SEW
IT-Medientechnik

NVS
Informatik

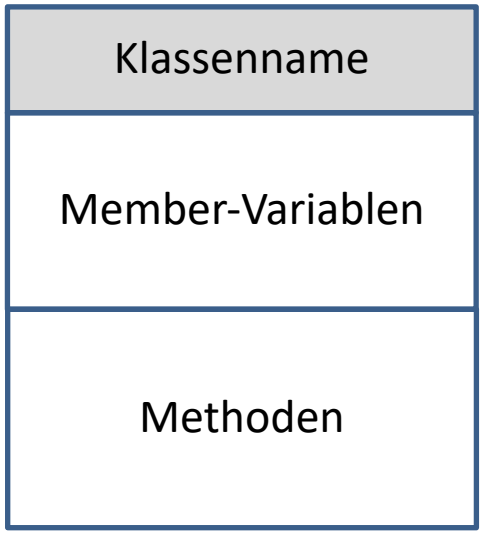
Beziehungen zwischen Klassen

UML Klassendiagramme

UML-Klassendiagramm

UML Unified Modeling Language

Eine Klasse wird als Rechteck mit drei Sektionen dargestellt:



UML-Klassendiagramm

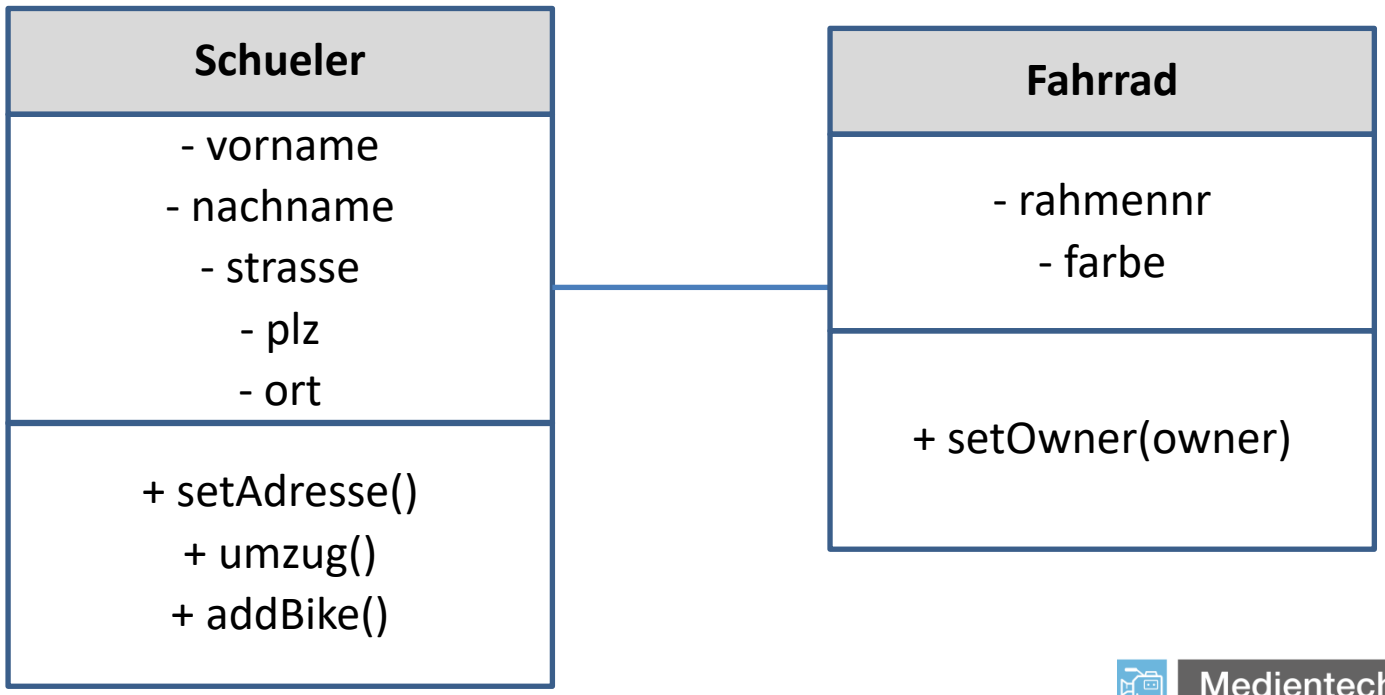
Der Detaillierungsgrad der Variablen- bzw. Methodenbeschreibung ist frei wählbar.

Fraction
numerator denominator
shorten() fromString() toString()

Fraction
- numerator: int - denominator: int
+ shorten() : Fraction + fromString(str: String) + toString() : String

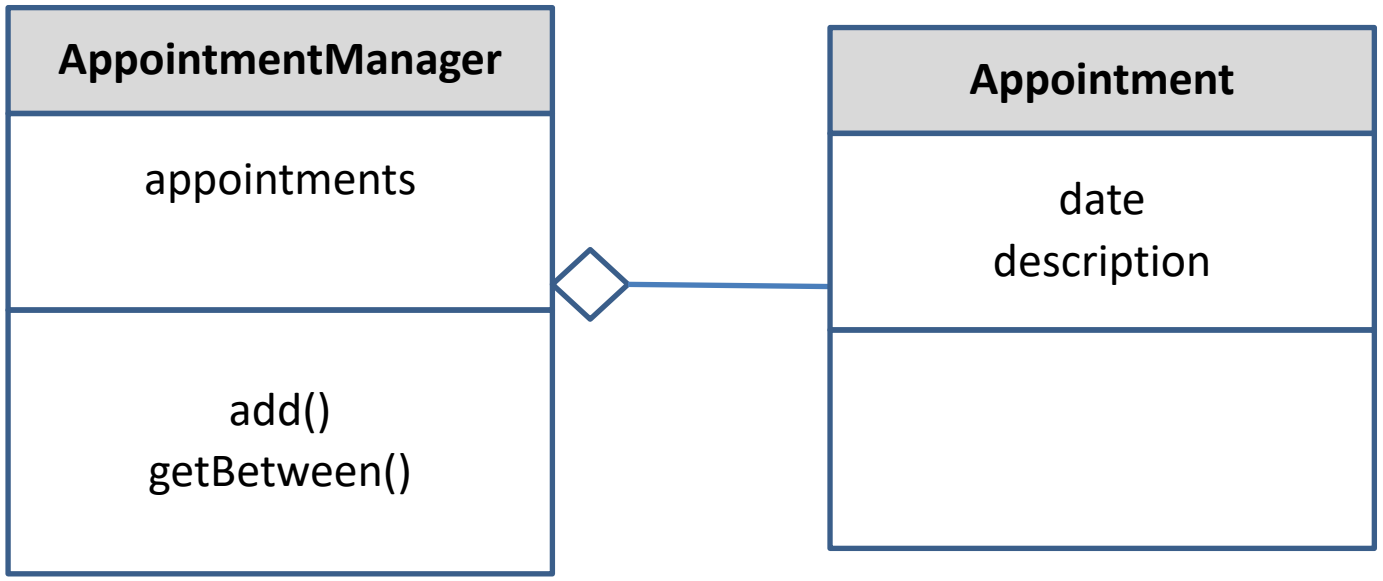
Assoziation

Enthält eine Klasse Member-Variablen einer anderen Klasse, so spricht man von einer **Assoziation**. Assoziationen sind benannte Beziehungen zwischen Klassen, wie zum Beispiel „Schüler geht in Schule“ oder „Lehrer unterrichtet Schüler“, ...



