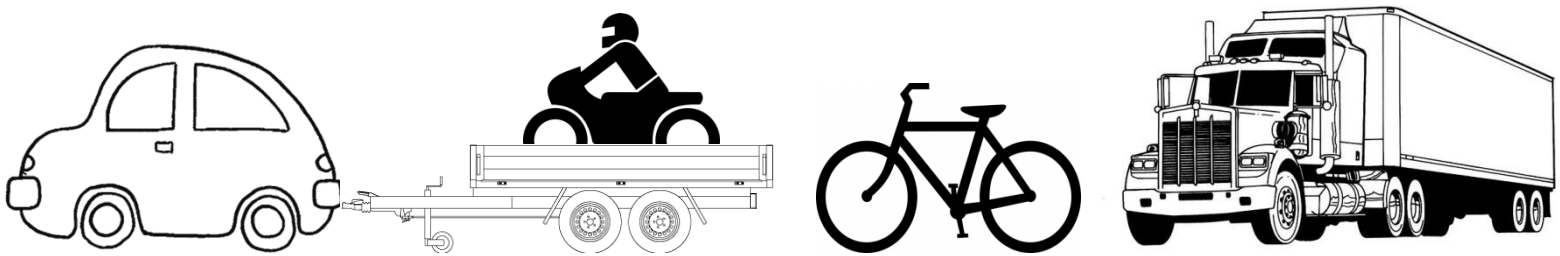
Sichtbarkeit, Vererbung, Methoden (Teil 2), Konstruktor

Inhalt - Überblick

1. **Java Grundlagen: Entwicklungszyklus, Entwicklungsumgebung**
2. Datentypen, Kodierung, Binärzahlen, Variablen, Arrays
3. Ausdrücke, Operatoren, Schleifen und Verzweigungen
4. Blöcke, Sichtbarkeit und Methoden (Teil 1)
5. Grundkonzepte der Objektorientierung
6. Objektorientierung: Sichtbarkeit, Vererbung, Methoden (Teil 2), Konstruktor
7. Packages, lokale Klassen, abstrakte Klassen und Methoden, Interfaces, enum
8. Arbeiten mit Objekten: Identität, Listen, Komparatoren, Kopien, Wrapper, Iterator
9. Fehlerbehandlung: Exceptions und Logging
10. Utilities: Math, Date, Calendar, System, Random
11. Rekursion, Sortieralgorithmen und Collections
12. Nebenläufigkeit: Arbeiten mit Threads
13. Benutzeroberflächen mit Swing
14. Streams: Auf Dateien und auf das Netzwerk zugreifen

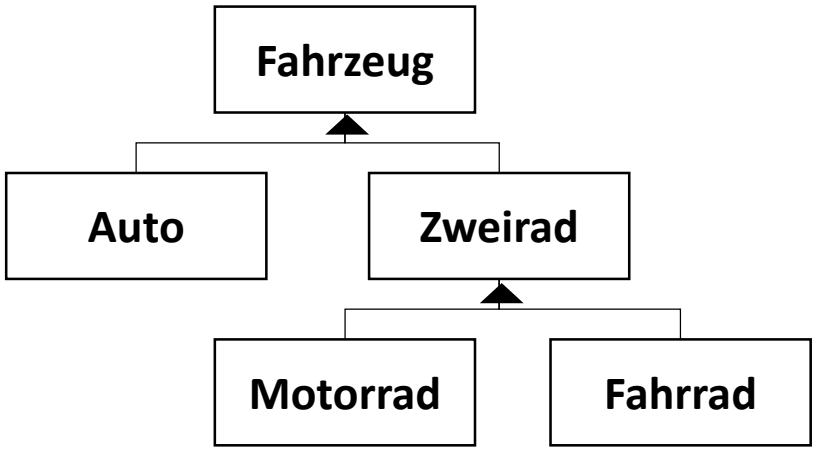
Es gibt drei Beziehungsarten:

1. „is-a“-Beziehung (Generalisierung, Spezialisierung)
2. „part-of“-Beziehung (Aggregation, Komposition)
3. Verwendungs- und Aufrufbeziehung



Generalisierung und Spezialisierung:

