```
public class FangDieMaus {

    public static void main(String[] args) {

        Katze Miez  = new Katze();
        Maus  Pieps = new Maus();

        Miez.fangen(Pieps);

        if(Pieps.gefangen()==false && Miez.istMuede()==true){

            Miez.schlafen();
            Pieps.grinsen();
        }

    }
}
```

…. Der Weg zur Objektorientierung ….

Interpretieren Sie folgenden Quellcode – Beispiel 1

```
public static void main(String[] args) {

    Auto speedy  = new Auto("BMW");
    Auto flitzer = new Auto("Mercedes");

    double streckeKm = 150;

    speedy.geschwindigkeit = 300;
    flitzer.geschwindigkeit = 200;

    IO.print(speedy.modell + ": ");
    IO.println(speedy.berechneFahrtzeit(streckeKm));

    IO.print(flitzer.modell + ": ");
    IO.println(flitzer.berechneFahrtzeit(streckeKm));
}
```

Interpretieren Sie folgenden Quellcode – Beispiel 2

```
public static void main(String[] args) {

    Lehrer   maier = new Lehrer();
    Schueler toni  = new Schueler();

    maier.erklaertStoff();

    if(toni.hoertzu()==true){

        maier.getHappy();
    }
    else{
        maier.getFrustie();
    }
}
```

Interpretieren Sie folgenden Quellcode – Beispiel 3

```
public static void main(String[] args) {

    Disko galaxy = new Disko();
    galaxy.qmTanzflaeche = 50;

    int besucher = 800;

    if( besucher/galaxy.qmTanzflaeche > 3){

        IO.println("Cooler Schuppen");
    }
    else{

        IO.println("Tote Hose ....");
    }
}
```

Versuch der Zuordnung:

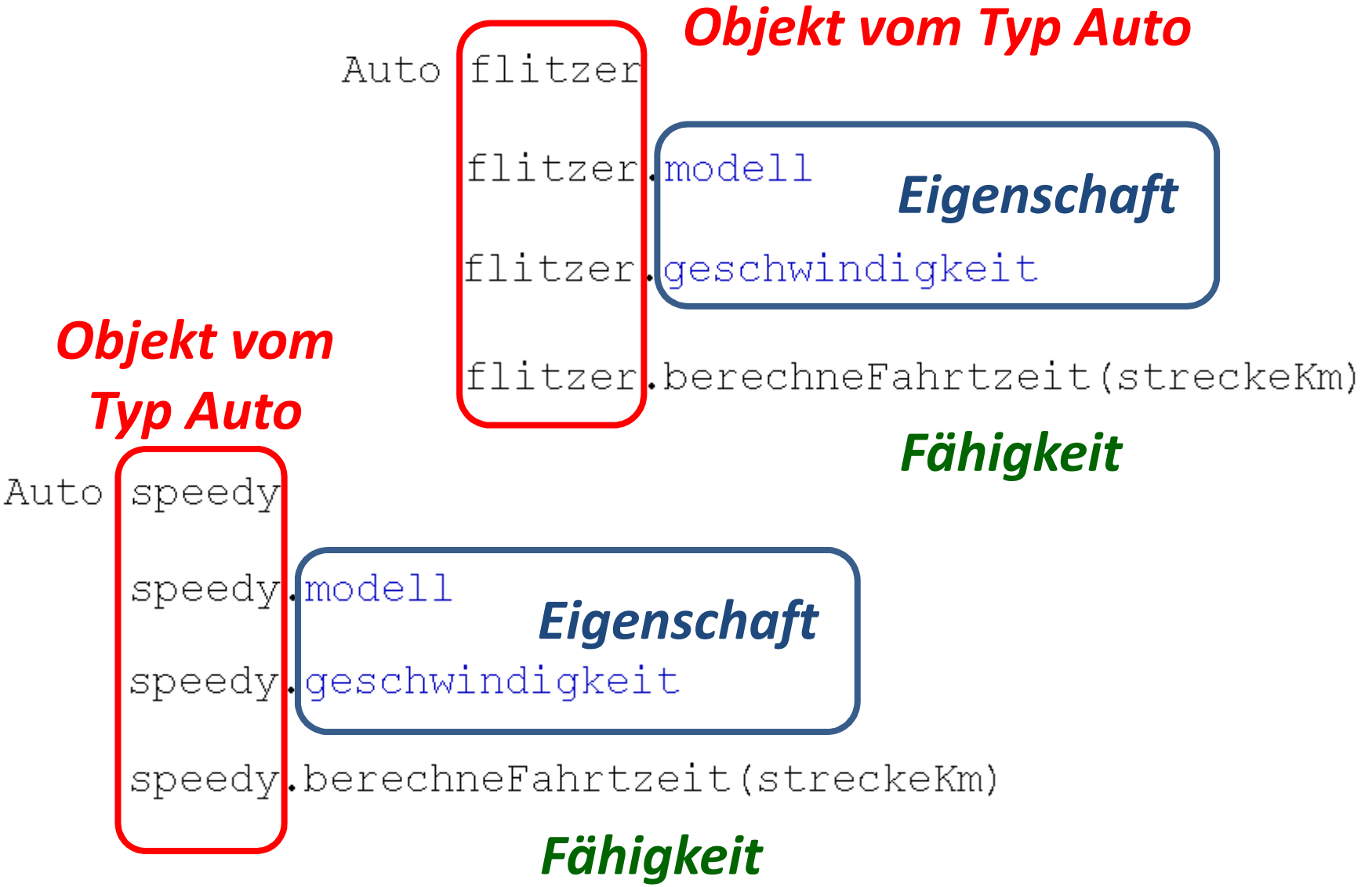
Wo finden Sie jeweils in den gezeigten Beispielen