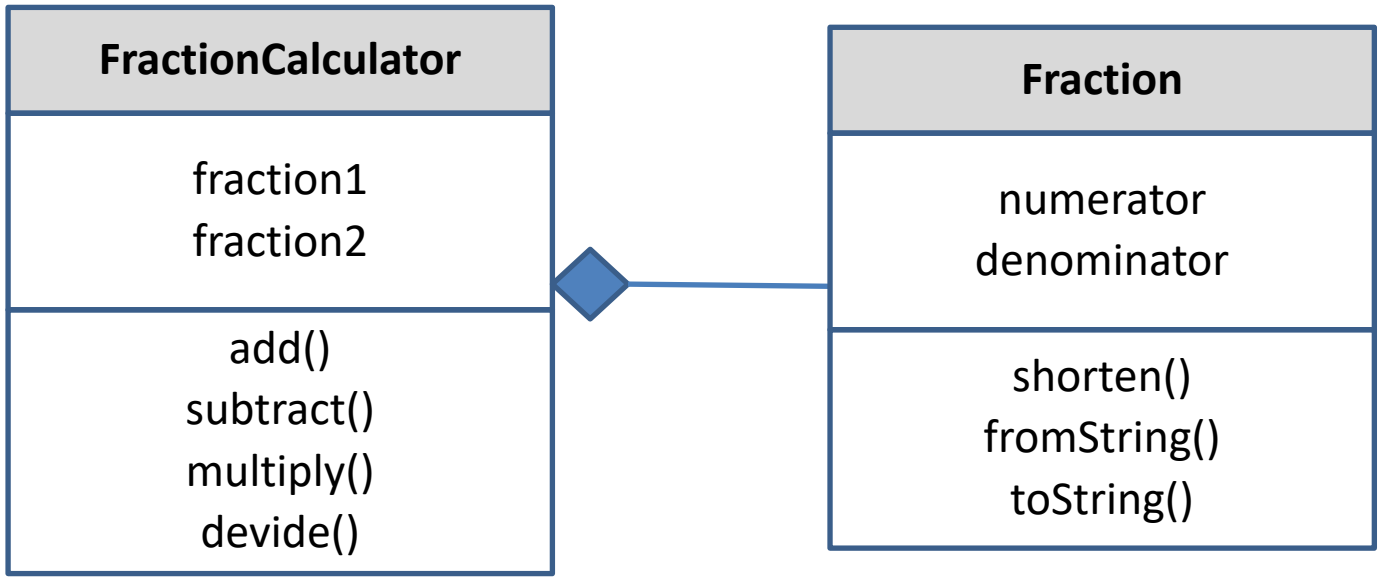
Aggregation

Aggregationen sind spezielle Assoziationen – nämlich solche, mit der Beziehung „a consists of b“ (besteht aus).