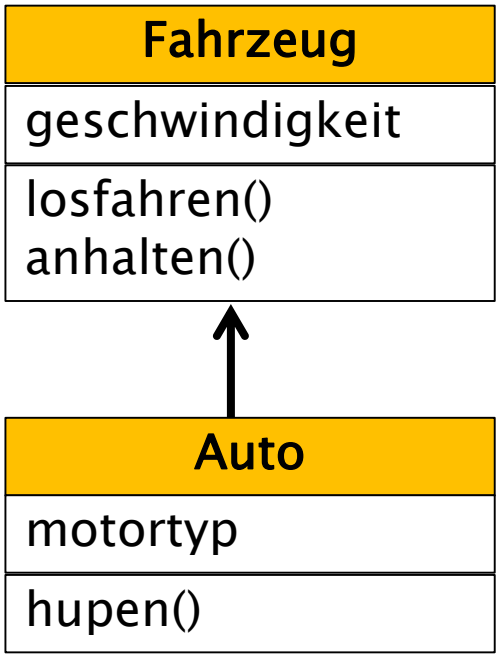
- Ein Auto ist ein Fahrzeug
- Ein Zweirad ist ein Fahrzeug
- Ein Fahrrad und ein Motorrad sind Zweiräder



Vererbung

Mit Hilfe der Vererbung werden Eigenschaften vorhandener Klassen auf neue Klassen übertragen.

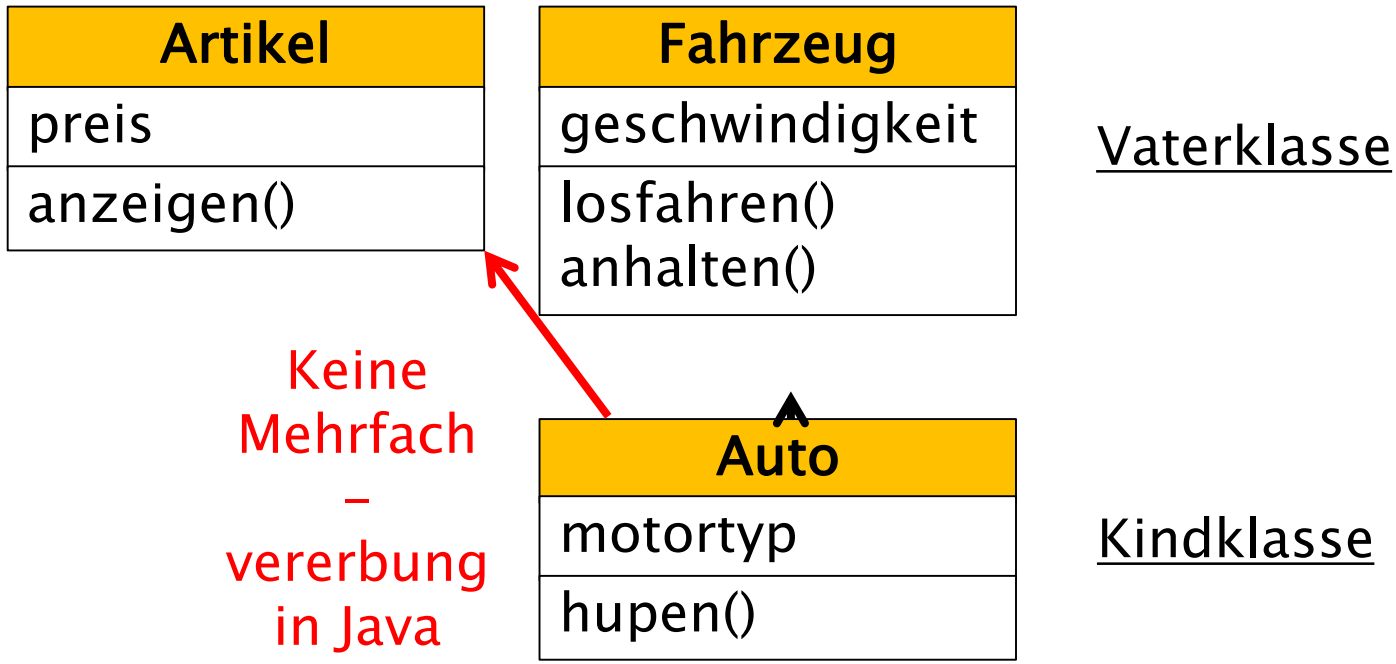
Ein Klasse kann von einer anderen abgeleitet sein.