Objekte

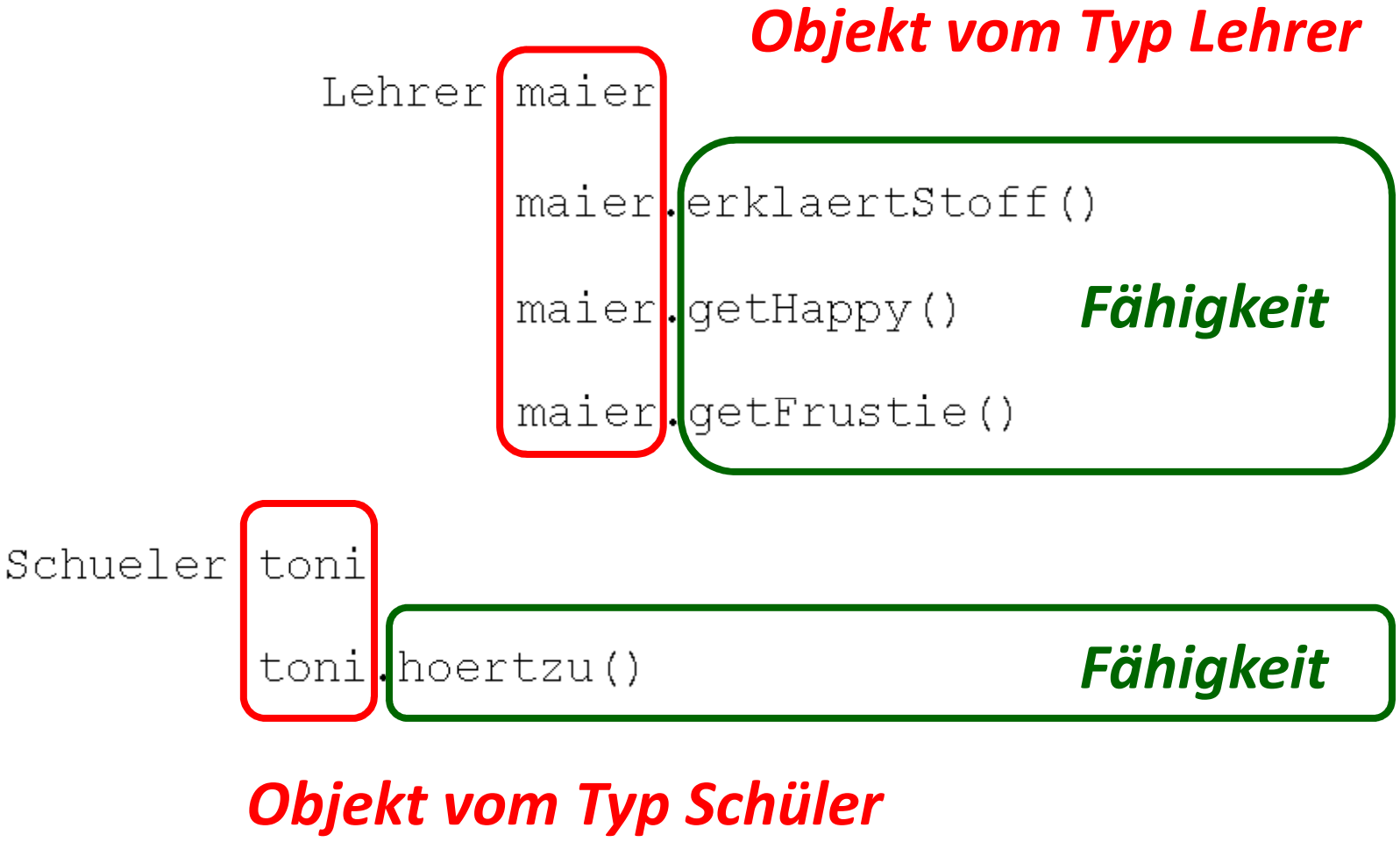
Fähigkeiten von Objekten

Eigenschaften von Objekten

Versuch der Zuordnung: Beispiel 01



Versuch der Zuordnung: Beispiel 02

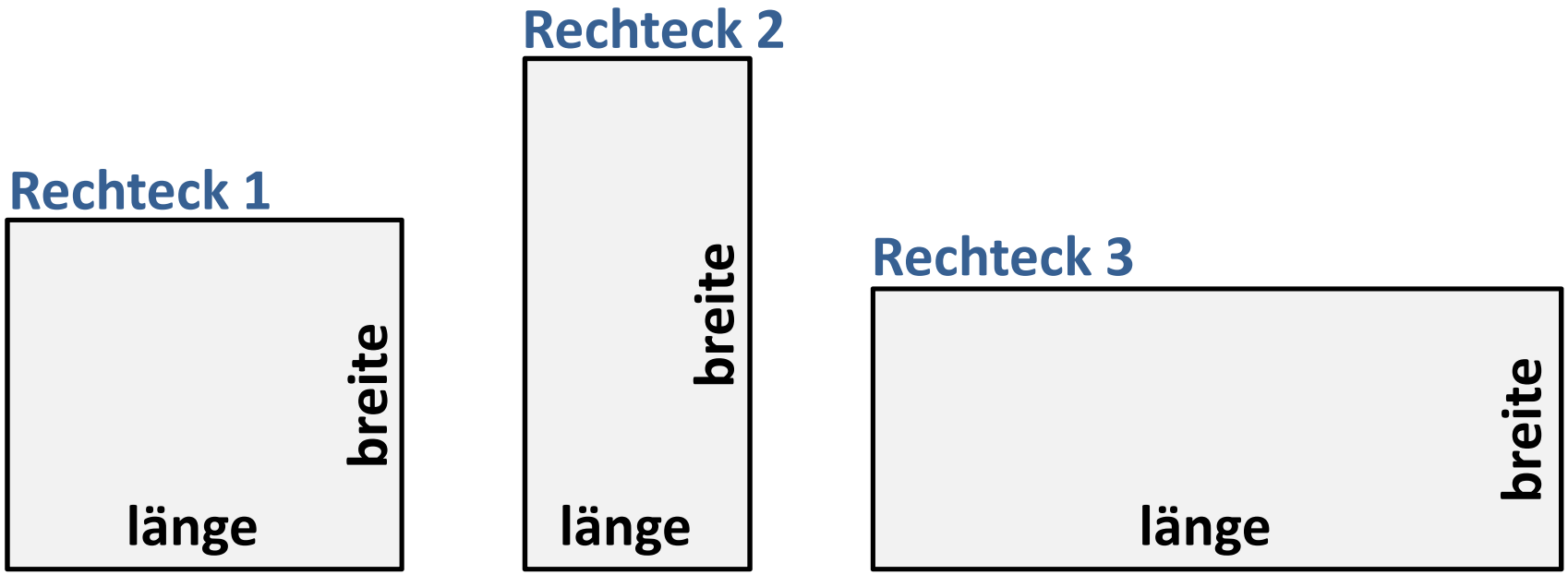


Versuch der Zuordnung: Beispiel 03

Der Weg zur Objektorientierung

Einfaches Beispiel:

Drei Rechtecke sollen nach ihrer Fläche absteigend sortiert ausgegeben werden.



Der Weg zur Objektorientierung

Zunächst die klassisch strukturierte Variante:

```
public static void main(String[] args) {

    double laenge1, laenge2, laenge3;
    double breite1, breite2, breite3;
    double maximale_flaeche = 0;
    double mittlere_flaeche = 0;
    double minimale_flaeche = 0;
    String max = "", mittel = "", min = "";
```

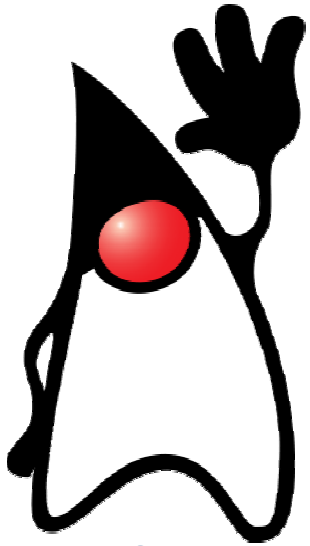
Kein definierter Zusammenhang zwischen den Variablen

```
    laenge1 = IO.readDouble("Laenge Rechteck 1: ");
    breite1 = IO.readDouble("Breite Rechteck 1: ");

    laenge2 = IO.readDouble("Laenge Rechteck 2: ");
    breite2 = IO.readDouble("Breite Rechteck 2: ");

    laenge3 = IO.readDouble("Laenge Rechteck 3: ");
    breite3 = IO.readDouble("Breite Rechteck 3: ");
```