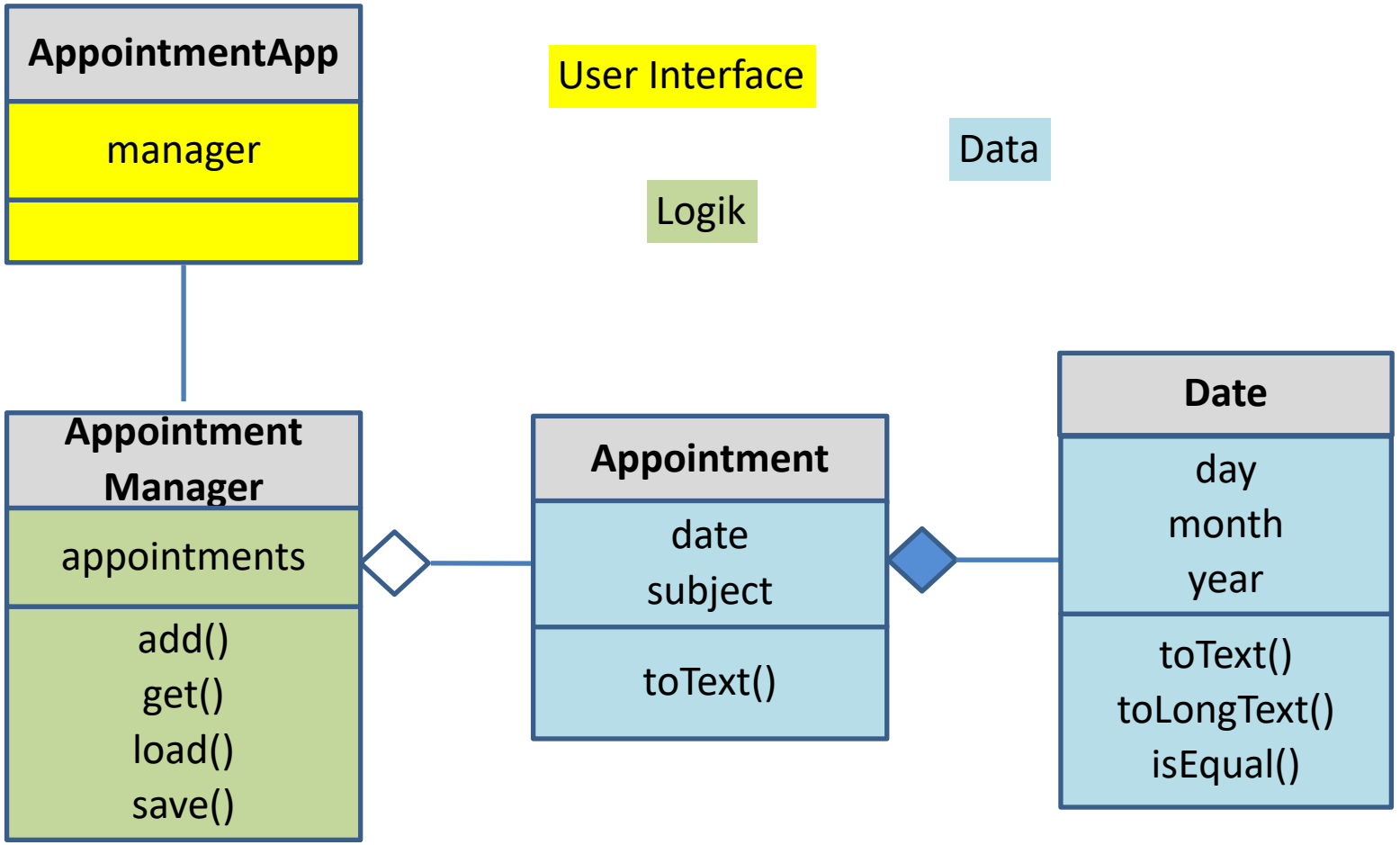


Komposition

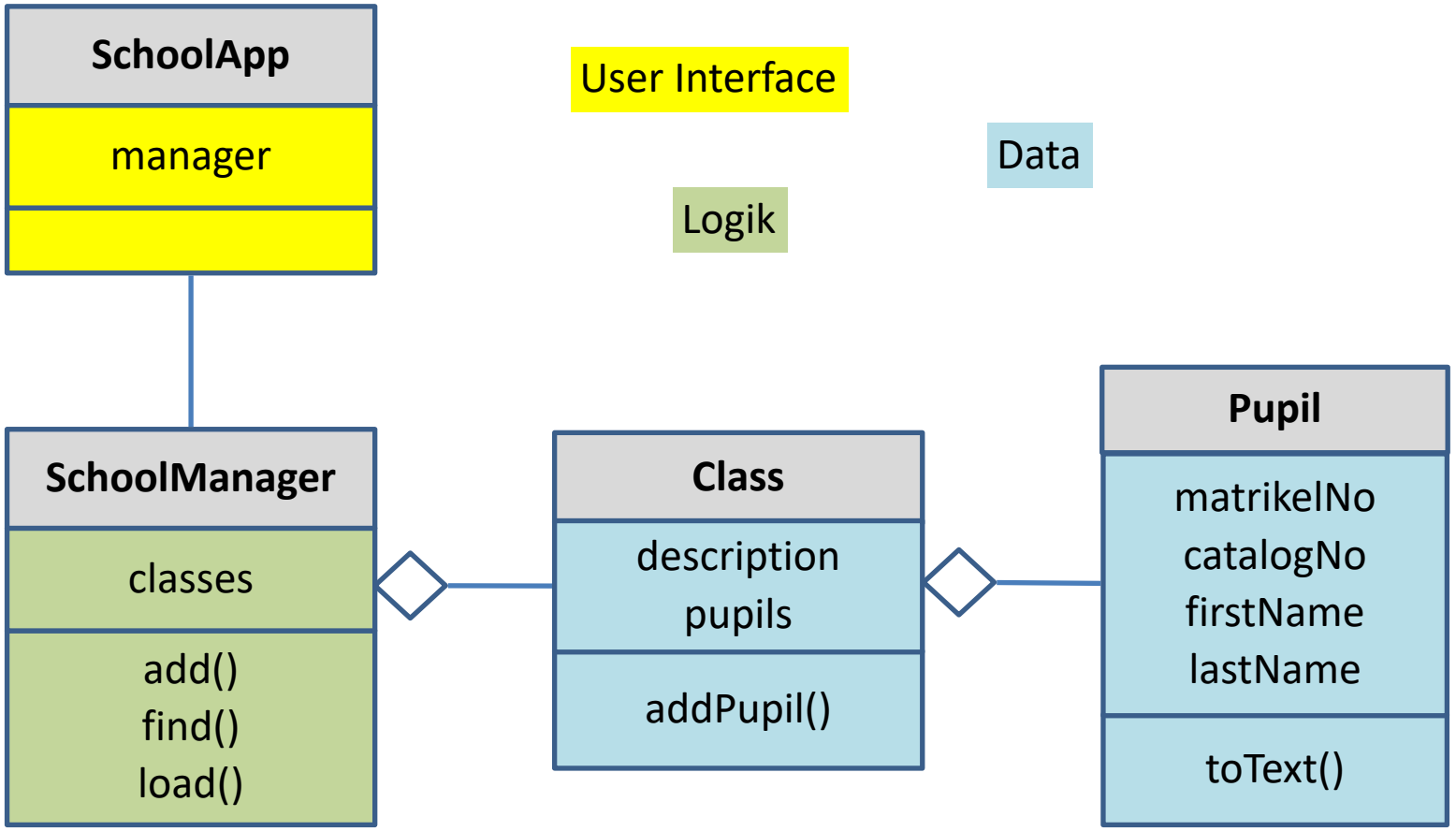
Sind die aggregierten Objekte von der Anzahl festgelegt und zur Funktionalität unbedingt erforderlich, so spricht man von einer **Komposition**.