Vaterklasse

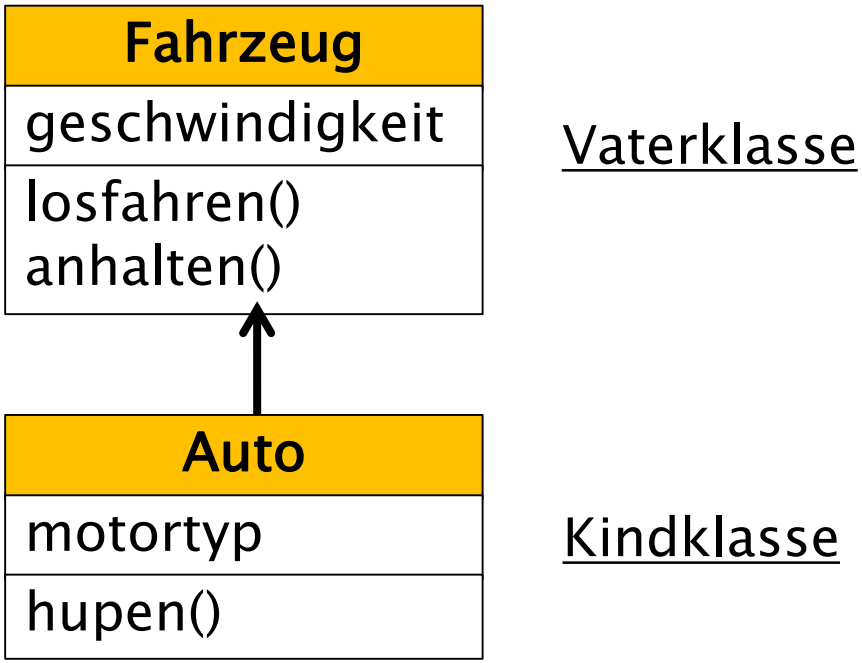
Kindklasse

Einfachvererbung (Java) vs. Mehrfachvererbung



Vererbung implementieren

```
2 public class Auto extends Fahrzeug {