Der Weg zur Objektorientierung

Zunächst die klassisch strukturierte Variante:

```
if((laenge1 * breite1 > laenge2 * breite2) && (laenge1 * breite1 > laenge3 * breite3)){

    max = "Rechteck 1";
    maximale_flaeche = laenge1 * breite1;

    if(laenge2 * breite2 > laenge3 * breite3){

        mittel = "Rechteck 2";
        mittlere_flaeche = laenge2 * breite2;

        min = "Rechteck 3";
        minimale_flaeche = laenge3 * breite3;
    }
    else{

        mittel = "Rechteck 3";
        mittlere_flaeche = laenge3 * breite3;

        min = "Rechteck 2";
        minimale_flaeche = laenge2 * breite2;
    }
}
```



Tippfehler?
Übersicht?
Wunde Finger?

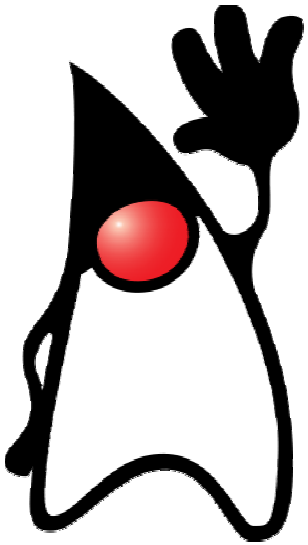
.. dieser Block muss noch zwei weitere Male durchgeführt werden …

Der Weg zur Objektorientierung

Zunächst die klassisch strukturierte Variante:

```
IO.println("Hier die sortierten Rechtecke:");
IO.println(max + " mit einer Flaeche von " + maximale_flaeche);
IO.println(mittel + " mit einer Flaeche von " + mittlere_flaeche);
IO.println(min + " mit einer Flaeche von " + minimale_flaeche);
```

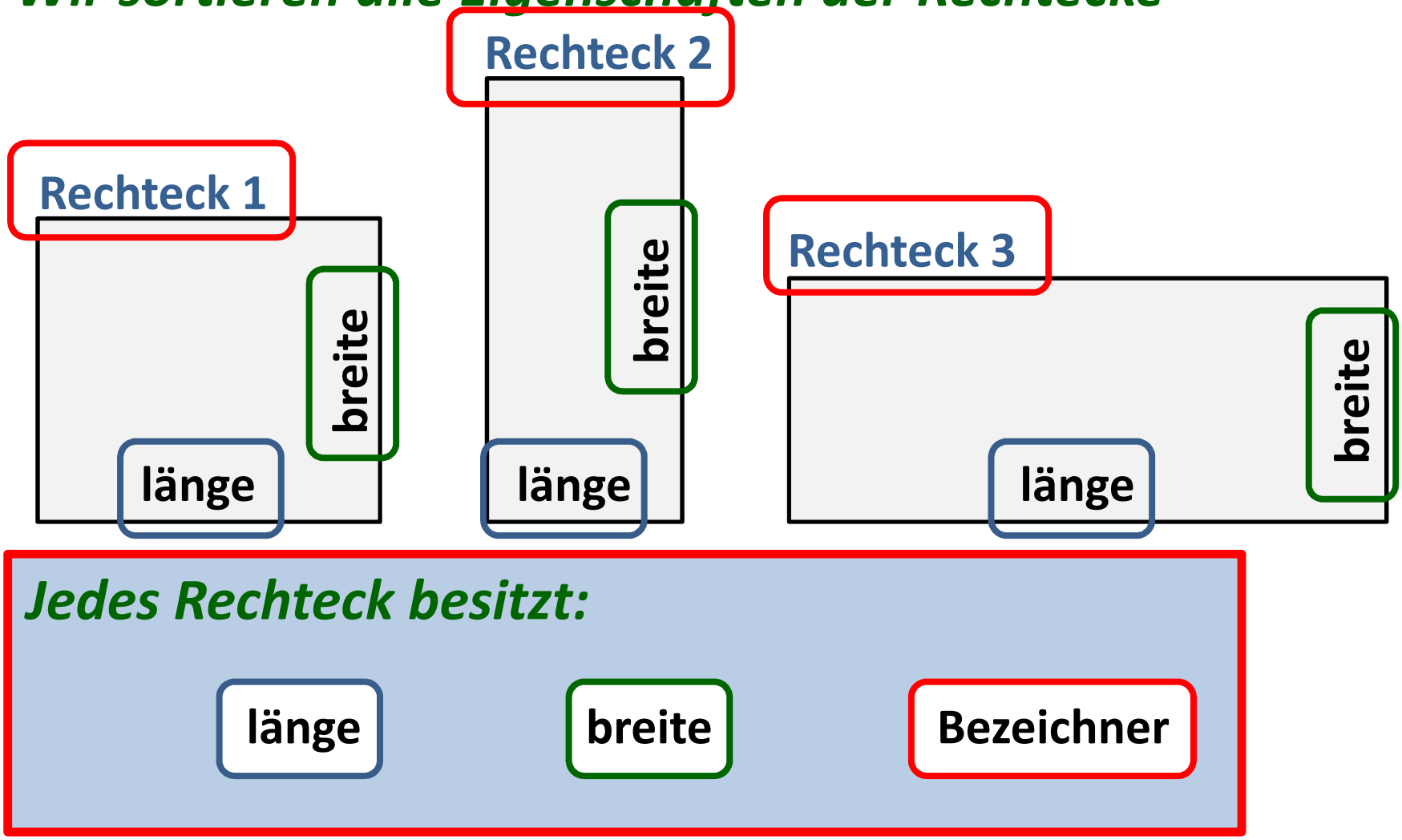
```
Laenge Rechteck 1: 3
Breite Rechteck 1: 6
Laenge Rechteck 2: 4
Breite Rechteck 2: 8
Laenge Rechteck 3: 3
Breite Rechteck 3: 8
Hier die sortierten Rechrtecke:
Rechteck 2 mit einer Flaeche von 32.0
Rechteck 3 mit einer Flaeche von 24.0
Rechteck 1 mit einer Flaeche von 18.0
```



