ARV OG SUBKLASSER 1

IN1010 gruppe 3

agneshb@uio.no

I dag:

- Arv og subklasser
- Abstrakte klasser
- Referansetyper

Info

- **Innlevering 1 har frist i dag, 12. februar kl 23.59**
- Kan be ut utsettelse via nettskjema på semestersiden → obligatoriske innleveringer
- Husk å lese testfilene for riktig variabelnavn

Subklasse og superklasse

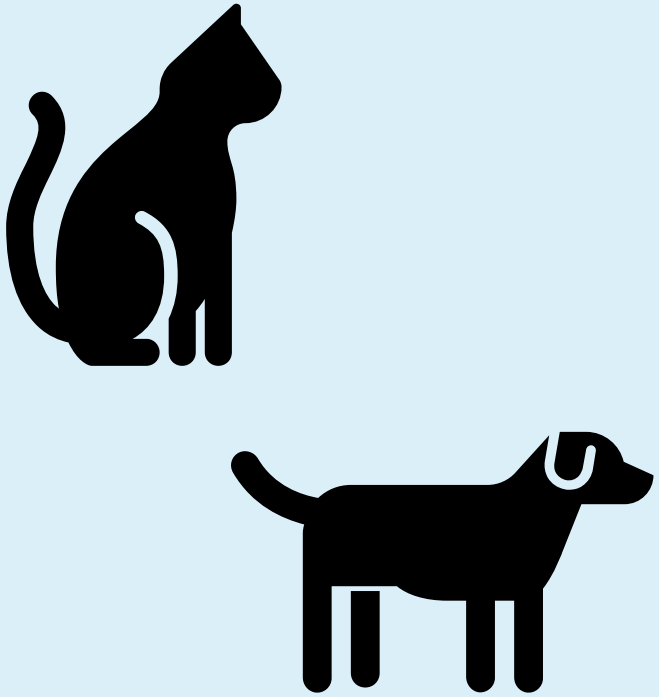
- En **subklasse** er en klasse som bygger på en allerede spesifisert klasse (= **superklasse**)
- Subklassen **arver** egenskaper fra superklassen
- Subklassen er en «utvidelse» av superklassen («extends»)