```

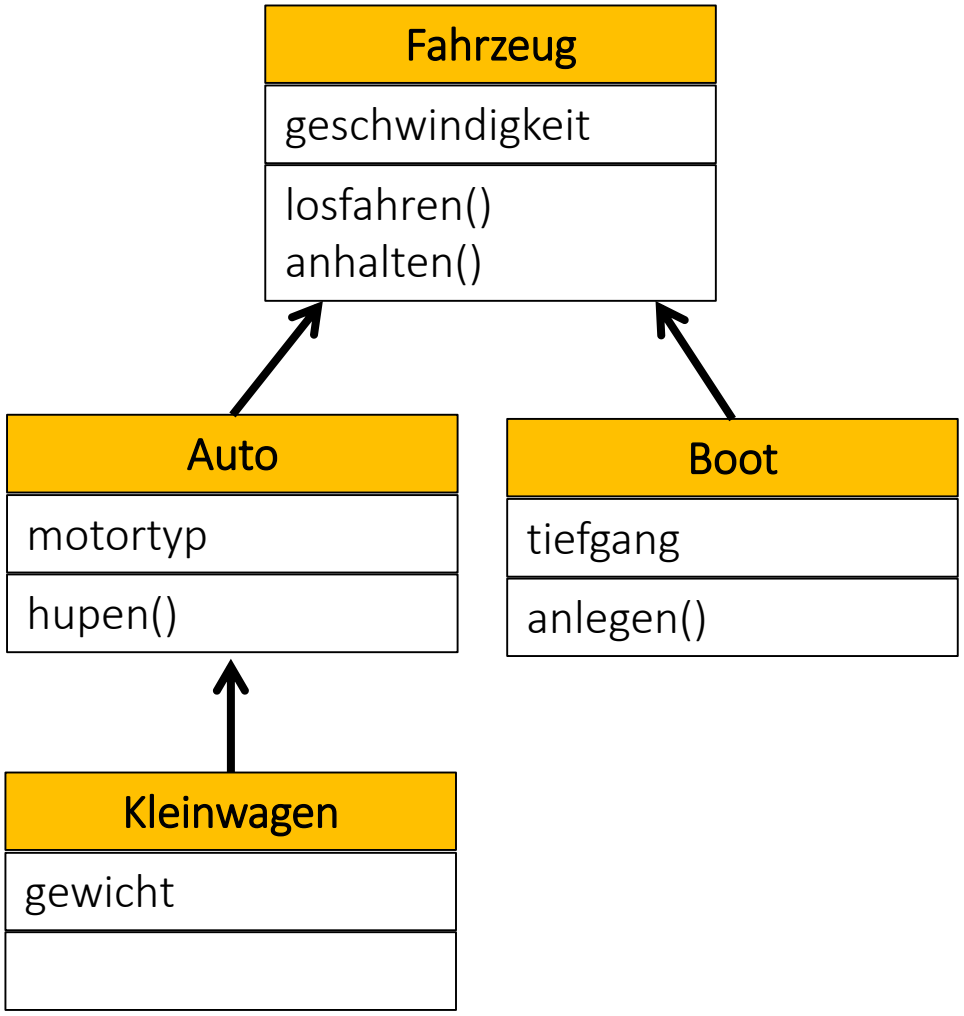


[Fahrzeug.java, Auto.java]

Vererbungsbaum

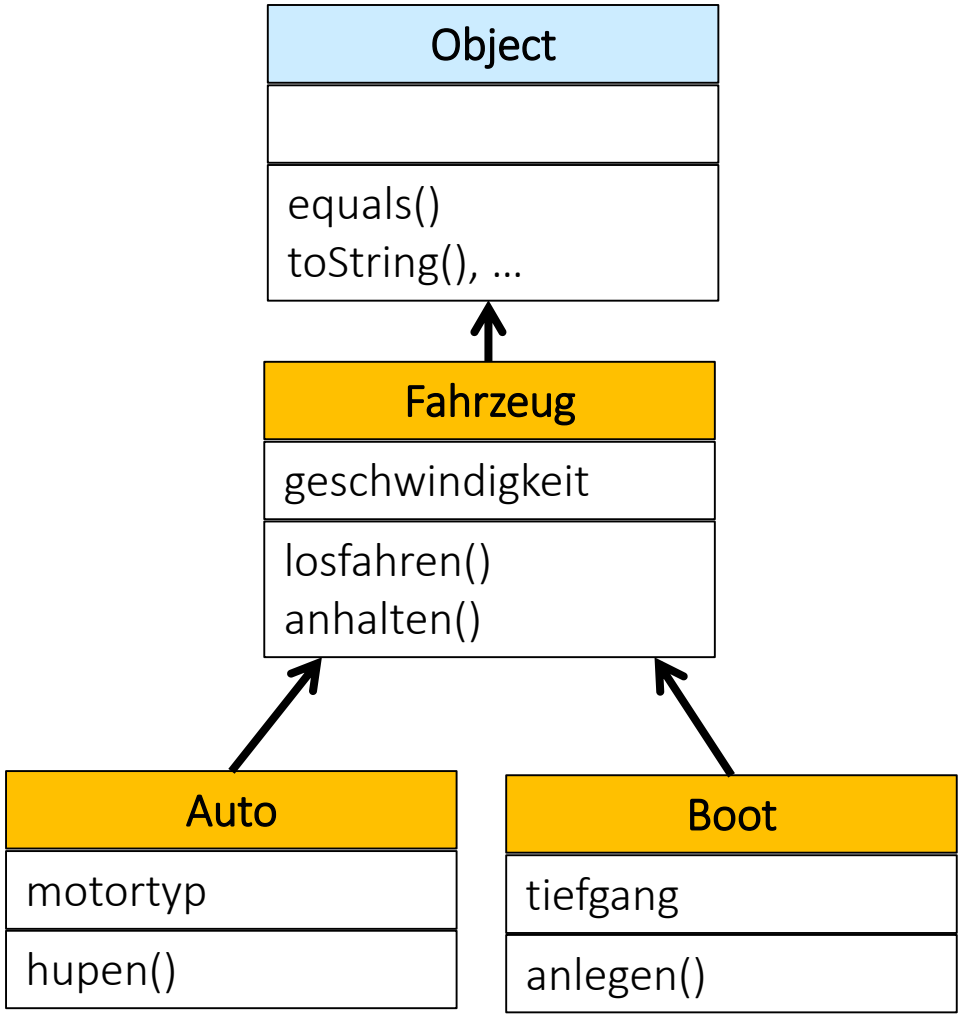
Eine Vaterklasse kann beliebig viele Kindklassen haben

