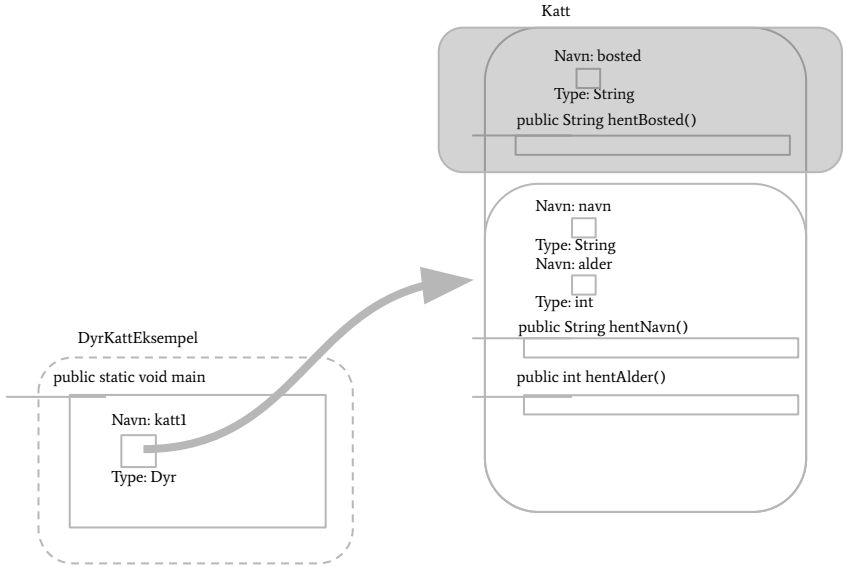



```java
class DyrKattEksempel{

    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())
    }
}
```

```java
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;
    }
}
```

```java
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

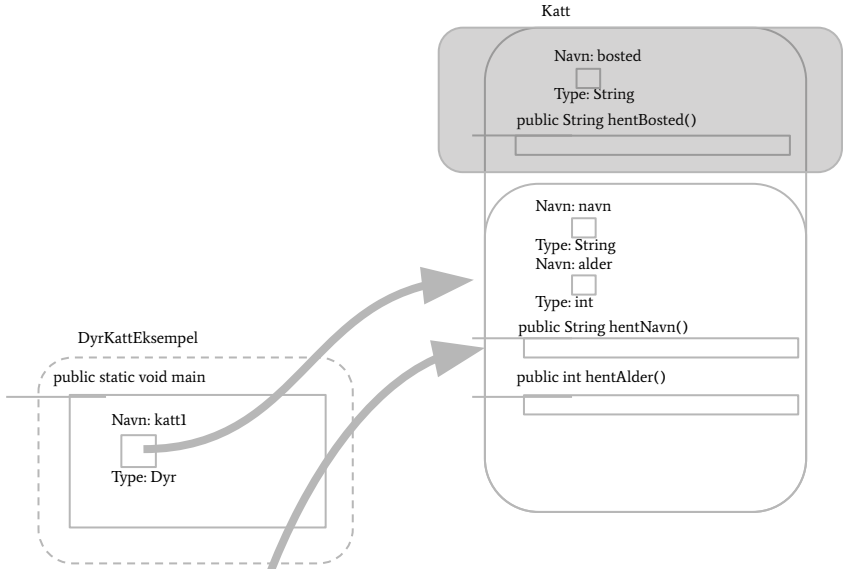