Skrives:

```
class <subklasse> extends <superklasse>
```

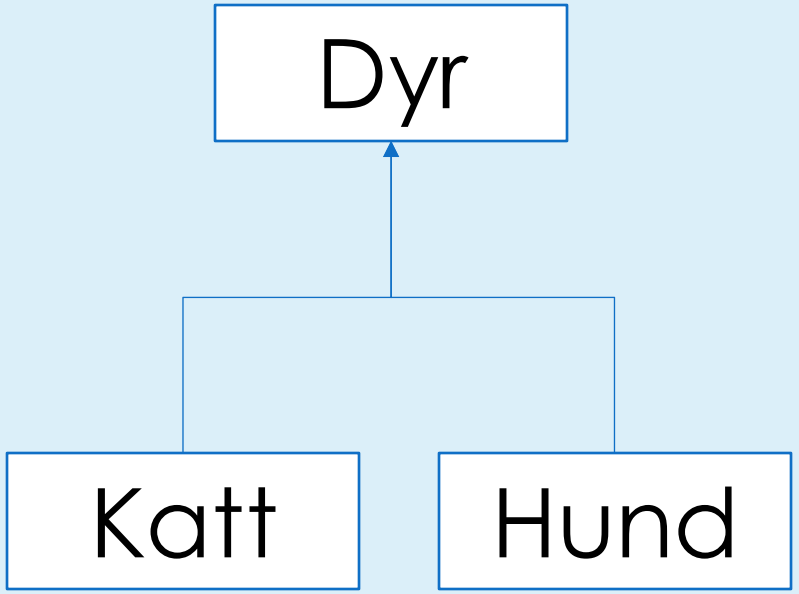
Eksempel

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



Klassehierarki

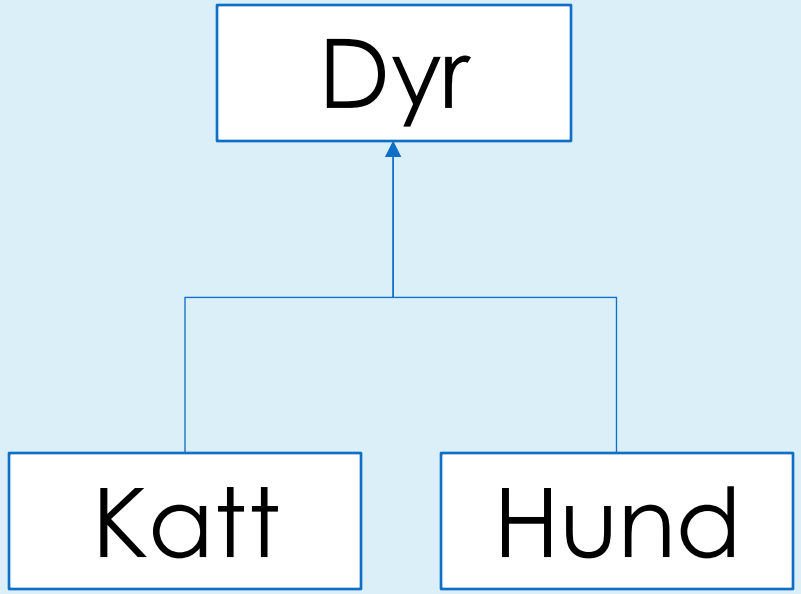
- Superklasser og subklasser danner et hierarki
  - Som kategorier og underkategorier
  - Underkategorier, kan igjen ha underkategorier.
- Dette kan tegnes med **klassehierarki**
  - Nyttig for å holde oversikt



```
public class Dyr {…
```