Wie kann man dies verbessern?

Der Weg zur Objektorientierung – ein Ansatz

Wir sortieren alle Eigenschaften der Rechtecke



Der Weg zur Objektorientierung – ein Ansatz

….erstellen eine Vorlage, die alles für ein eventuelles Rechteck beinhaltet …….

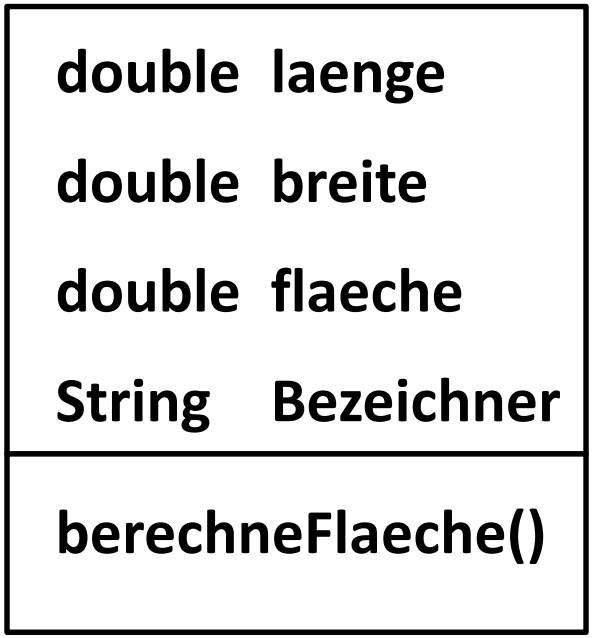
Diese Vorlage nennt man Klasse

Eigenschaften

(was wird abgespeichert?)

Fähigkeiten

(was kann ein Rechteck?)



Der Weg zur Objektorientierung – ein Ansatz

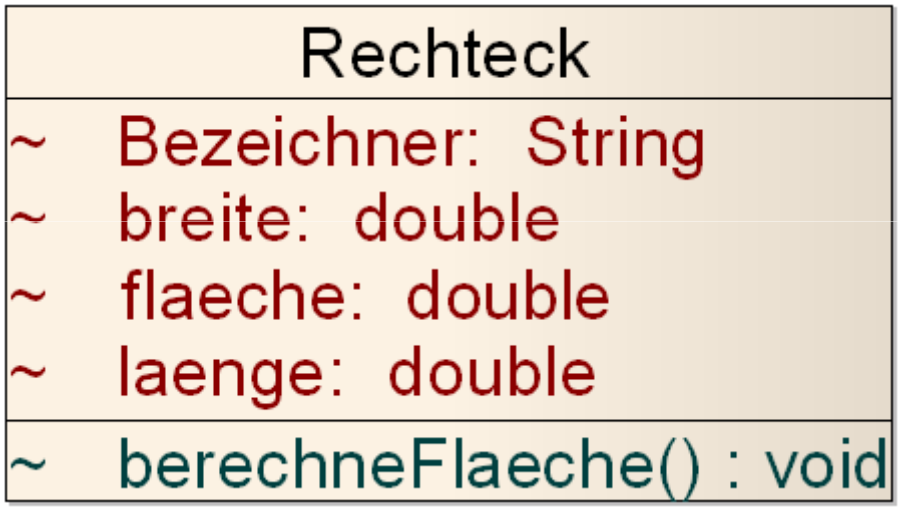
….so sieht‘s aus …….

