Arv og Subklasser

I dag:

- Enhetstesting
- Arv og subklasser
- Abstrakte klasser
- Private, public, protected
- Repetisjon av ArrayList, HashMap og hvordan man kan bruke foreach-løkke

Info

- **Oblig 1 har frist mandag 5. februar kl 23.59**
- Kan be ut utsettelse via nettskjema på semestersiden → obligatoriske innleveringer
- Husk å lese testfilene for riktig variabelnavn

- AstroDiscourse er nå oppe

- Mentimeter fra neste uke<3 (forhåpentligvis)

Enhetstesting

- Test hver del eller hvert objekt for seg
- Etter å ha testet hver del kan man teste det sammen (=integrasjonstesting)
- Prøv flere ulike rekkefølger på kall.
- Spesielt viktig å sjekke edge-cases:
  - Hva skjer hvis man prøver å hente ut noe fra en tom liste?
  - Hva skjer hvis en variabel har verdi null?
  - Hva skjer hvis man prøver å sette inn noe i et fullt array?

```java
public class Teller {

    int tall;

    public Teller(int starttall){
        tall = starttall;
    }

    void tellOpp(){
        tall ++;
    }

    void tellNed(){
        tall--;
    }

    void hentTeller(){
        System.out.println(tall);
    }
```

```java
    Run | Debug
    public static void main(String[] args) {
        //en teller som starter på 0:
        Teller t = new Teller(starttall:0);

        //sjekker verdien før metoden:
        t.hentTeller();

        //kaller metode som skal endre t sitt tall:
        t.tellOpp();

        //sjekker om verdien er den forventede verdien:

        t.hentTeller();

    }
}
```

Arv og subklasser

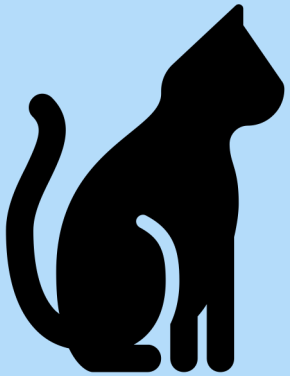
Subklasse og superklasse

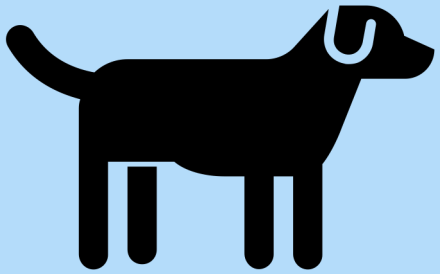
- En **subklasse** er en klasse som bygger på en allerede spesifisert klasse (= **superklasse**)
- Subklassen **arver** egenskaper fra superklassen
- Subklassen er en «utvidelse» av superklassen («extends»)

Skrives: Class <subklasse> extends <superklasse>

Eksempel

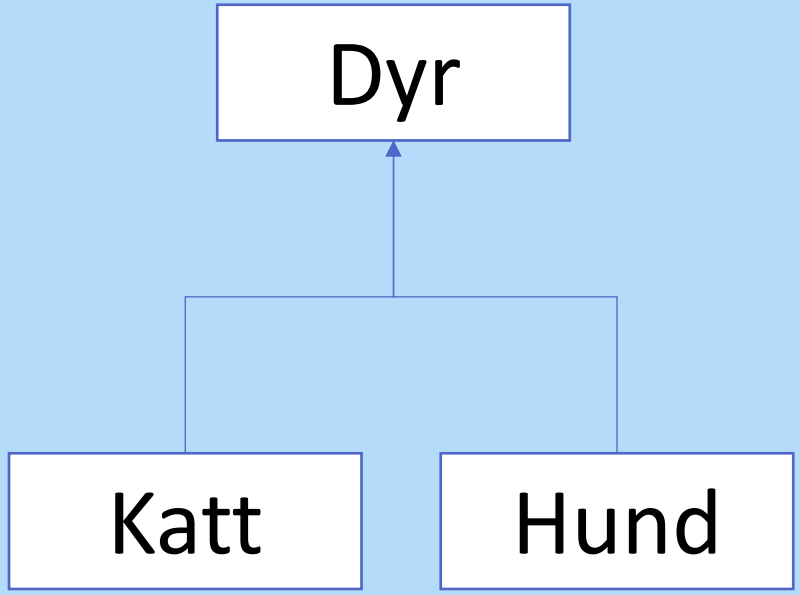
- En *superklasse* **Dyr**
- To *subklasser* **Katt** og **Hund**
- Alle dyr har noe til felles.
  - For eksempel spiser de noe
- Ulike typer dyr kan ha ulike egenskaper i tillegg til det som er felles.
  - Feks maler en katt, og en hund bjeffer.