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

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

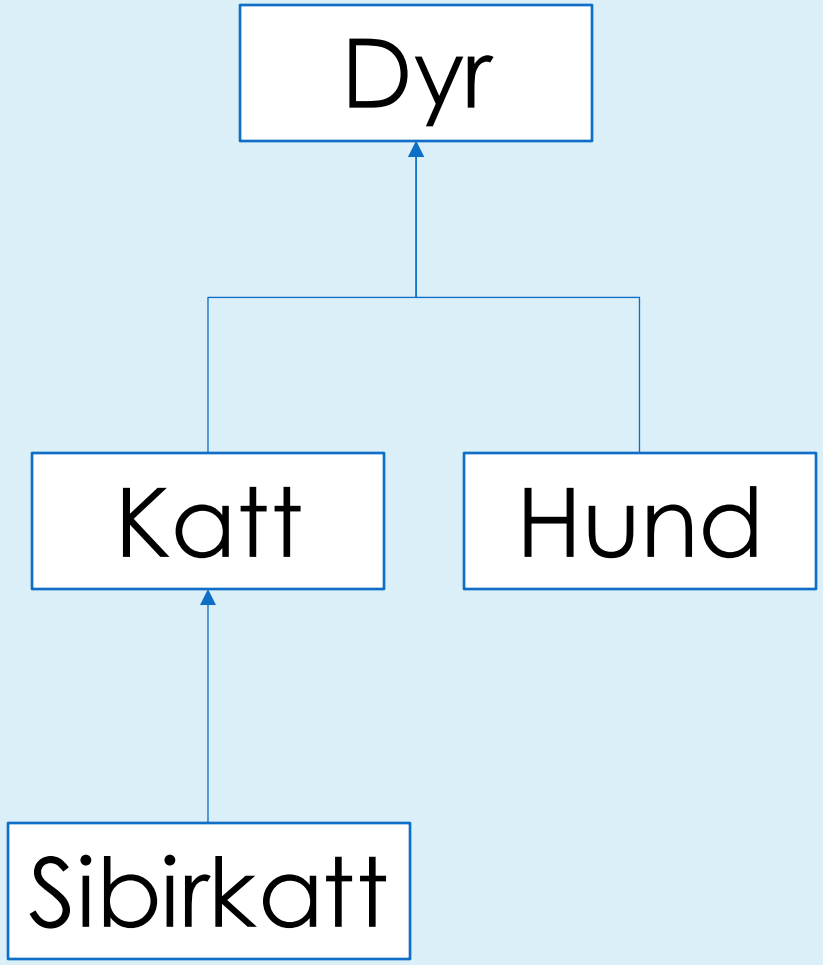


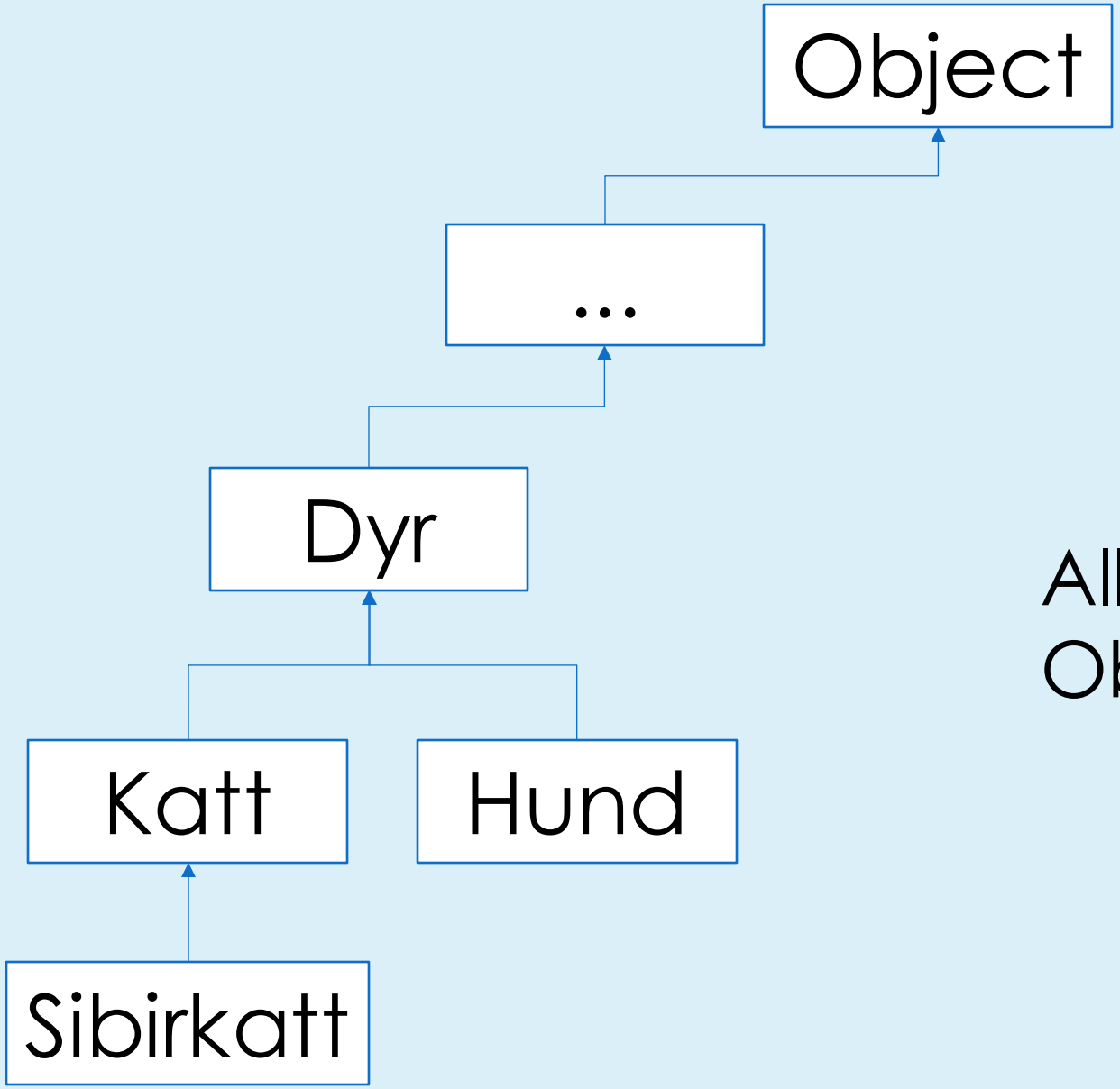
```
public class Dyr {…
```

```
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()

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

```java
public class Dyr {

    String navn;

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

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



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.

```
public class Dyr {

    String navn;

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

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