```java
class DyrKattEksempel{

    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())
    }
}
```

```java
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;
    }
}
```

```java
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

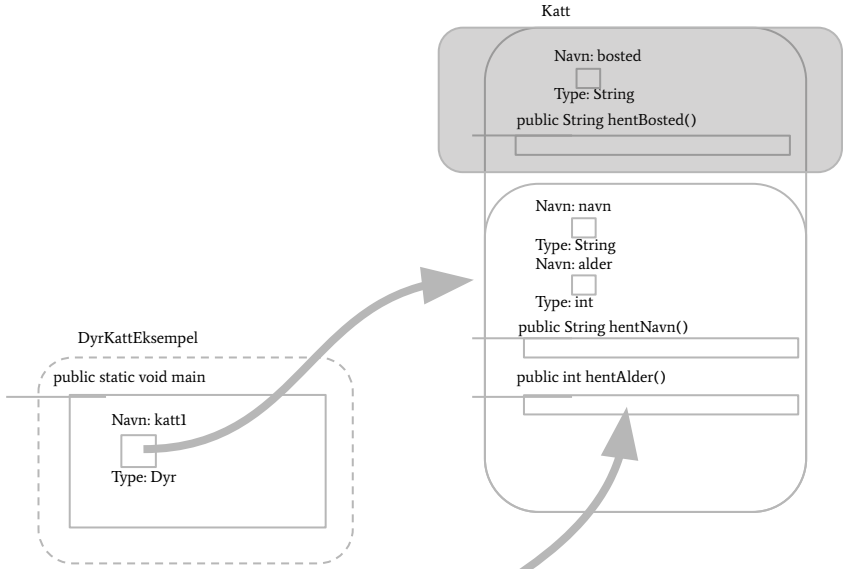



```java
class DyrKattEksempel{

    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())
    }
}
```

```java
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;
    }
}
```

```java
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

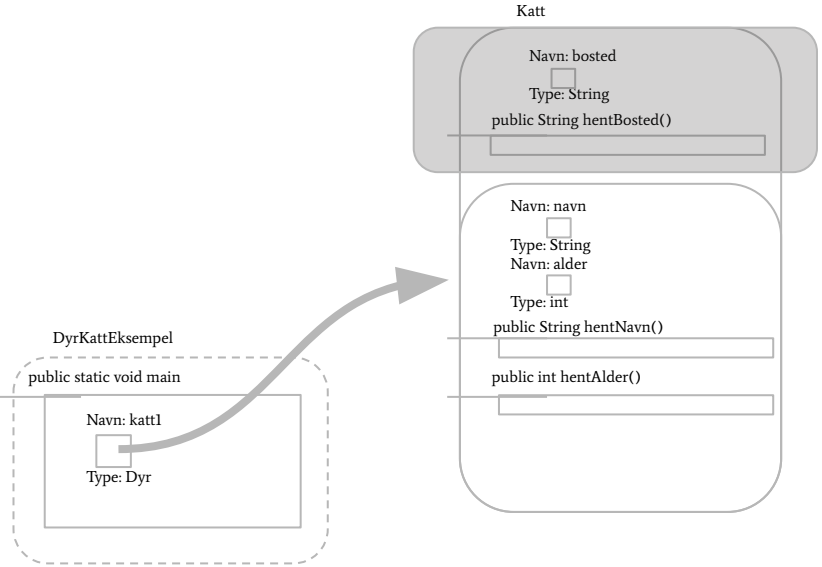



```java
class DyrKattEksempel{

    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())
    }
}
```

```java
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;
    }
}
```

```java
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!



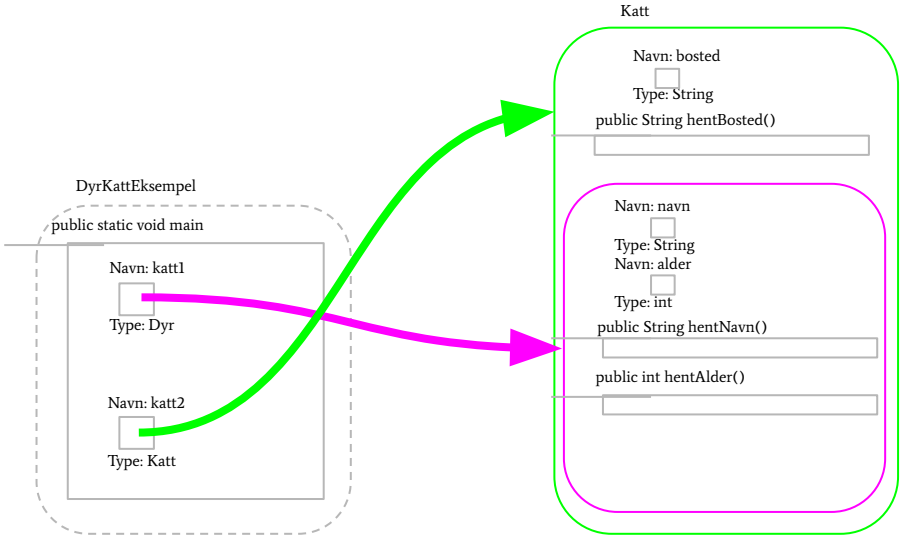

```java
class DyrKattEksempel{

    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

```java
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;
    }
}
```

```java
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;
    }
}
```

```java
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());