Eine Kindklasse kann auch Vaterklasse für andere Klassen sein



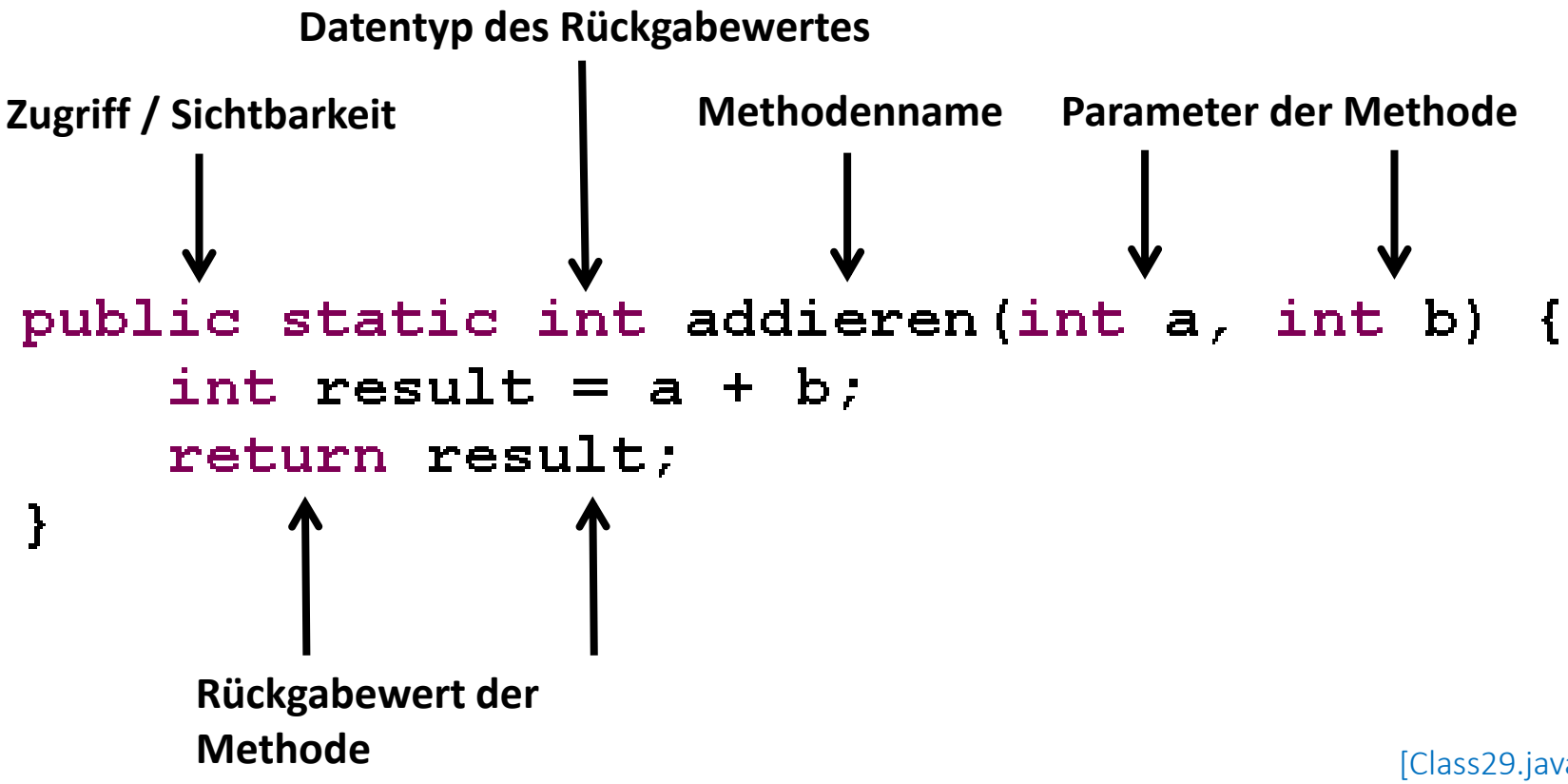
Object-Klasse

Die Klasse Object ist in der Java API enthalten und stellt die Wurzel-Klasse (root) **jeder** Vererbungs-hierarchie dar