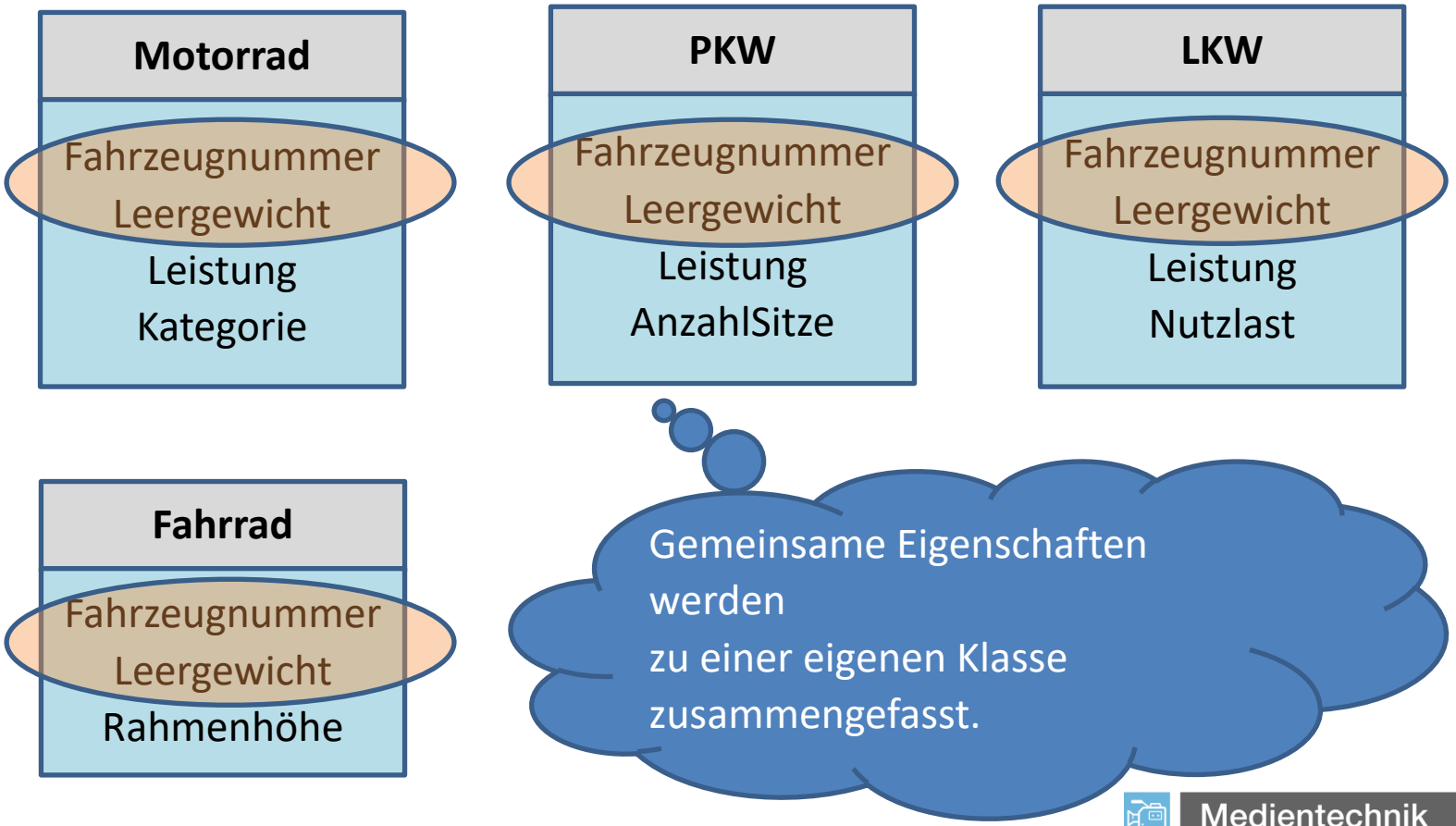
Assoziation, Aggregation und Komposition



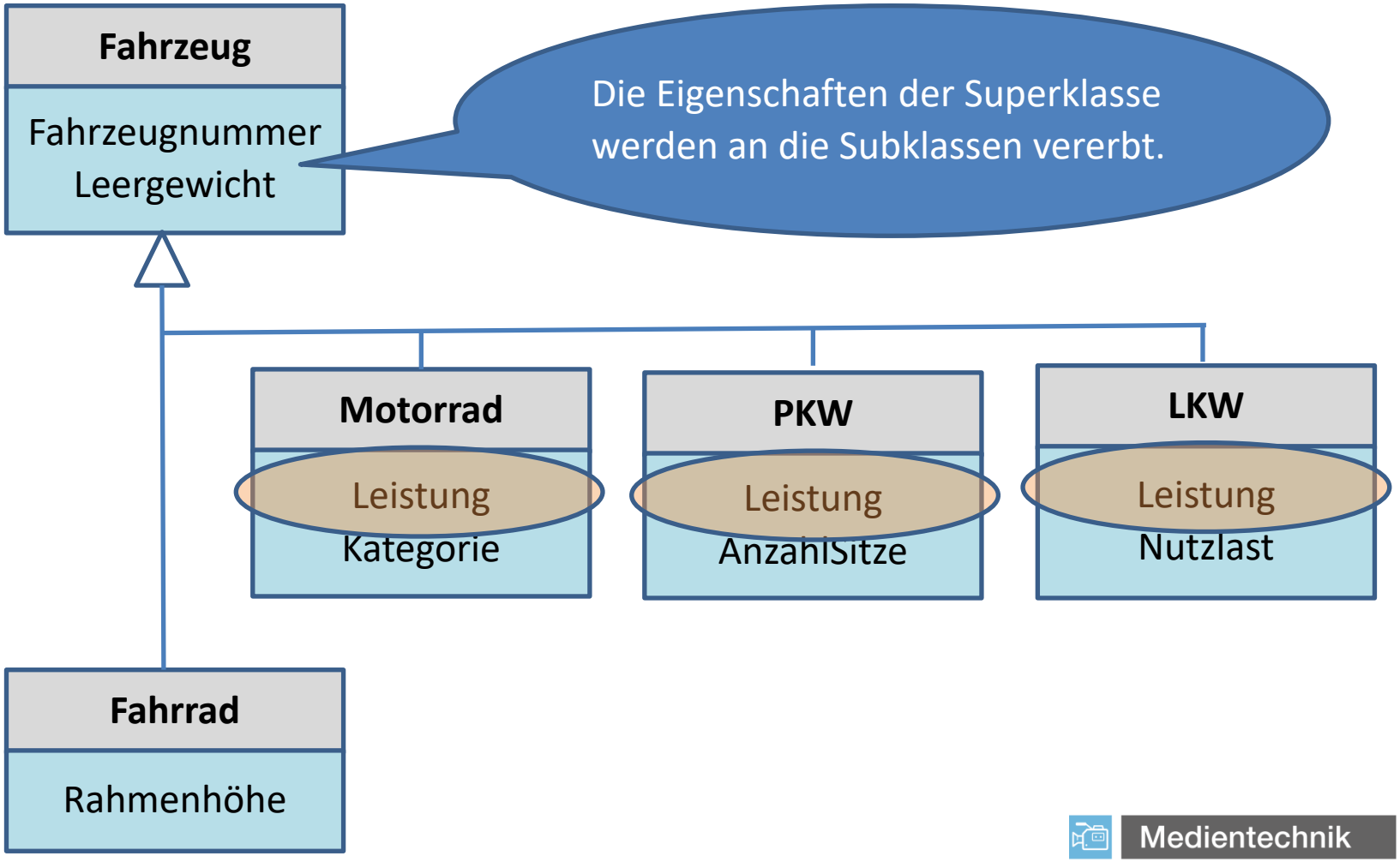
Assoziation, Aggregation und Komposition