Klassehierarki

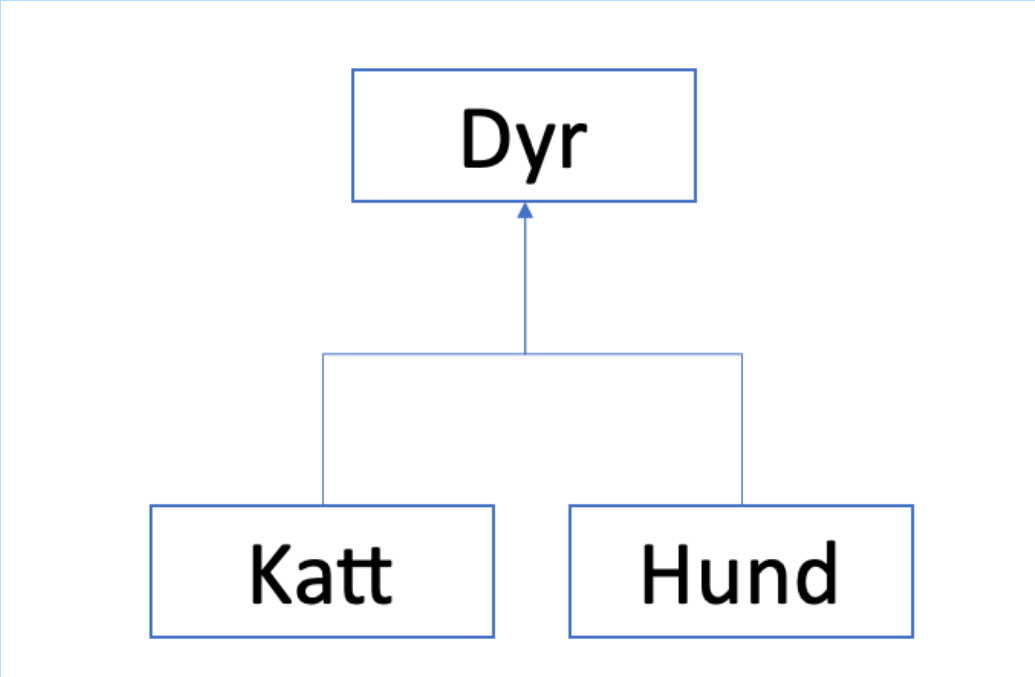
- Superklasser og subklasser danner et hierarki
  - Som kategorier og underkategorier
  - Underkategorier, kan igjen ha underkategorier.
- Dette kan tegnes med **klassehierarki**
  - Nyttig for å holde oversikt



```
public class Dyr {…
```

```
public class Hund extends Dyr{…
```

```
public class Katt extends Dyr{…
```

