Elektronikschule Tettnang	Softwareentwicklung	Datum:	Klasse:
	Vererbung	Name:	

Vererbung

Die Vererbung ist eine Art von Beziehung zwischen Klassen. Von einer Vererbung spricht man immer wenn zwei Klassen im Zusammenhang „ist ein“ stehen. In UML wird die Vererbung auch Generalisierung genannt. Die generalisierte Klasse nennt man dann Elternklasse. Die spezialisierte Klasse nennt man Kindklasse.

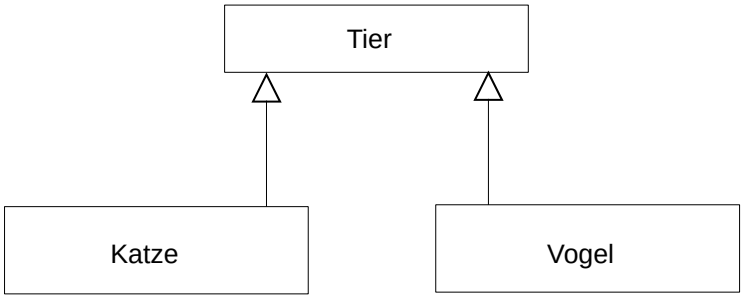
Beispiel:
Eine Katze **ist ein** Tier. Ein Vogel **ist ein** Tier. Ein Motor **ist ein** Einzelteil (vom Auto).
Die Klassen Tier und Einzelteil sind hier Elternklassen. Die Klassen Katze, Vogel und Motor sind hier Kindklassen.

Gegenbeispiel:
~~Ein Mitarbeiter ist eine Firma. Ein Fahrzeughalter ist ein Auto. Ein Raum ist ein Haus.~~

Beschreibung in UML

Die Vererbung in UML wird mit einem Pfeil mit einer fest definierten Spitze dargestellt. Die Klasse an der Pfeilspitze nennt man Elternklasse. Die Klasse am Pfeilende nennt man Kindklasse. Bei der Vererbung gibt es keine Multiplizitäten. (Vgl.: ~~Eine Katze ist mehrere Tiere.~~ Das macht keinen Sinn.)

Beispiel:



Vorteil der Vererbung:
Es gibt Eigenschaften/Methoden die alle Tiere gemeinsam haben und Eigenschaften/Methoden die individuell für Katzen und Vögel sind. Die gemeinsamen Eigenschaften/Methoden müssen nur einmal in der Elternklasse implementiert werden. Die individuellen Eigenschaften/Methoden werden in der jeweiligen Kindklasse implementiert.

Umsetzung in Java

Die Vererbung zwischen der Elternklasse und der Kindklasse wird mit dem Schlüsselwort `extends` hergestellt. Zur besseren Übersicht sind die Klassen hier erstmal leer.

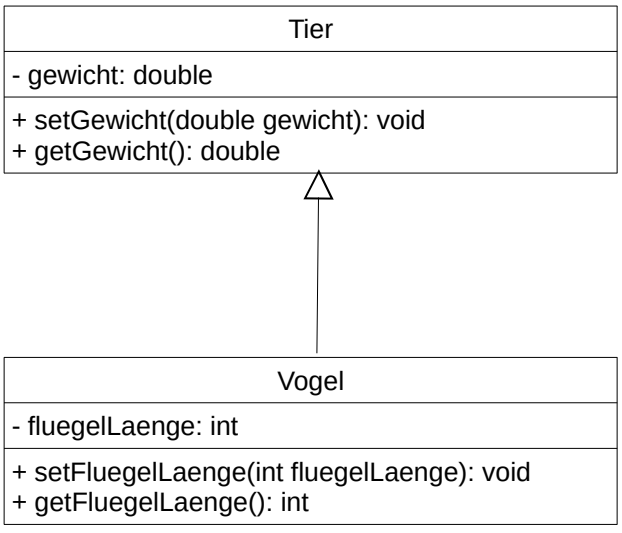
```
public class Tier {

}

public class Katze extends Tier {

}

public class Vogel extends Tier {

}
```

Vererbung von Eigenschaften / Methoden

Die gemeinsamen Eigenschaften/Methoden müssen nur einmal in der Elternklasse implementiert werden. Die individuellen Eigenschaften/Methoden werden in der jeweiligen Kindklasse implementiert. In der Kindklasse sind nun durch die Vererbung alle Eigenschaften/Methoden der Vaterklasse verfügbar, außer denen, die den Zugriff „private“ besitzen.

```
public class Tier {
    private double gewicht;

    public void setGewicht(double gewicht) {
        this.gewicht = gewicht;
    }
    public double getGewicht() {
        return this.gewicht;
    }
}

public class Vogel extends Tier {
    private int fluegelLaenge;

    public void setFluegelLaenge(int fluegelLaenge) {
        this.fluegelLaenge = fluegelLaenge;
    }
    public int getFluegelLaenge() {
        return this.fluegelLaenge;
    }
}

public class Test {
    public static void main(String[] args) {
        Vogel v = new Vogel();
        v.setGewicht(2.4);
        v.setFluegelLaenge(30);
        System.out.println("Gewicht: " + v.getGewicht());
        System.out.println("Flügellänge: " + v.getFluegelLaenge());
    }
}
```