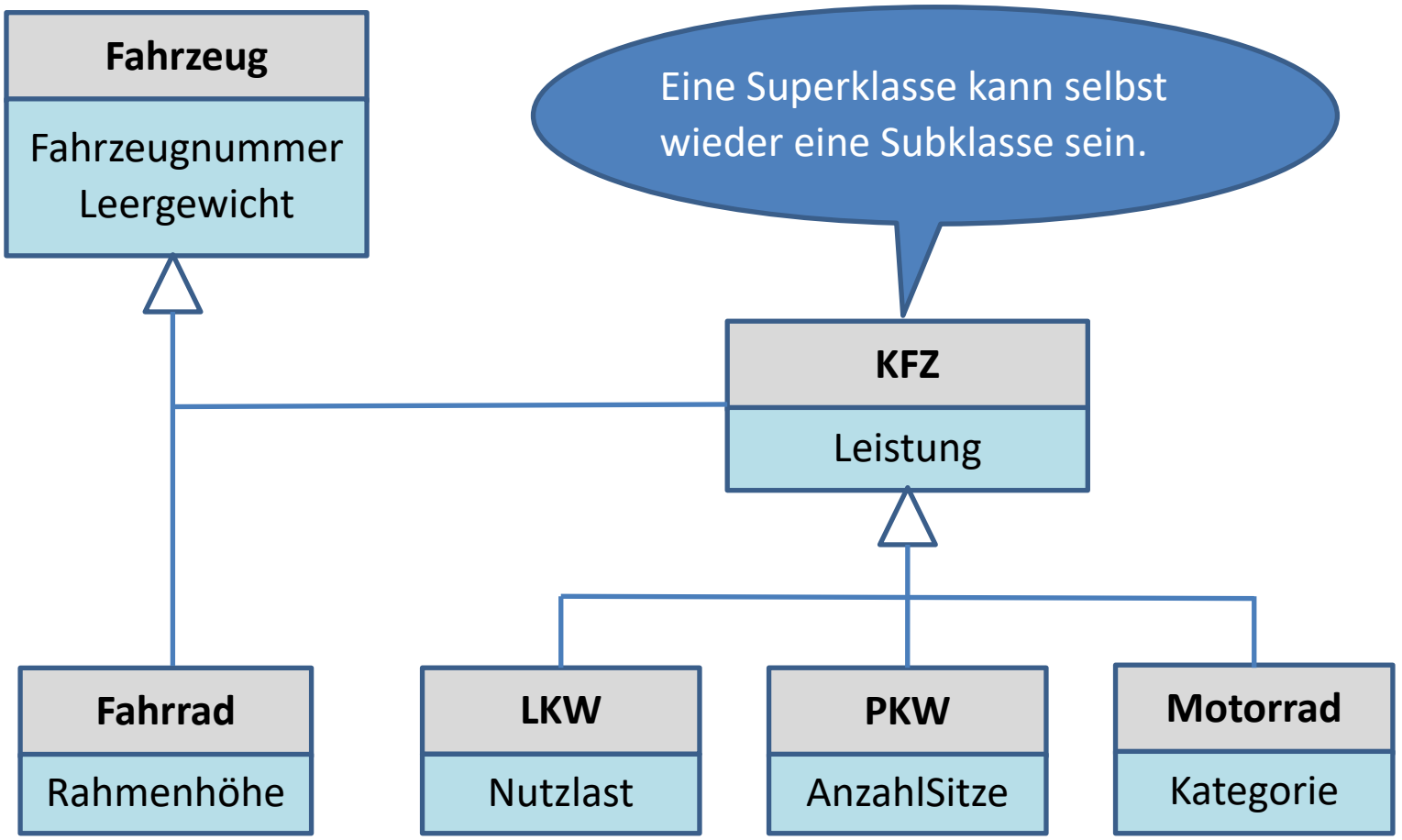
Vererbung


Motorrad
Fahrzeugnummer
Leergewicht
Leistung
Kategorie
PKW
Fahrzeugnummer
Leergewicht
Leistung
AnzahlSitze
LKW
Fahrzeugnummer
Leergewicht
Leistung
Nutzlast
Fahrrad
Fahrzeugnummer
Leergewicht
Rahmenhöhe
Gemeinsame Eigenschaften
werden
zu einer eigenen Klasse
zusammengefasst.

