Femte seminar

IN1010 - GRUPPE 2

18 feb 2025

this week>

- Arv og komposisjon
  - Referansetyper
  - Abstrakte klasse
  - Når bruker man hva?
- Polymorfi
  - @Override
  - Overriding vs overloading
  - toString() og equals()
- Seminaroppgaver

Hva husker dere fra arv og subklasser?

- BRUK 1 MINUTT PÅ Å SKRIVE NED ALT DERE HUSKER FRA ARV OG SUBKLASSER **INDIVIDUELT**
- **DISKUTER SAMMEN** HVA DERE HUSKER

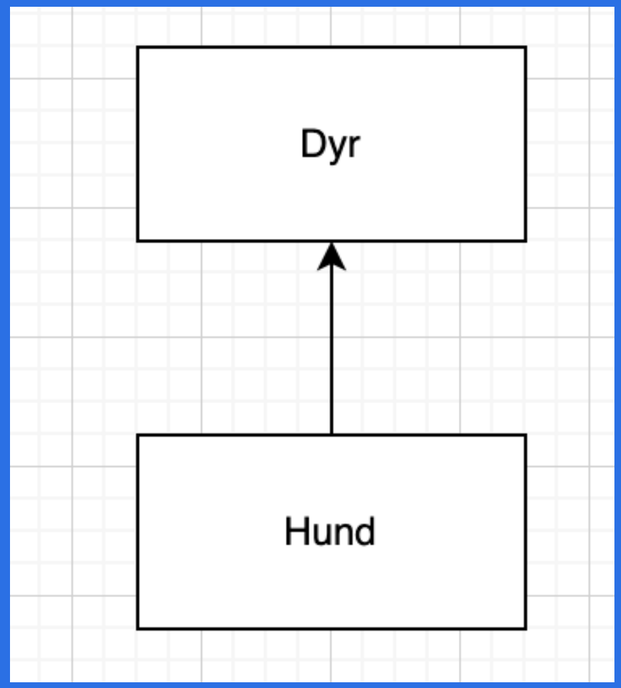
Referansetyper

```java
class Dyr{
    public void skrivUt(){
        System.out.println(x:"Jeg er et dyr");
    }
}

class Hund extends Dyr{
    @Override
    public void skrivUt(){
        System.out.println(x:"Jeg er en hund");
    }
}

class referansetype{
    Run | Debug
    public static void main(String[] args){
        Dyr obj1 = new Hund();
        obj1.skrivUt();

    }
}
```



<---Hva blir printet ut her?

HVORFOR ER DET RØD STREK?

```java
class Dyr{
    public void skrivUt(){
        System.out.println(x:"Jeg er et dyr");
    }
}

class Hund extends Dyr{
    @Override
    public void skrivUt(){
        System.out.println(x:"Jeg er en hund");
    }

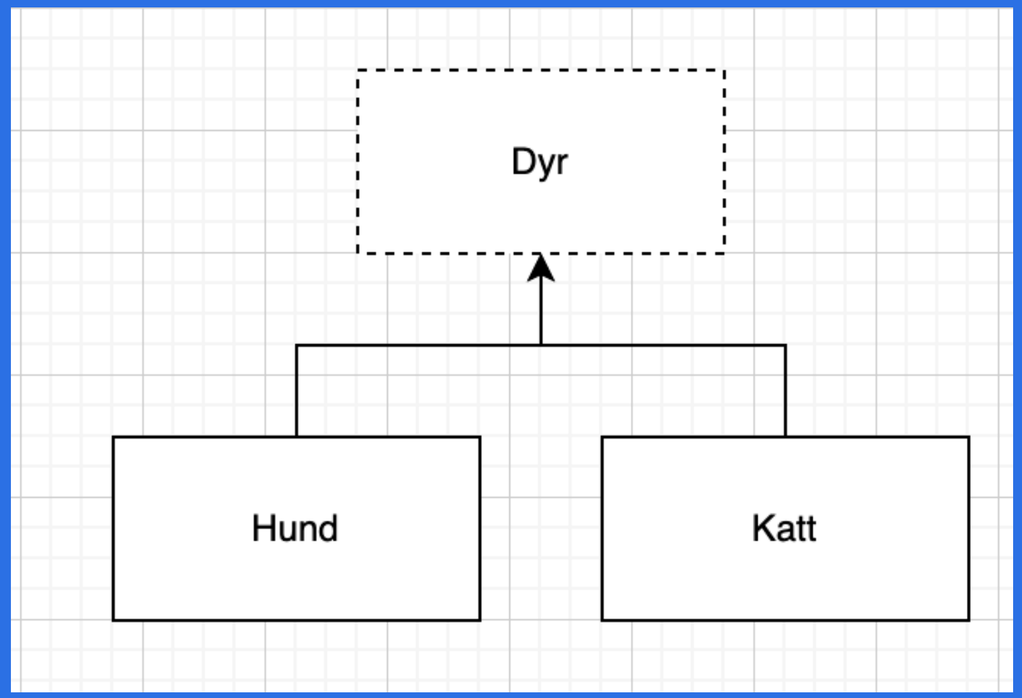
    public void snakk(){
        System.out.println(x:"Woff");
    }
}

class referansetype{
    Run | Debug
    public static void main(String[] args){
        Dyr obj1 = new Hund();
        obj1.skrivUt(); // Jeg er en hund
        obj1.snakk();

    }
}
```