Sichtbarkeit

Aufbau einer Methode

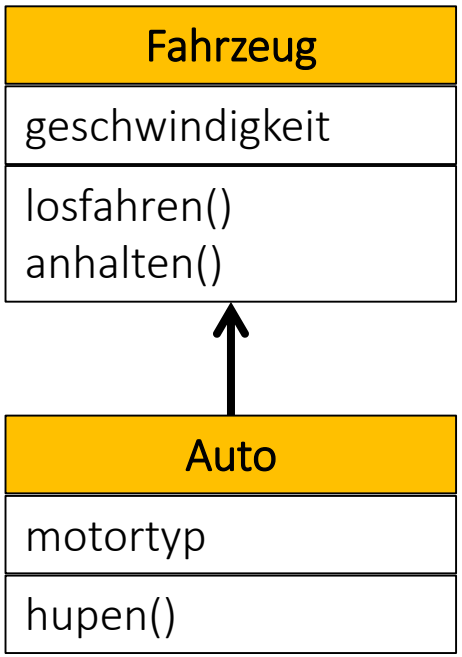


[Class29.java]

Eine Kindklasse kann

Methoden und Variablen

der Vaterklasse(n) erben



Vaterklasse

Außerdem können

Methoden und Variablen

von anderen Klassen

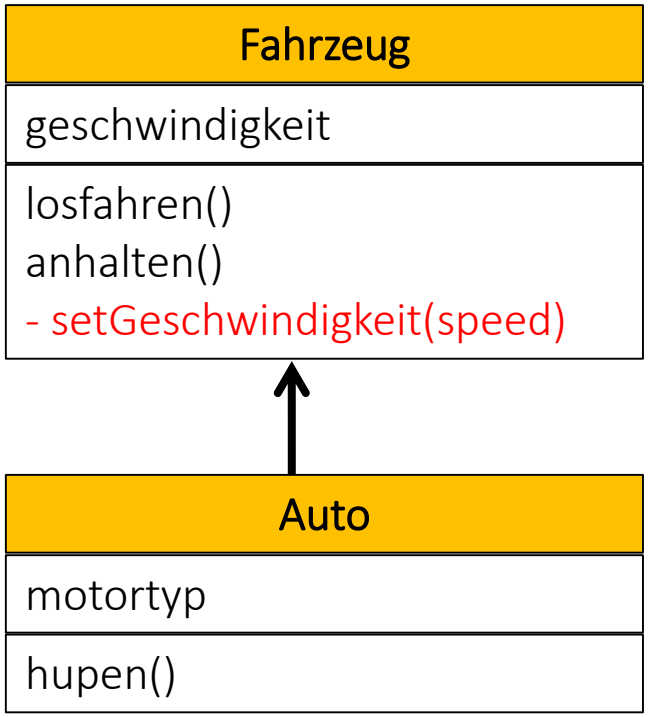
genutzt werden

Kindklasse

Elemente des Typs **public** sind in der Klasse selbst, in Methoden abgeleiteter Klassen und für den Aufrufer von Instanzen der Klasse sichtbar

Elemente des Typs **private** sind in der Klasse selbst sichtbar. Für abgeleitete Klassen und Aufrufer sind diese verdeckt.