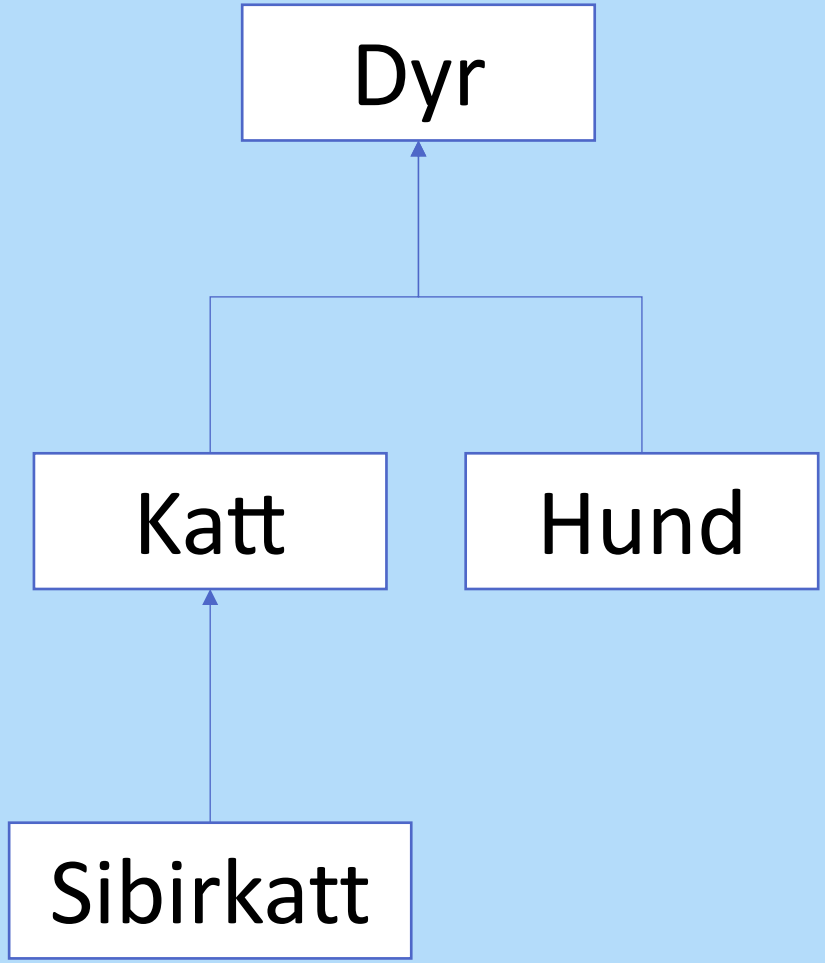


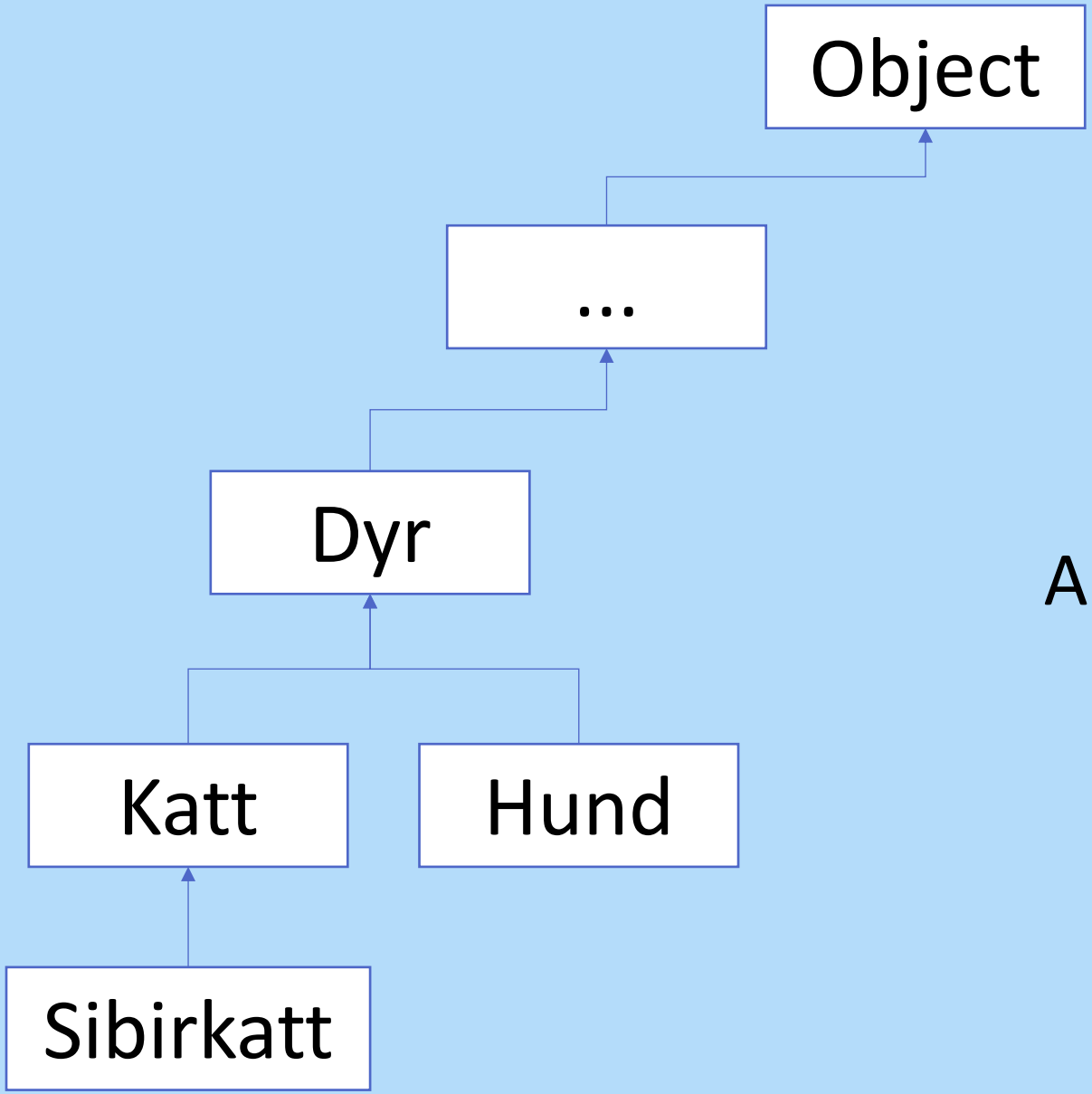
```
public class Dyr {…
```

```
public class Hund extends Dyr{…
```

```
public class Katt extends Dyr{…
```

```
public class Sibirkatt extends Katt{…
```