Eigenschaften

(was wird abgespeichert?)

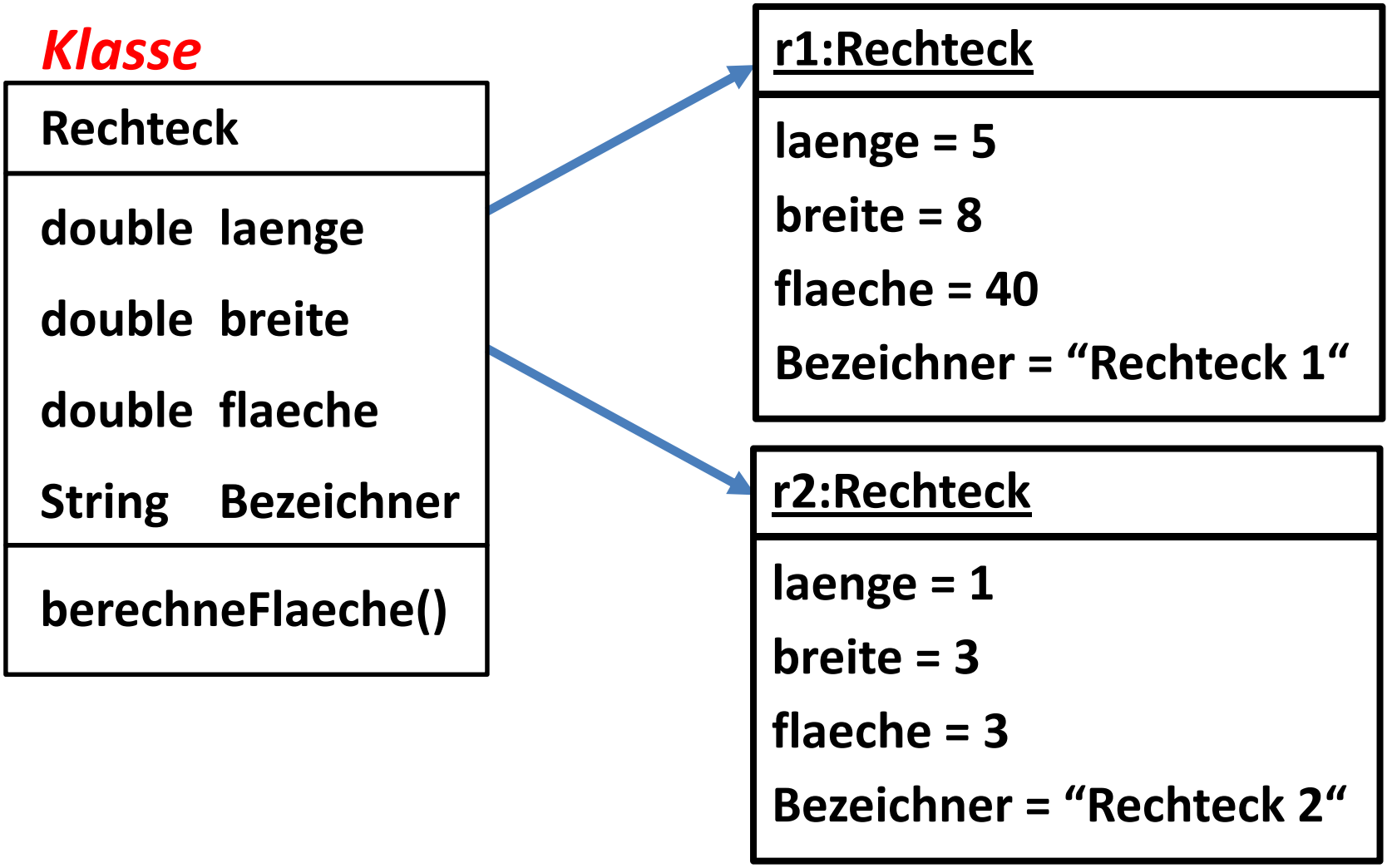
Fähigkeiten

(was kann ein Rechteck?)