    }
}
```

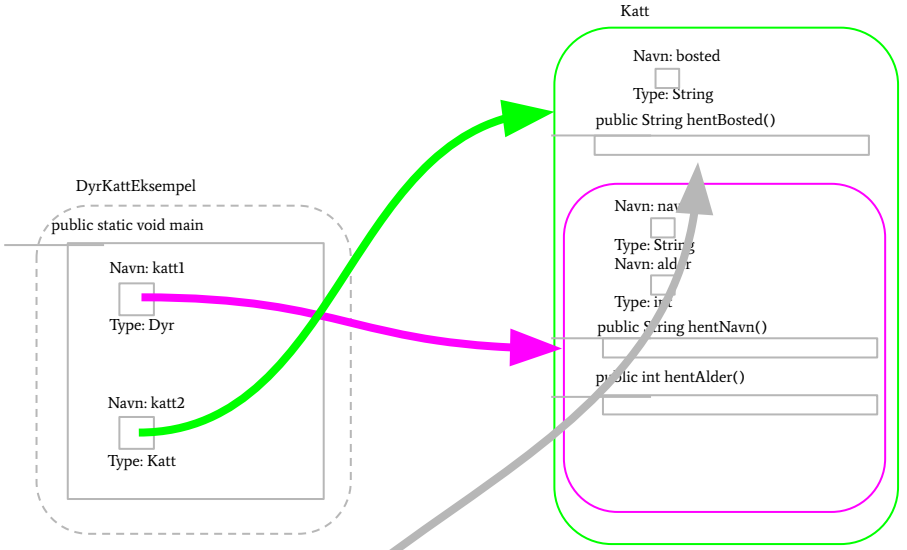


Når vi nå ser på katte objektet men katt2 sine briller får vi også alle egenskapene til en Katt. (Det som er markert i grønt)

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

```java
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;
    }
}
```

```java
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;
    }
}
```



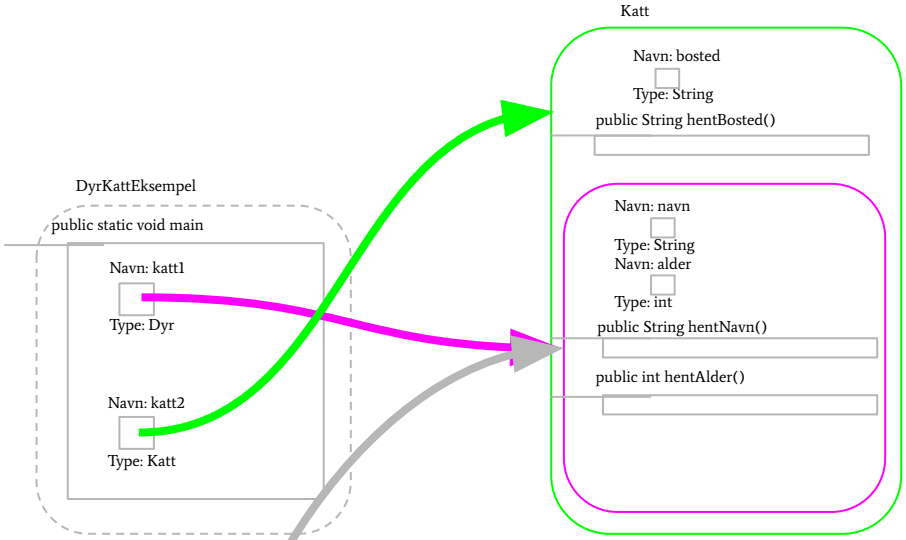

```java
class DyrKattEksempel{

    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

```java
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;
    }
}
```

```java
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;
    }
}
```

```java
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

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

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

super() konstruktør

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

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

super() konstruktør

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

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

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

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

super() konstruktør

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

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

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

Kaller på katt.skrivUtInfo().

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

super() konstruktør

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

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

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

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

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

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

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

Bostedsadresse blir ikke skrevet ut

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

.super

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

  public Katt(String navn, int alder, String bostedsadresse){
    super(navn, alder);
    this.bostedsadresse = bostedsadresse;
  }
  public String hentBosted(){
    return bostedsadresse;
  }
  public void skrivUtInfo(){
    super.skrivUtInfo();
    System.out.println(bostedsadresse);
  }
}
```

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

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

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

  public Katt(String navn, int alder, String bostedsadresse){
    super(navn, alder);
    this.bostedsadresse = bostedsadresse;
  }
  public String hentBosted(){
    return bostedsadresse;
  }
  public void skrivUtInfo(){
    super.skrivUtInfo();
    System.out.println(bostedsadresse);
  }
}
```

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

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

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

  public Katt(String navn, int alder, String bostedsadresse){
    super(navn, alder);
    this.bostedsadresse = bostedsadresse;
  }
  public String hentBosted(){
    return bostedsadresse;
  }
  @Override
  public void skrivUtInfo(){
    super.skrivUtInfo();
    System.out.println(bostedsadresse);
  }
}
```