```
public class Katt extends Dyr{

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

}
```

super() i konstruktøren

- I konstruktøren til en subklasse
- Initialere instansvariabler 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{

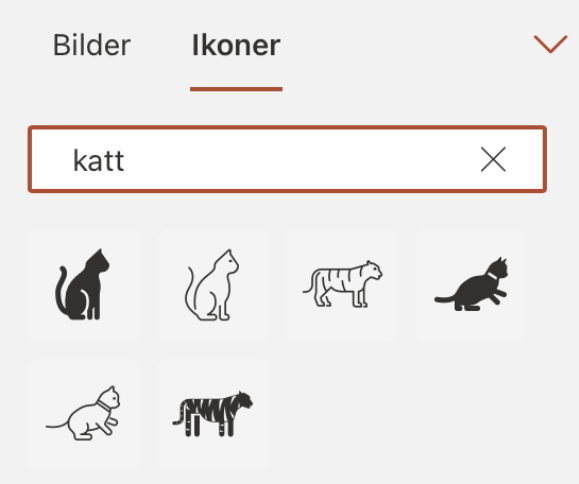
    //instansvariabler for studenter, men ikke alle personer:
    String linje;
    int trinn;

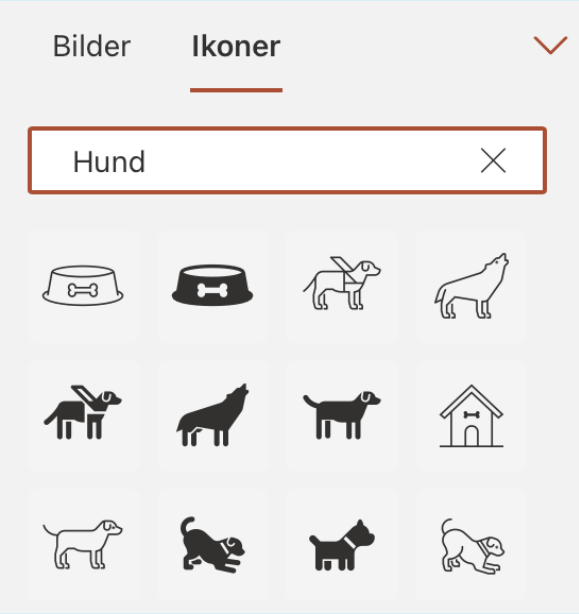
    public Student(String navn, int alder, String linje, int trinn) {
        super(navn, alder);
        this.linje = linje;
        this.trinn = trinn;
    }
}
```

Abstrakte Klasser

- 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.

- `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;
    }

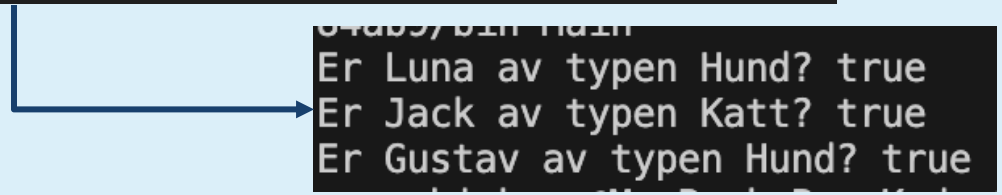
    void spis(){
        System.out.println(x:"Jeg spiser.");
    }
}
```

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));
```



```java
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
```