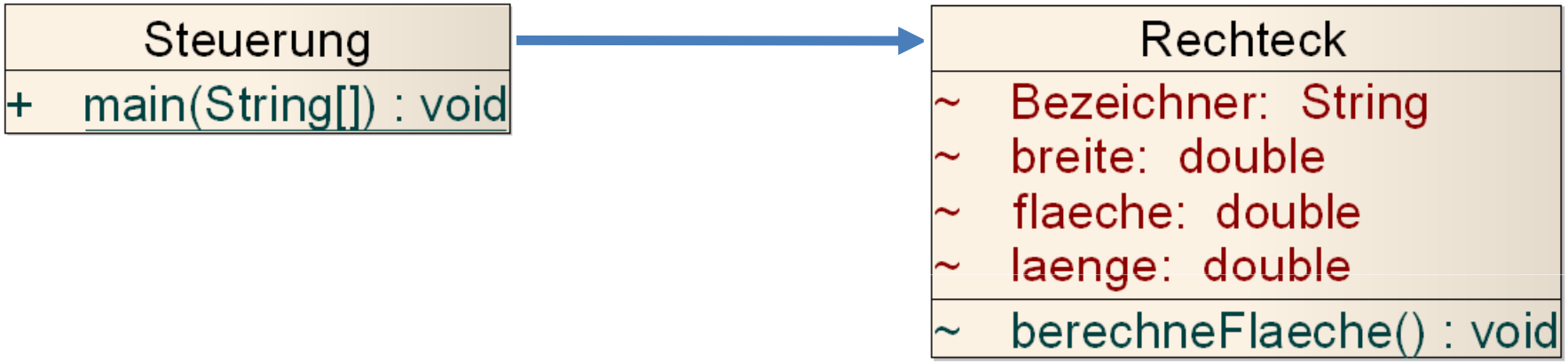
Der Weg zur Objektorientierung – ein Ansatz

….aus der Vorlage entstehen Objekte….



Der Weg zur Objektorientierung – ein Ansatz

Wie erzeugt man nun ein Objekt?




```
public class Steuerung {

    public static void main(String[] args) {

        Rechteck r1 = new Rechteck();

    }
}
```

Der Weg zur Objektorientierung – ein Ansatz

Wie erzeugt man nun ein Objekt?

```
public class Steuerung {

    public static void main(String[] args) {

        Rechteck r1 = new Rechteck();

    }
}
```

oder umgangssprachlich

Mach mir bitte aus der Klasse Rechteck ein Objekt mit Namen r1.

Zusammenhang Klasse ⟷ Objekt? - nochmal

Klasse

definiert die Eigenschaften und Fähigkeiten eines möglichen Objektes

Objektvorlage

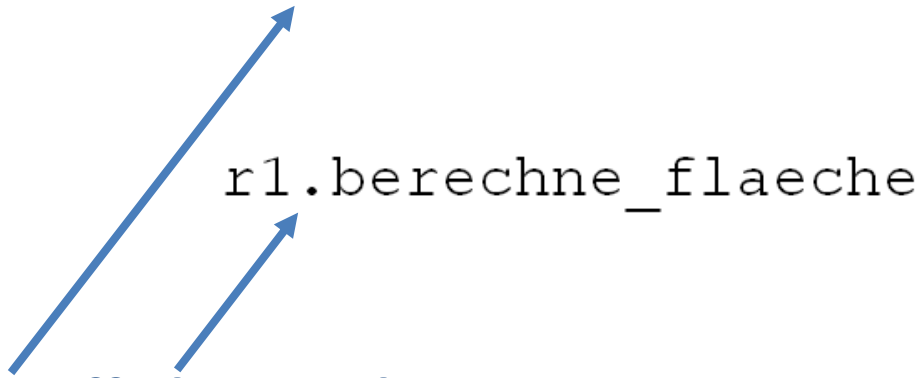
Objekt

real im Speicher existierendes Objekt, das über die in der Klasse definierten Eigenschaften und Fähigkeiten verfügt.

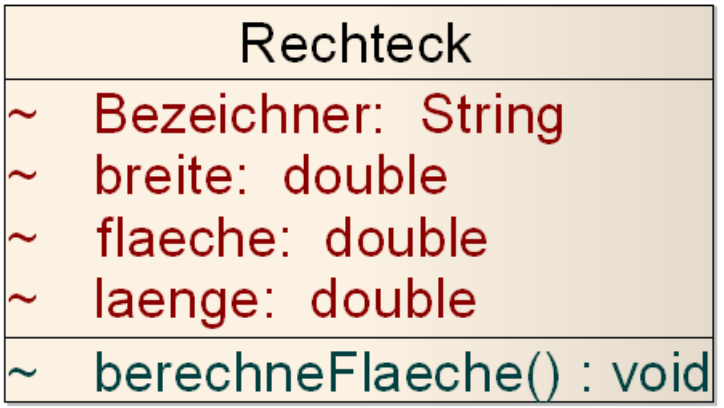
Der Weg zur Objektorientierung – ein Ansatz

Wie greife ich auf ein Objekt zu?

```
public class Steuerung {

    public static void main(String[] args) {

        Rechteck r1 = new Rechteck();

        r1.flaeche = 5;


        r1.berechne_flaeche();
```



Zugriff über Punktoperator



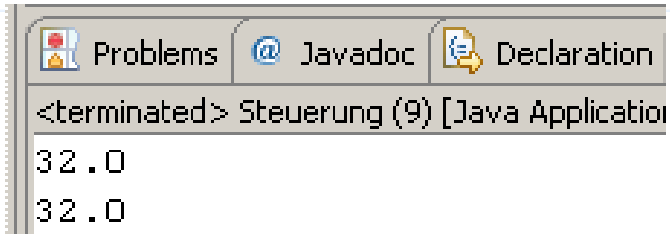
Der Weg zur Objektorientierung – ein Ansatz