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

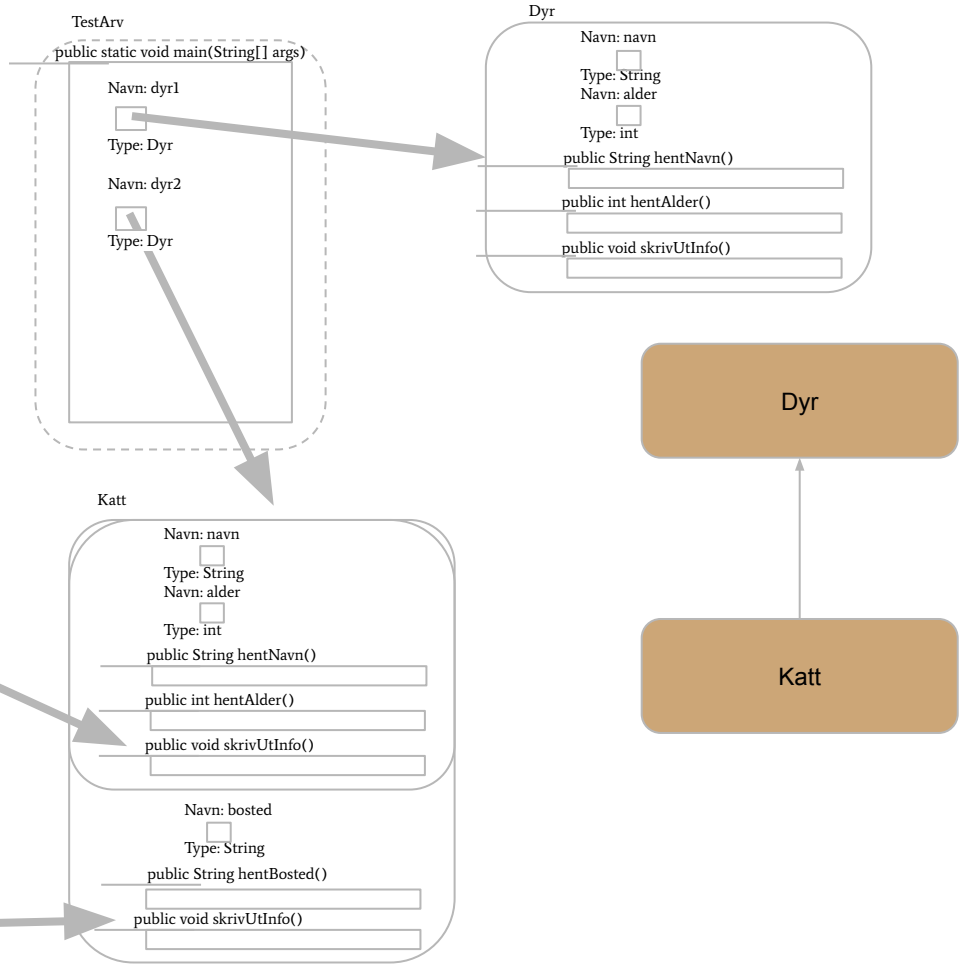
Men vi burde bruke @Override for å være sikker på at programmet vårt skjønner at skrivUtInfo i Katt skal overskrive skrivUtInfo() i Dyr.