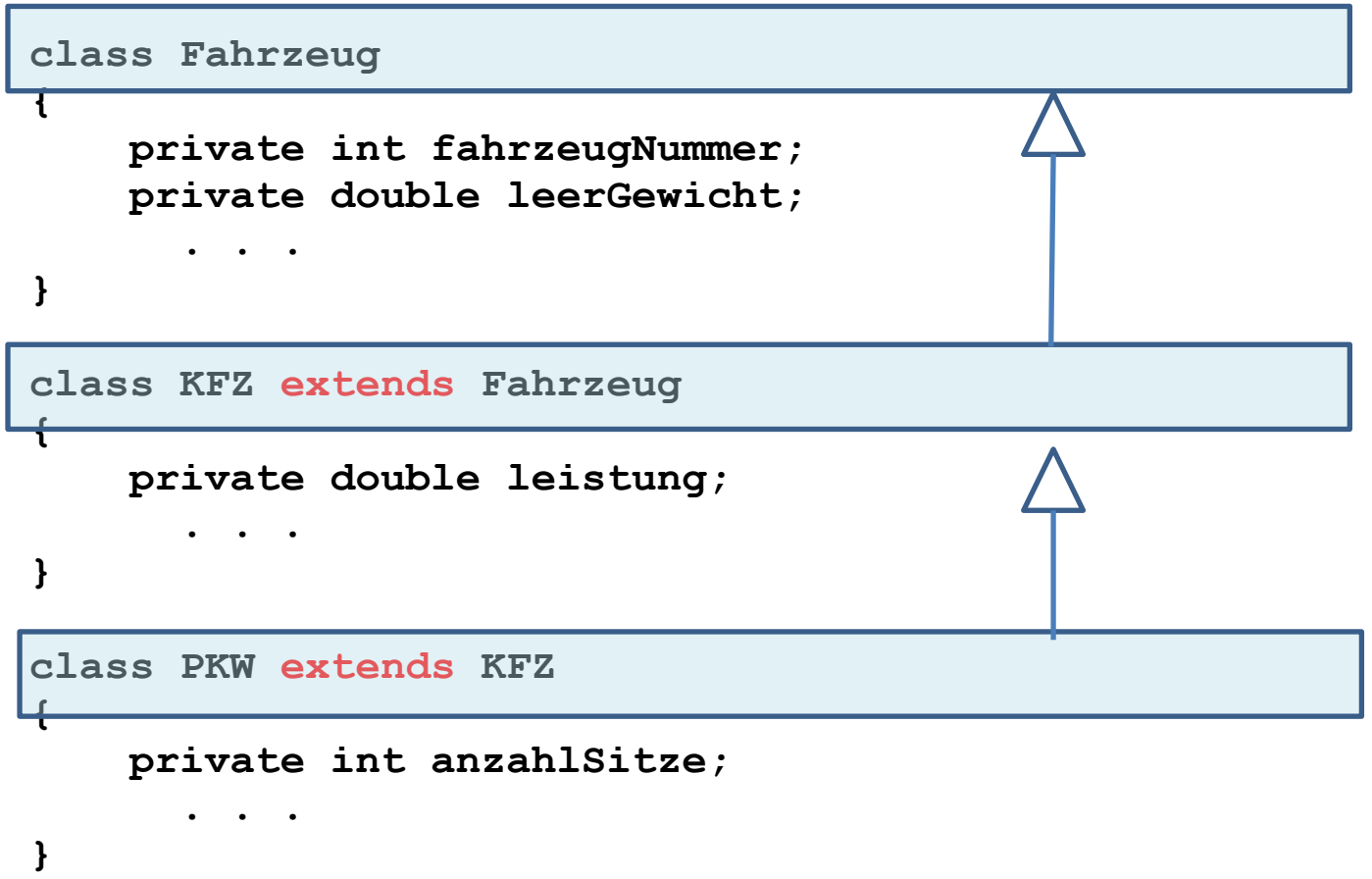
Vererbung


Fahrzeug
Fahrzeugnummer
Leergewicht
Die Eigenschaften der Superklasse werden an die Subklassen vererbt.
Motorrad
Leistung
Kategorie
PKW
Leistung
AnzahlSitze
LKW
Leistung
Nutzlast
Fahrrad
Rahmenhöhe

Vererbung


Fahrzeug
Fahrzeugnummer
Leergewicht
Eine Superklasse kann selbst wieder eine Subklasse sein.
KFZ
Leistung
Fahrrad
Rahmenhöhe
LKW
Nutzlast
PKW
AnzahlSitze
Motorrad
Kategorie

Vererbung


class Fahrzeug
{
private int fahrzeugNummer;
private double leerGewicht;
. . .
}
class KFZ extends Fahrzeug
{
private double leistung;
. . .
}
class PKW extends KFZ
{
private int anzahlSitze;
. . .
}

Vererbung

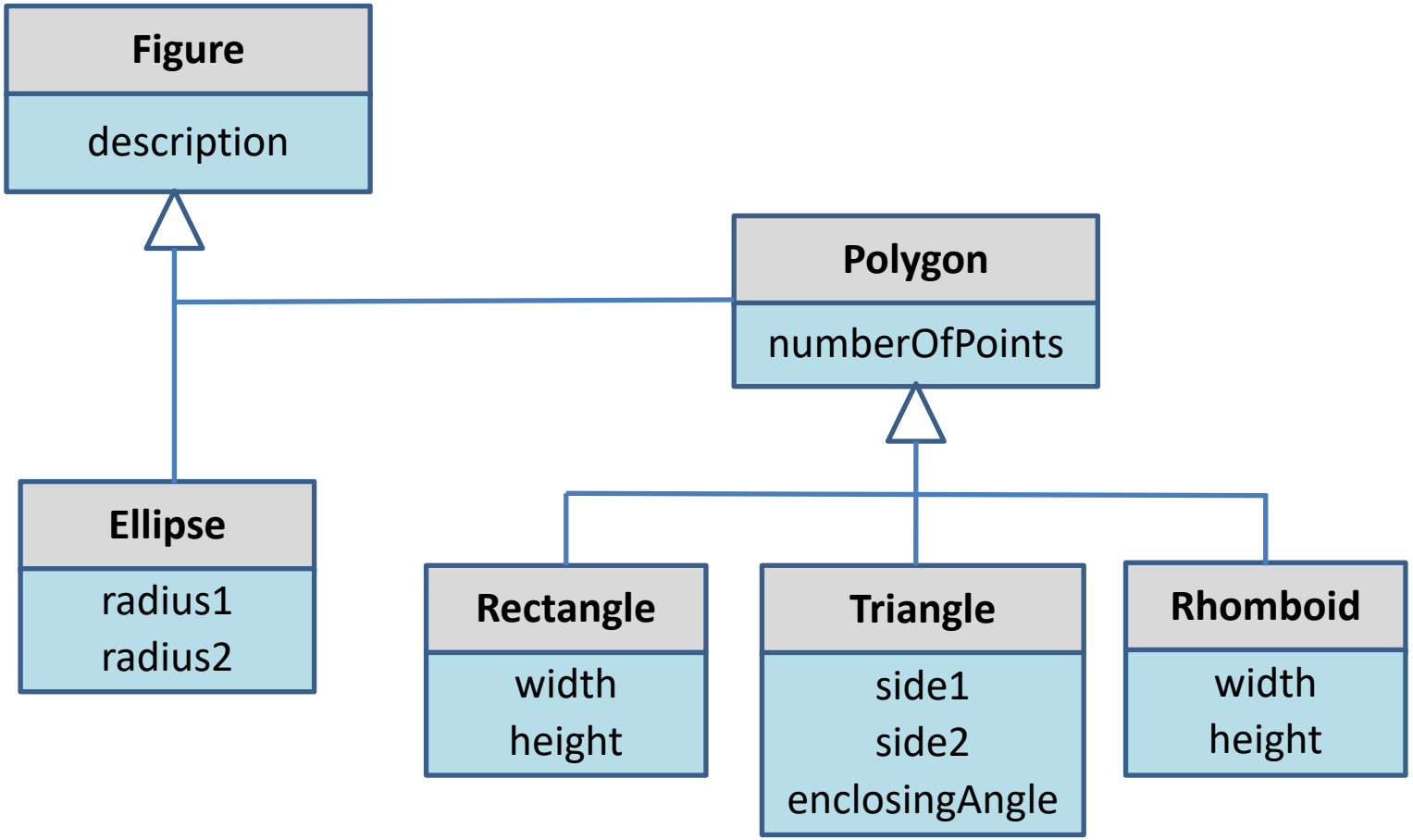
```
Fahrzeug [] fahrzeuge = new Fahrzeug[10];

fahrzeuge[0] = new Fahrrad();
fahrzeuge[1] = new PKW();
fahrzeuge[2] = new LKW();

. . .


for ( int i = 0; i < fahrzeuge.length; i++) {
    if ( fahrzeuge[i] instanceof Fahrrad ) {
      System.out.println("Fahrrad");
      Fahrrad f = (Fahrrad)fahrzeuge[i];
      . . .
}
```