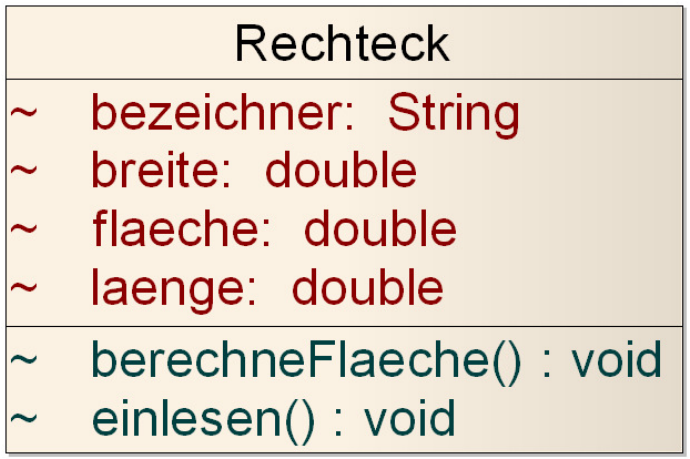
Ist dieser Klassenentwurf sinnvoll? – kleiner Test

```
public static void main(String[] args) {

    Rechteck r1 = new Rechteck();

    r1.breite = 4;
    r1.laenge = 8;

    r1.berechneFlaeche();

    IO.println(r1.flaeche);

    r1.breite++;

    IO.println(r1.flaeche);

}
```

flaeche wurde nicht aktualisiert

Ausgabe:





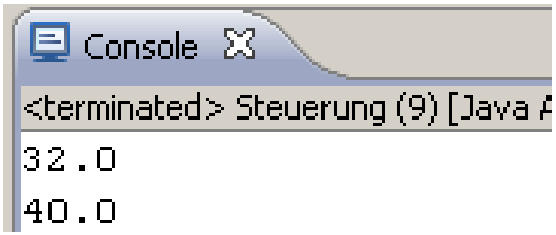
Der Weg zur Objektorientierung – stateless

Macht es Sinn die Fläche eines Rechtecks abzuspeichern?

```
public static void main(String[] args) {

    Rechteck r1 = new Rechteck();

    r1.laenge = 8;
    r1.breite = 4;

    IO.println(r1.getFlaeche());

    r1.breite++;

    IO.println(r1.getFlaeche())

}
```

Nein

Ausgabe:




```
public class Rechteck {

    double laenge;
    double breite;
    String bezeichner;

    double getFlaeche(){

        return laenge * breite;
    }
}
```

Der Weg zur Objektorientierung – stateless

Das Prinzip der zustandslosen Programmierung

```
public class Rechteck {

    double laenge;
    double breite;
    String bezeichner;

    double getFlaeche(){

        return laenge * breite;
    }
}
```

Die Fläche eines Objektes wird nicht abgespeichert, sondern bei jedem Aufruf neu berechnet.

Damit existieren nie veraltete Werte

Der Weg zur Objektorientierung

Was ist bei der Erstellung einer Klasse zu beachten?

***Namensgebung*:**

Soll eine Klasse Vorlage für Objekte vom Typ Rechteck sein, so muss der Klassenname dem Typ der Objekte entsprechen.

Gilt auch für den Dateinamen: ***Datei Rechteck.java***

```
public class Rechteck {

    double laenge;
    double breite;
    String bezeichner;

    double getFlaeche(){

        return laenge * breite;
    }
}
```

Klassenname darf nicht mit einer Zahl oder Sonderzeichen beginnen.

Der Weg zur Objektorientierung

Was ist bei der Erstellung einer Klasse zu beachten?

Verwenden Sie keine Umlaute bei der Wahl von Bezeichnern:

Auch wenn das Betriebssystem dies erlaubt:

Verzichten Sie darauf.

```
double höhe = 0;
```

```
double hoehe = 0;
```

✓

Kein Bezeichner, sondern Inhalt

```
String text = "Überraschung!";
```

✓

Der Weg zur Objektorientierung

Was ist bei der Erstellung einer Klasse zu beachten?

Wählen Sie für Bezeichner „sprechende Namen“

```
public class Rechteck {
```

lowerCaseCapitalCamelCase-Schreibweise

```
    double laengeInCm;
    double breiteInCm;
```

Eigenschaften und Methoden beginnen mit Kleinbuchstaben

```
    void berechneFlaeche(){

        flaeche = laenge * breite;
    }
}
```

Für Objektbezeichner → CapitalCamelCase-Schreibweise