- obj1 har referansetype av Dyr
- selv om objekt-typen er Hund så blir den sett på som en instans av dyr pga referansen
- typekonverterer hvis man ønsker “å låse opp” hund-egenskapene

Abstrakte klasser



Kan det opprettes instans av Dyr-klassen nå?

- abstrakt betyr at man ikke kan opprette objekt av klassen
- brukes som en en “blueprint” /”skall” for subklassene

```java
abstract class Dyr{
    int antBein;

    public Dyr(int bein){
        antBein = bein;
    }

    abstract void lagLyd();
}

class Hund extends Dyr{

    public Hund(int bein){
        super(bein); //Sender parameter til Dyr-klassen
    }

    @Override
    void lagLyd() {
        System.out.println(x:"Woff");
    }
}

class r2{
    Run | Debug
    public static void main(String[] args){
        Dyr obj1 = new Hund(bein:4);
        obj1.lagLyd(); //printes ut woff

    }
}
```

SUPER

- .super brukes for å akksesere variabler/metoder i superklassen
- super() kaller på konstruktøren til superklassen, og sender parameterne videre opp i klassehierakiet

Hva skal man velge mellom arv og komposisjon?

- Tydelig sammenheng mellom klasser
- Gjenbruk av metoder
- Tett kobling
- f.eks Dyr, pattedyr, hund, katt,

- Komponert basert design
- Fleksibelt
- Lav kobling
- f.eks innlevering1 game of life
  - Celle, Rutenett, Verden osv.

polymorfi

- Når en subklasse overskriver en eksisternede metode i superklassen
  - Polymorfe metoder
- @Override brukes for å vise hvilke metoder som ble omdefinert

Override vs Overload

- Omdefinerer en metode fra en subklasse
- Den nederste metoden i subklassen i hierakiet blir brukt

```java
class Dyr{
    public void skrivUt(){
        System.out.println(x:"Jeg er et dyr");
    }
}

class Hund extends Dyr{
    @Override
    public void skrivUt(){
        System.out.println(x:"Jeg er en hund");
    }
```

- Metoder med samme navn, men ulik signatur (antall og type paramter)
- Gjelder for både subklasser og i samme klasser
- Avhengig av hvilken signatur du bruker når du kaller på metoden

```java
    // Overbelastet metode for å beregne omkrets for et rektangel
    double omkrets() {
        return 2 * (lengde + bredde);
    }

    // Ny overbelastet metode som tar lengde og bredde som parametere
    double omkrets(double lengde, double bredde) {
        return 2 * (lengde + bredde);
    }
```

toString()

- Definert metode i Object-klassen (allers klasses superklasse)
  - Alle klasser arver derfor metoden
- Gir en streng-representasjon av et objekt
  - Standard -> skriver ut navnet på objektet
- Kan omdefinere den til å skrive ut hva vi ønsker

```
@Override
public String toString() {
    return "Person{navn='" + navn + "', alder=" + alder + "}";
}
```

Seminaroppgaver