Vererbung - Konstruktorverkettung

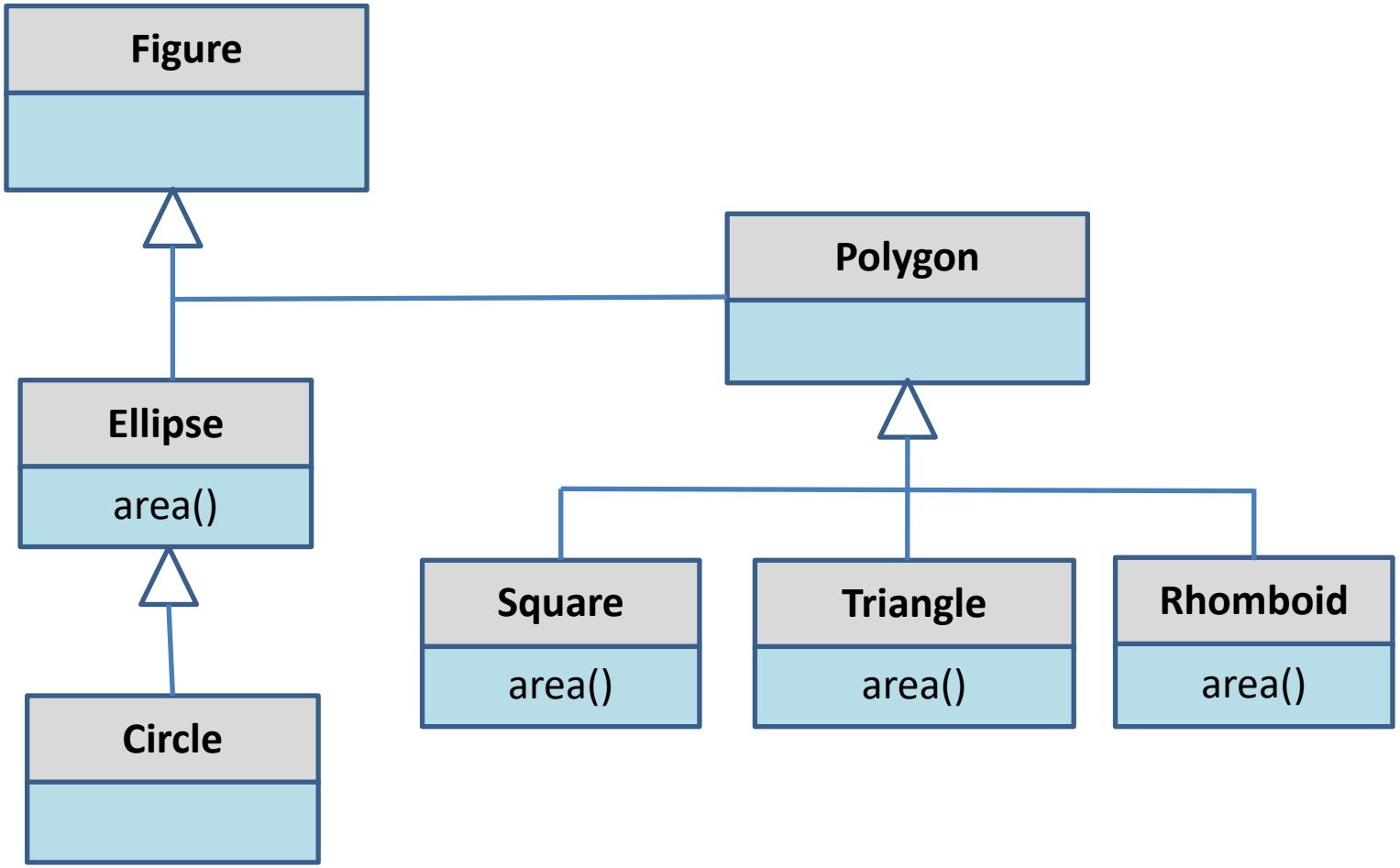
```
class Fahrzeug
{
    private double leerGewicht;
    public Fahrzeug(double leergewicht) {
        this.leerGewicht = leergewicht;
    }
        . . .
}

class KFZ extends Fahrzeug
{
    private double leistung;

    public KFZ(double leergewicht, double leistung) {
        super(leergewicht);
        this.leistung = leistung;
    }
       . . .
}
```