In UML:



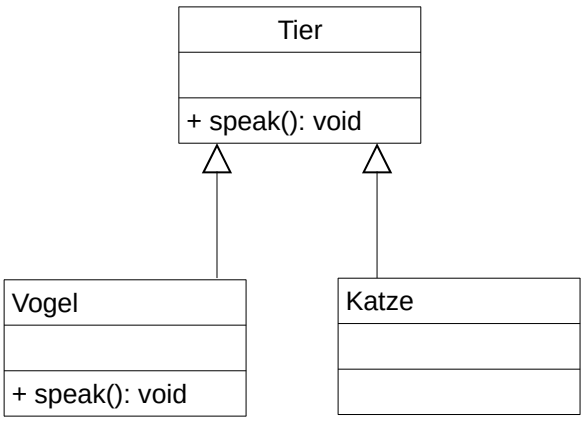
Methoden der Elternklasse überschreiben

In der Elternklasse implementierte Methoden können in der Kindklasse überschrieben werden. Dadurch hat man in der Elternklasse noch ein Backup, falls es für eine bestimmte Kindklasse (noch) keine Spezialisierung dieser Methode gibt.

Im folgenden Beispiel wird die Methode `speak()` in der Elternklasse Tier von der Kindklasse Vogel überschrieben:

```
public class Tier {
    public void speak() {
        System.out.println("schnauf schnauf");
    }
}

public class Vogel extends Tier {
    public void speak() {
        System.out.println("zwitscher zwitscher");
    }
}

public class Katze extends Tier {
    // noch keine speak() Methode implementiert. Es wird auf die
    // speak() Methode von Tier zurückgegriffen.
}

public class Test {
    public static void main(String[] args) {
        Vogel v = new Vogel();
        Katze k = new Katze();

        v.speak();     // Ausgabe „zwitscher zwitscher“
        k.speak();     // Ausgabe „schnauf schnauf“
    }
}
```

In UML:



Merke:

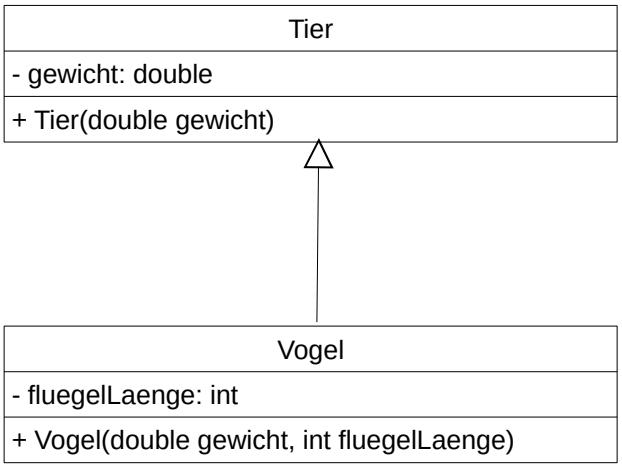
Jede Klasse hat die Klasse java.lang.Object als Elternklasse.

Konstruktoraufruf mit `super`

Falls die Elternklasse keinen Standardkonstruktor anbietet, muss explizit im Konstruktor der Kindklasse ein Konstruktor der Elternklasse über das Schlüsselwort `super` aufgerufen werden. Der Aufruf von `super` muss ganz zu Beginn im Konstruktor erfolgen.

```
public class Tier {
    private double gewicht;

    public Tier(double gewicht) {
        this.gewicht = gewicht;
    }
}

public class Vogel extends Tier {
    private int fluegelLaenge;

    public Vogel(double gewicht, int fluegelLaenge) {
        super(gewicht);
        this.fluegelLaenge = fluegelLaenge;
    }
}

public class Test {
    public static void main(String[] args) {
        Vogel v = new Vogel(2.4, 30);
    }
}
```

In UML:



Merke:

`super` ist eine Referenz auf den Namensraum der Elternklasse. Damit lassen sich überschriebene Objektmethoden und Konstruktoren der Elternklasse aufrufen.

`this` ist eine Referenz auf Objektmethoden und Konstruktoren, die sich auf die aktuelle Klasse bezieht.