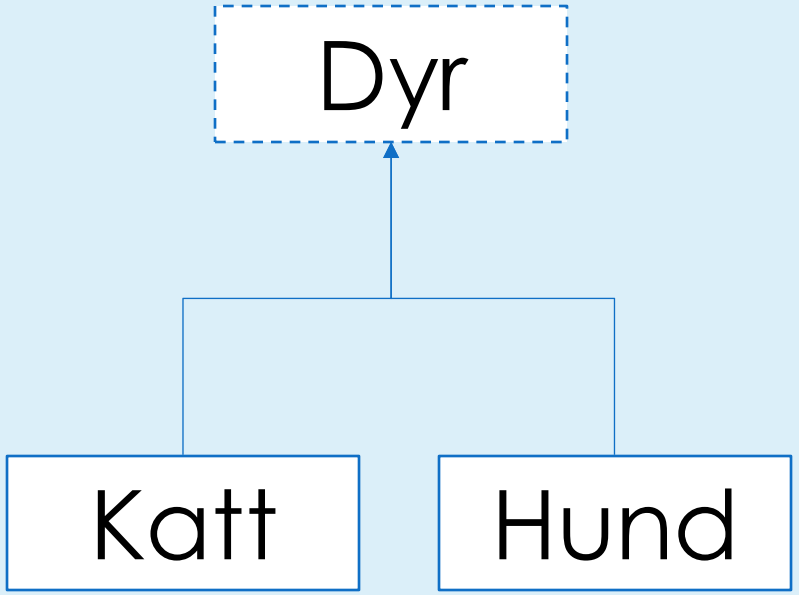
Alle subklassene er også av typen til sin(e) superklasser.

Referansetyper

- Referanser har lov til å peke nedover, ikke oppover.
- Altså: klasser med samme egenskaper.

```
Dyr dyr = new Katt()
```

→ Katte-egenskapene er skjult.

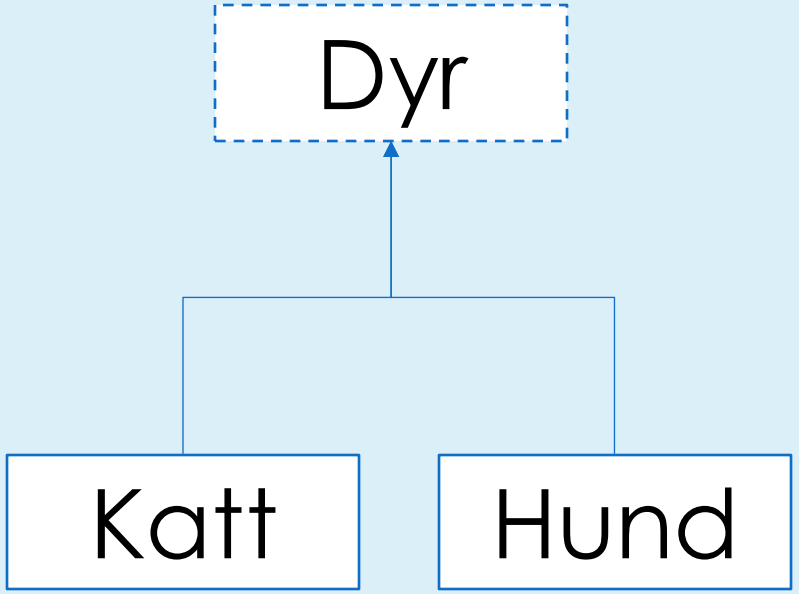


Casting

- Kan caste for å få tak i skjulte egenskaper.

```
Dyr dyr = new Katt()
Katt katt = (Katt) dyr;
```

- Bruk instanceof for å unngå feilmelding.



- Seminaroppgaver: Emneside → grupper → gruppe3 → uke3
- Trix

Oppgaver!

Lag et program som administrerer en samling (ArrayList) av kjøretøy. Programmet skal ha følgende klasser:

1. Kjøretøy-klasse:
    - Egenskaper: id (String), merke (String), modell (String), år (int)
    - Konstruktør: for å initialisere egenskapene
    - Hent-metoder og sett-metoder for egenskapene (kun hent-metode for id)
2. Bil-klasse (arver fra Kjøretøy):
    - Ekstra egenskap: seter (int)
    - Konstruktør: for å initialisere egenskapene (inkludert de arvede egenskapene fra Kjøretøy)
    - Hent-metode og sett-metode for egenskapen seter
3. Motorsykkel-klasse (arver fra Kjøretøy):
    - Ekstra egenskap: motortype (String)
    - Konstruktør: for å initialisere egenskapene (inkludert de arvede egenskapene fra Kjøretøy)
    - Hent-metoder og sett-metoder for egenskapene
4. Program-klasse:
    - Main-metode: Opprett en samling (ArrayList) for å lagre kjøretøy. Legg inn to instanser (objekter) av Bil og to instanser (objekter) av Motorsykkel. Skriv ut detaljene for kjøretøyene i samlingen.