Alle klasser arver fra Object

Superklassen Dyr

- Alle dyr må ha et navn (i dette eksempelet)
  - Instansvariabelen navn
- Alle dyr kan spise
  - Metoden spis()

```java
public class Dyr {

    String navn;

    Dyr(String navn){
        this.navn = navn;
    }

    void spis(){
        System.out.println(x:"Jeg spiser.");
    }
}
```

Egenskaper som er spesielle for spesifikke typer dyr legges til i subklassene de tilhører.



Subklassen Katt

- Katt arver fra Dyr
- Alle objekter av klassen Katt må få et navn når de opprettes
- Instansvariabelen «navn» arves fra Dyr → bruker super(navn) i konstruktøren.

```java
public class Dyr {

    String navn;

    Dyr(String navn){
        this.navn = navn;
    }

    void spis(){
        System.out.println(x:"Jeg spiser.");
    }
}
```

```java
public class Katt extends Dyr{

    Katt(String navn) {
        super(navn);
    }

}
```

super() i konstruktøren

- Brukes i konstruktøren til en subklasse for å initialere instansvariabler som arves fra superklassen.
- Kaller superklassen sin konstruktør med subklassen sin(e) verdier for variablene.
- Kan initialisere subklasse-spesifikke instansvariabler under.

```java
public class Person {

    //instansvariabler:
    String navn;
    int alder;

    //konstruktør:
    public Person(String navn, int alder){
        this.navn = navn;
        this.alder = alder;
    }
}
```

```java
public class Student extends Person{

    //instansvariabler for studenter, men ikke alle personer:
    String linje;
    int trinn;

    public Student(String navn, int alder, String linje, int trinn) {
        super(navn, alder);
        this.linje = linje;
        this.trinn = trinn;
    }
}
```



Subklassen Katt forts.

- Kan tilpasse metoden spis() til å passe til katter
- Bruker **@Override**
- Samme metodesignatur
- Ulikt innhold

```java
public class Dyr {

    String navn;

    Dyr(String navn){
        this.navn = navn;
    }

    void spis(){
        System.out.println(x:"Jeg spiser.");
    }
}
```

```java
public class Katt extends Dyr{

    Katt(String navn) {
        super(navn);
    }

    @Override
    void spis() {
        System.out.println(x:"jeg spiser kattemat");
    }
```



Katt forts.