muss die erste Anweisung im Konstruktor sein

Vererbung - Polymorphie



Vererbung - Polymorphie

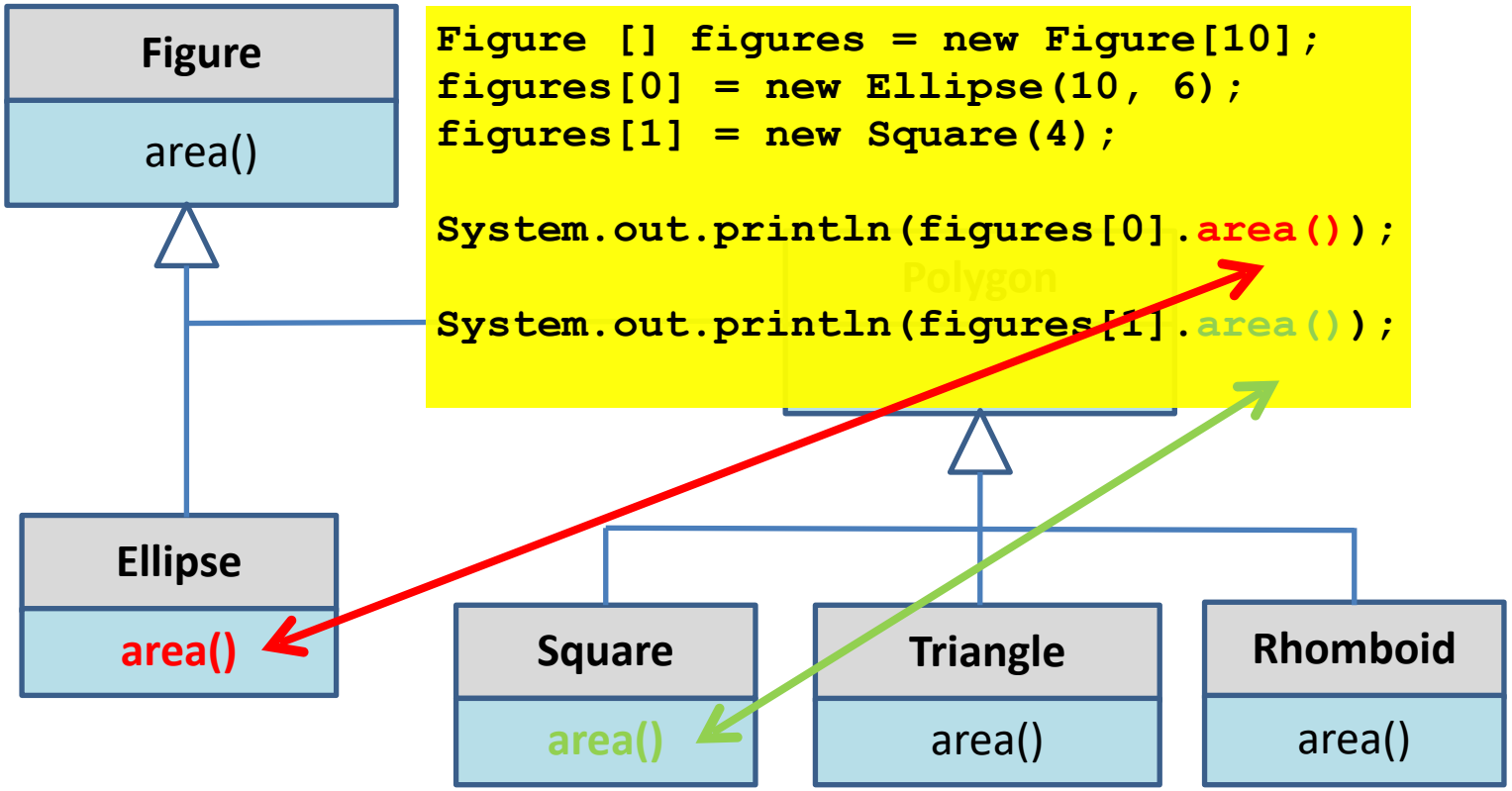


Vererbung

```
Figure[] figures = new Figure[10];
figures[0] = new Ellipse(10, 6);
figures[1] = new Square(4);
. . .
```

```
for ( int i = 0; i < figures.length; i++) {
    if ( figures[i] instanceof Square) {
      Square s = (Square)figures[i];
      System.out.println(s.area());
    } else if ( figures[i] instanceof Ellipse) {
      . . .
}
```

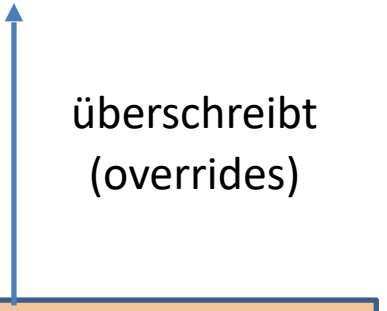
Vererbung - Polymorphie



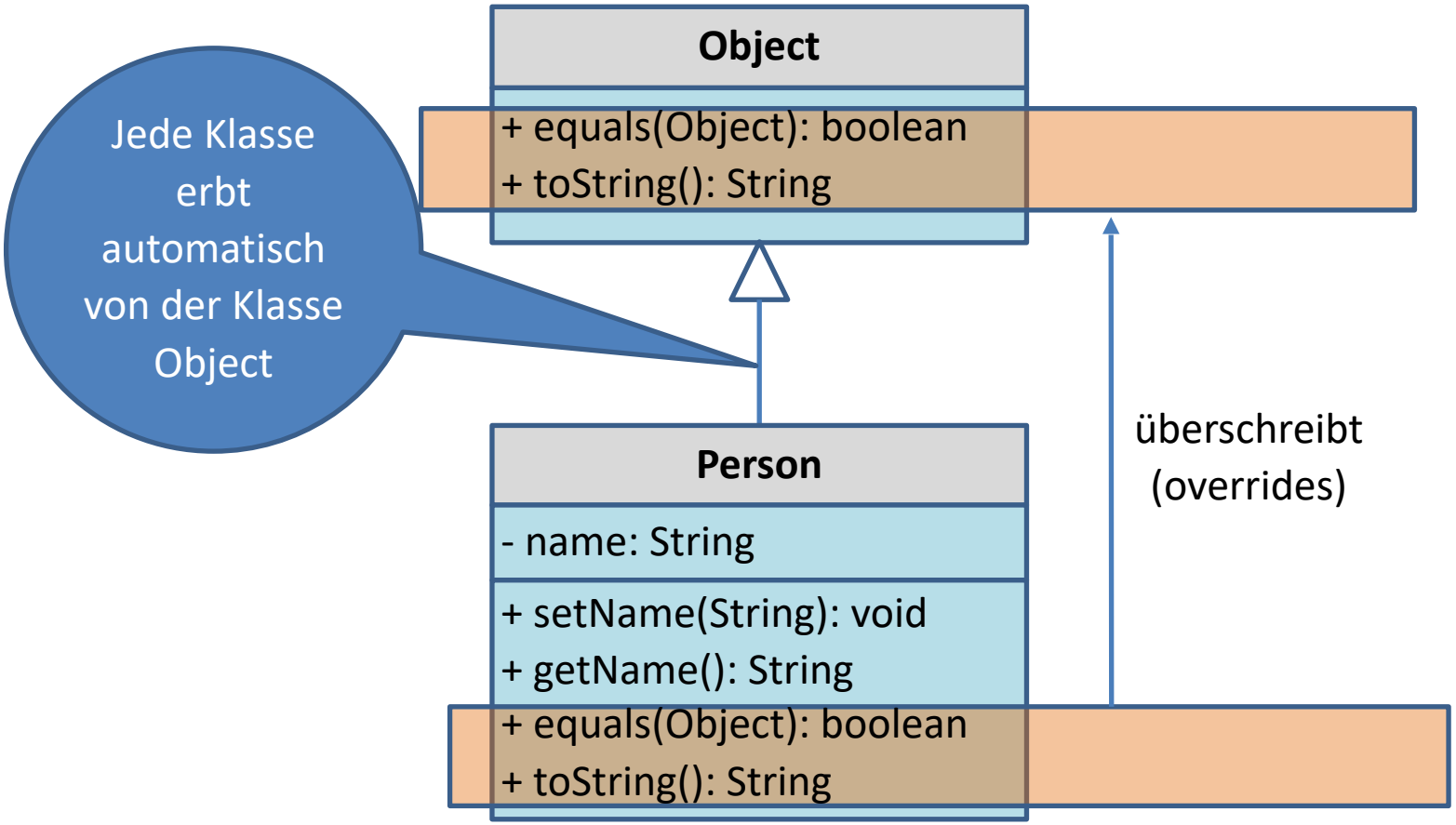
Vererbung - Polymorphie

```
abstract class Figure {
        . . .
    abstract public double area();
        . . .
}
```

```
class Ellipse extends Figure {
        . . .
    @Override
    public double area() {
        return Math.Pi*a*b;
    }
        . . .
}
```



Vererbung – Die Klasse Object



Vererbung – Die Klasse Object

Überschreibt von **`Object`**

```
class Person {
    private String name;
    @Override
    public boolean equals(Object obj) {
        if ( obj instanceof Person )
            return this.name != null &&
                   this.name.equals(((Person)obj).name);
        return false;
    }
}
```

Vererbung – Die Klasse Object

Überschreibt von **Person**

```
class Pupil extends Person {
    private String class;
    @override
    public boolean equals(Object obj) {
        if ( obj instanceof Pupil)
            return super.equals(obj) &&
                   this.class != null &&
                   this.class.equals((Pupil)obj).class);
        return false;
    }
}
```

HTL LEONDING

Noch Fragen?

Schön, hier zu lernen