

# OOP in Java

## Erzeugen von Klassen durch Vererbung

**Beispiel:** Mensch/ Schüler/ Manager



Mensch
Größe, Gewicht...
essen, trinken...



**Vererbung** = Weitergeben von Eigenschaften und Methoden einer Oberklasse (Superklasse/ Generalisierung) an eine neue, abgeleitete Unterklasse (Subklasse/ Spezialisierung).

Die Subklasse kann dabei das geerbte Verhalten auch redefinieren und/ oder um zusätzliche Attribute und

Schüler
Größe, Gewicht, Schule, Klasse...
essen, trinken, lernen, Arbeiten schreiben...

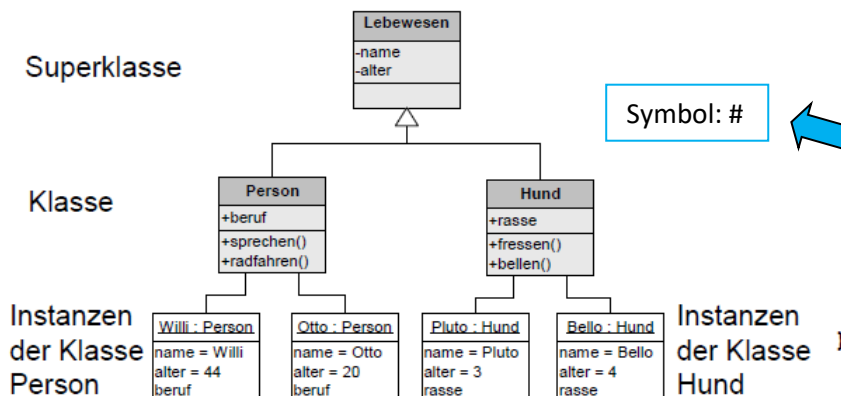
Manager
Größe, Gewicht, Firma, Einkommen...
essen, trinken, verhandeln...

### Vorteile:

- Wiederverwendung von Teilen von Klassen
- Einbinden von bereits programmiertem Code

Syntax in Java am Beispiel:

## Superklasse - Klasse - Instanz



Symbol: #



```
public class Lebewesen {
    protected String name;
    protected int alter;
}
```

```
Lebewesen (int age, String name){...}
public void schlafen() {...}
public void essen() {...}
```

Instanzen  
der Klasse  
Person

Instanzen  
der Klasse  
Hund

Die Spezialisierung erfolgt durch Implementierung der zusätzlichen Eigenschaften und Methoden sowie durch Überschreiben der Methoden.

```
public class Person extends Lebewesen {
    protected String beruf; //neue Variable
    Person () {...}

    public void essen() {...} //überschreiben

    public void radfahren() {...} //neue Methode
}
```

Implementierung der Klasse Hund.

```
public class Hund extends Lebewesen {
    protected String rasse; //neue Variable
    Hund () {...}

    public void essen() {...} //überschrieben
    public void bellen() {...} //neue Methode
}
```

## Zugriffskontrolle

- Standardeinstellung: `package`
- Normale Einschränkungen:
  - `public` - das Interface der Klasse
  - `private` - versteckt, nur von der Klasse verwendbar; *bei Vererbung für die Subklasse auch nicht sichtbar!*
- Zusätzliche Einschränkungen bei Vererbung:
  - `protected` - wie `package`, für *alle* Subklassen (auch aus anderen Paketen) aber sichtbar und überladbar!



## Aufruf von Konstruktoren

Soll ein Konstruktor der Superklasse mit Parametern aufgerufen werden, muss er explizit (mit `super(...)`) aufgerufen werden.

**ACHTUNG!** Der Aufruf des Konstruktors der Superklasse ist die erste Anweisung im Konstruktor der Subklasse.

```
class Hund extends Lebewesen{
    Hund(String name) {
        super(name);
        ...
    }
}
```

## Schlüsselwort `super`

Mit `super` können generell Methoden von Superklassen aufgerufen werden. Beispiel:

```
class Hund {
    void bellen(String laut) {
        System.out.println(laut);
    }
}

class Schaeferhund extends Hund {
    void bellen(String laut) {
        String newLaut = laut + laut;
        super.bellen(newLaut);
    }
}
```

- Die Kennzeichnung einer Klasse als abstrakt **verhindert**, dass Instanzen dieser Klasse erzeugt werden können.
- Eine Klasse mit **abstrakten** Methoden muss als **abstrakte** Klasse definiert werden.
- Alle abstrakten Methoden müssen in den nicht abstrakten Subklassen **implementiert** werden.
- Nicht alle Methoden einer abstrakten Klasse müssen abstrakt definiert sein.
- Java-Syntax der Klassendeklaration:

```
abstract class KlassenName {
    void aNormalMethod(int a) {...}
    abstract void aAbstractMethod(int b);
}
```

Block der Methode fehlt

## Abstrakte Klassen

- Soll es ein **einheitliches Interface** für alle abgeleiteten Subklassen definiert werden\*.
- Es wird sichergestellt (Komplier), dass die Methoden und Parameterlisten des Interfaces in der Subklasse implementiert werden.
- In mindestens einer Methode wird nur die Schnittstelle definiert (abstrakte Methode), ihre Implementierung erfolgt in den Subklassen.

## Abstrakte Superklasse `Lebewesen`

Die Klasse `Lebewesen` wird als **abstract** deklariert, die Methode `essen` muss in allen direkten Subklassen implementiert werden.

```
public abstract class Lebewesen {
    protected String name;
    protected int alter;

    Lebewesen (int age, String name){...};
    public void schlafen() {...};
    public abstract void essen();
    // muss in der Sub-Klasse implementiert werden
}
```