- Kan legge til en egen metode for at en katt skal male.
- En instans av Dyr som ikke er Katt vil ikke ha mal()-metoden

```java
public class Katt extends Dyr{

    Katt(String navn) {
        super(navn);
    }

    @Override
    void spis() {
        System.out.println(x:"jeg spiser kattemat");
    }

    void mal() {
        System.out.println(x:"mjau mjau");
    }
}
```

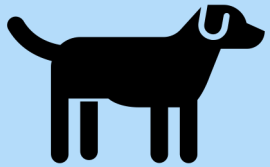
Tilsvarende for Hund

```java
public class Hund extends Dyr{


    Hund(String navn) {
        super(navn);
    }

    @Override
    void spis() {
        System.out.println(x:"Jeg spiser hundemat");
    }

    void bjeff(){
        System.out.println(x:"voff voff");
    }
}
```

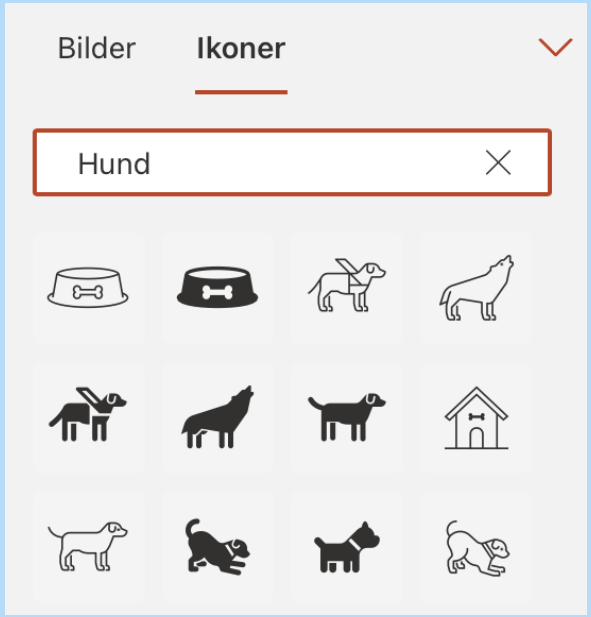


Abstrakte Klasser

- Men gir det egentlig mening å implementere et objekt av klassen Dyr?
- Noen ganger ønsker man en superklasse som inneholder de instansvariablene og metodene som er felles for subklassene, men uten at denne superklassen kan implementeres.







Abstrakte Klasser forts.

- Bruker «abstract»
- Nå er det **ikke mulig å opprette et dyreobjekt**:
  - ~~Dyr d = new Dyr(«Jack»)~~
- Man kan fortsatt opprette katte- og hundeobjekter:
  - Katt k = new Katt(«Jack»)
  - Hund h = new Hund(«Jackie»)

```java
public abstract class Dyr {

    String navn;


    Dyr(String navn){
        this.navn = navn;
    }

    void spis(){
        System.out.println(x:"Jeg spiser.");
    }
}
```

instanceof

- Brukes for å sjekke om et objekt er av en spesifikk type/klasse

```
Run | Debug
public static void main(String[] args) {
    Hund h = new Hund(navn:"Luna");
    Katt k = new Katt(navn:"Jack");
    Sibirkatt s = new Sibirkatt(navn:"Gustav");

    System.out.println("Er Luna av typen Hund? "+ (h instanceof Hund));
    System.out.println("Er Jack av typen Katt? "+ (k instanceof Katt));
    System.out.println("Er Gustav av typen Hund? "+ (s instanceof Sibirkatt));
```

```
Er Luna av typen Hund? true
Er Jack av typen Katt? true
Er Gustav av typen Hund? true
```

```
Run | Debug
public static void main(String[] args) {
    Hund h = new Hund(navn:"Luna");
    Katt k = new Katt(navn:"Jack");
    Sibirkatt s = new Sibirkatt(navn:"Gustav");

    System.out.println("Er Luna av typen Hund? "+ (h instanceof Hund));
    System.out.println("Er Jack av typen Katt? "+ (k instanceof Katt));
    System.out.println("Er Gustav av typen Hund? "+ (s instanceof Sibirkatt));


    System.out.println("Er Luna av typen Dyr? "+ (h instanceof Dyr));
    System.out.println("Er Jack av typen Dyr? "+ (k instanceof Dyr));
    System.out.println("Er Gustav av typen Dyr? "+ (s instanceof Dyr));

}
```

```
Er Luna av typen Hund? true
Er Jack av typen Katt? true
Er Gustav av typen Hund? true
Er Luna av typen Dyr? true
Er Jack av typen Dyr? true
Er Gustav av typen Dyr? true
```