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

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

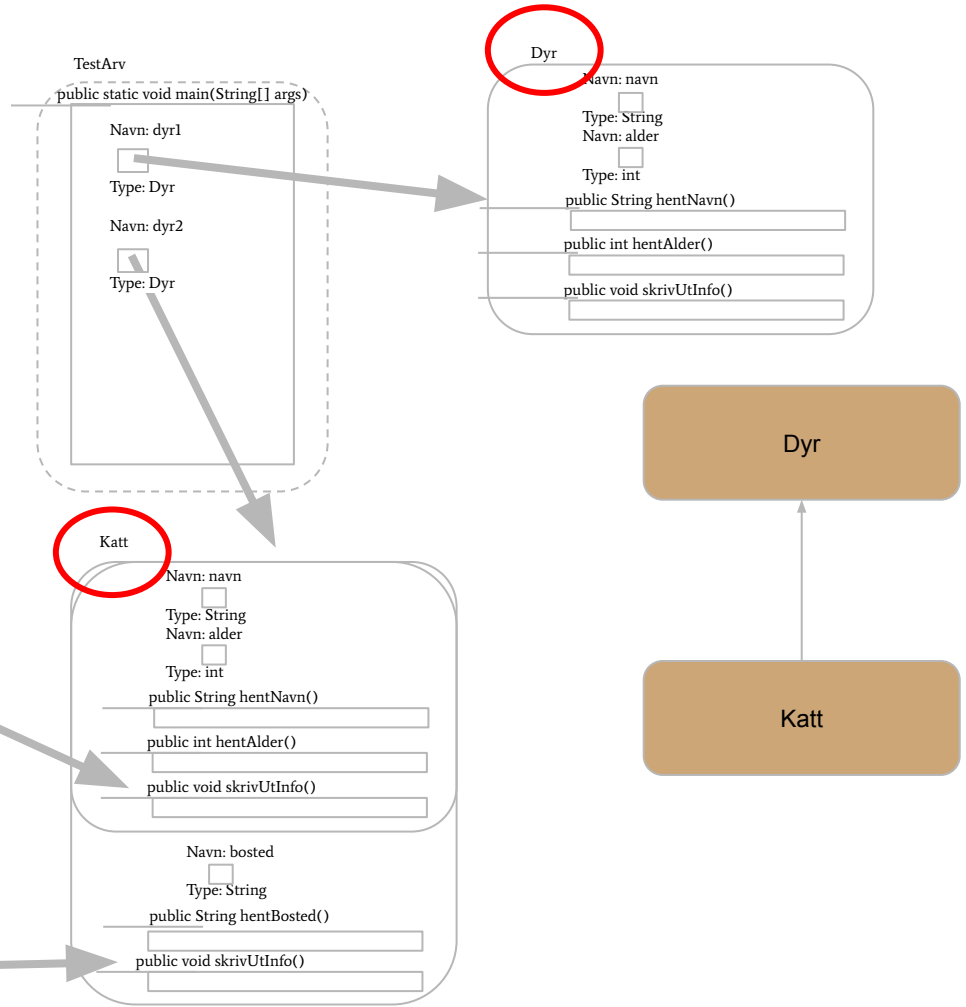
```java
class Katt extends Dyr{

  private String bostedsadresse;

  public Katt(String navn, int alder, String bostedsadresse){
    super(navn, alder);
    this.bostedsadresse = bostedsadresse;
  }
  public String hentBosted(){
    return bostedsadresse;
  }
  @Override
  public void skrivUtInfo(){
    super.skrivUtInfo();
    System.out.println(bostedsadresse);
  }
}
```



Polymorfi

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

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

  public Katt(String navn, int alder, String bostedsadresse){
    super(navn, alder);
    this.bostedsadresse = bostedsadresse;
  }
  public String hentBosted(){
    return bostedsadresse;
  }
  @Override
  public void skrivUtInfo(){
    super.skrivUtInfo();
    System.out.println(bostedsadresse);
  }
}
```



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

```java
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();
}
```

```java
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 :))

```java
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");
    }
}
```

```java
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);

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

}
```

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)

```java
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();
}
```

```java
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);

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

}
```

OK

X

Abstract - Ikke abstract subklasser

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

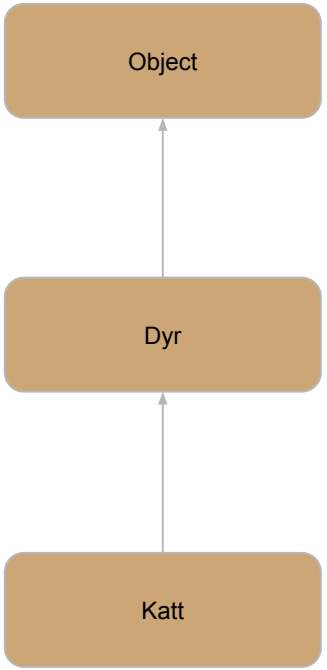
```java
class Katt extends Dyr{

    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");
    }
}
```

```java
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?");
    }
}
```

Object, og metodene toString og equals

Object er eller klassers superklasse!

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



Send Johanna en direktemelding i chatten

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

Svar gjerne også om svaret skulle være nei

Repetisjon forrige uke

Breakoutrooms

1. Slå på kamera og ha en presentasjonsrunde
2. Diskuter:
   a. Hva betyr det at en variabel med type Dyr kan ha en referanse til et objekt av klassen Katt, men ikke omvendt?
      (Variabel av type A kan referere til objekt av B, men ikke omvendt)
   b. Hva betyr kodeordet Protected? (Google gjerne eller se i forelesning hvis dere ikke husker!)
      Hva er forskjellen på private, public og protected?
   c. Hva gjør metoden instanceOf()?
3. Jobb sammen med ukesoppgavene, de ligger på emnesiden -> grupper
   a. Enten ved at én deler skjerm eller med jdoodle.com eller med codecollab.io/
   b. OBS: både jdoodle og codecollab er gratistjeneseter 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)

Jobbe med oppgaver

Ris, ros, forslag ?

https://nettskjema.no/a/180345