Public, private, protected

- **Public**: alle klasser kan se.
- **Protected**: alle subklasser kan se.
- **Private**: ingen klasser eller subklasser kan se, kun klassen selv.

ArrayList

- Klasse i Java-biblioteket
- Størrelsen må ikke oppgis ved opprettelse.
- Størrelsen kan endres ved behov.
- Noen metoder:
  - **size**()
  - **isEmpty**()
  - **contains**(element)
  - **get**(i)
  - **set**(i, element)
  - **add**(element)/**add**(i, element)

```
public static void main(String[] args) {
    ArrayList<Integer> tall = new ArrayList<>();
    tall.add(e:0);
    tall.add(e:1);
    tall.add(e:3);

    ArrayList<Katt> katter = new ArrayList<>();
    katter.add(new Katt(navn:"Ole"));
    katter.add(new Katt(navn:"Dole"));
    katter.add(new Katt(navn:"Doffen"));

}
```

For each med ArrayList

```java
ArrayList<Katt> katter = new ArrayList<>();
katter.add(new Katt(navn:"Ole"));
katter.add(new Katt(navn:"Dole"));
katter.add(new Katt(navn:"Doffen"));

for (Katt katt : katter) {
    System.out.println(katt.navn);
}
```

```
Ole
Dole
Doffen
```

HashMap

- Som Dictionary i Python
- Lagrer pekere til objekter uten en bestemt rekkefølge (et sett).
- Består av nøkkel-verdi-par
- Hvert objekt som legges inn må ha en tilhørende nøkkel for å gjøre oppslag i hashmapen.
- Noen metoder:
  - put(key, value)
  - get(key)
  - remove(key)
  - keySet()

```
Katt k1 = new Katt(navn:"Ole");
Katt k2 = new Katt(navn:"Dole");
Katt k3 = new Katt(navn:"Doffen");

HashMap<String, Katt> kattene = new HashMap<>();
kattene.put(k1.navn, k1);
kattene.put(k2.navn, k2);
kattene.put(k3.navn, k3);
```

For each over HashMap

```
Katt k1 = new Katt(navn:"Ole");
Katt k2 = new Katt(navn:"Dole");
Katt k3 = new Katt(navn:"Doffen");

HashMap<String, Katt> kattene = new HashMap<>();
kattene.put(k1.navn, k1);
kattene.put(k2.navn, k2);
kattene.put(k3.navn, k3);

//for each på hashmap - itererer over settet med nøkler
//for å aksesere verdien knyttet til nøkkelen
for (String kattenavn : kattene.keySet()) {
    System.out.println(kattene.get(kattenavn));
}
```

```
Katt@6b95977
Katt@7e9e5f8a
Katt@8bcc55f
```

For each over HashMap

```java
Katt k1 = new Katt(navn:"Ole");
Katt k2 = new Katt(navn:"Dole");
Katt k3 = new Katt(navn:"Doffen");

HashMap<String, Katt> kattene = new HashMap<>();
kattene.put(k1.navn, k1);
kattene.put(k2.navn, k2);
kattene.put(k3.navn, k3);

//for each på hashmap - itererer over settet med nøkler
//for å aksesere verdien knyttet til nøkkelen
for (String kattenavn : kattene.keySet()) {
    kattene.get(kattenavn).spis();
}
```

```
jeg spiser kattemat
jeg spiser kattemat
jeg spiser kattemat
```

Oppgaver!

- Seminaroppgaver: Emneside → grupper → gruppe3 → uke2
- Ukesoppgaver
- Oblig