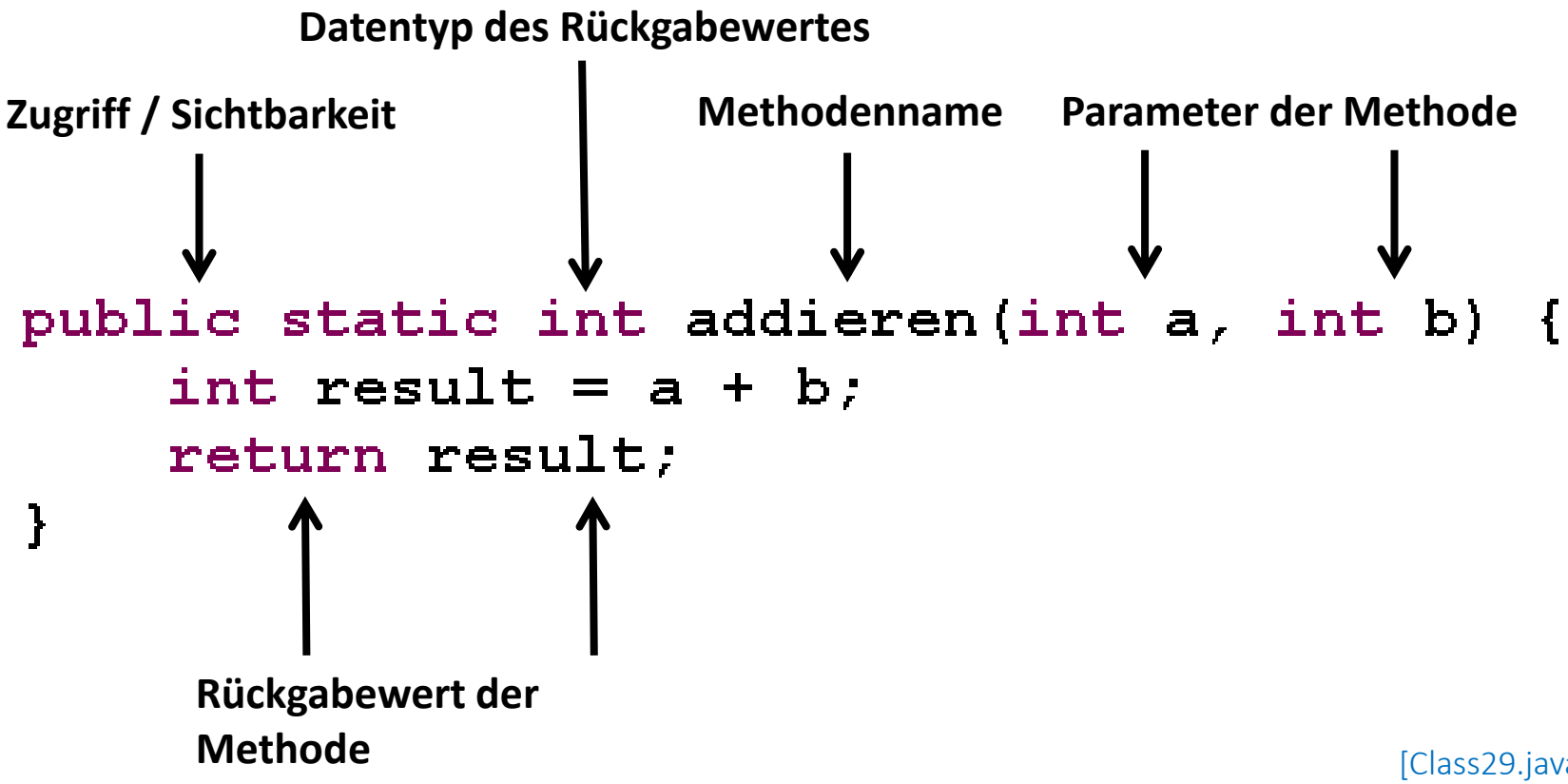
[default- / package-scope, protected]



[Fahrzeug.java, Auto.java]

Statische Methoden und Instanzvariablen

Aufbau einer Methode



[Class29.java]

Klassen- vs. Instanz**variablen**

- Eine **Klassenvariable** existiert je **Klasse** genau einmal
- Eine **Instanzvariable** existiert je **Objekt / Instanz** genau einmal
- Wird bei der Deklaration eines Attributs der Modifier **`static`** verwendet, handelt es sich um eine Klassenvariable. Klassenvariablen sind für alle Instanzen identisch.
- Ohne die Angabe des **`static`**-Modifiers handelt es sich um eine Instanzvariable

Klassen- vs. Instanz**variablen**

```
1  public class Tier {
2
3      public String art;
4      public double gewicht;
5      public String farbe;
6
7      public static int anzahlTiere;
8
9      public Tier() {
10         anzahlTiere = anzahlTiere + 1;
11     }
12 }
```

Instanzvariablen

Klassenvariable

Klassen- vs. Instanz**variablen**

- Auf **Klassenvariablen** wird über den **Klassennamen** zugegriffen
- Auf **Instanzvariablen** wird über den **Instanzen / Objekte** zugegriffen

```
1  public class Main {
2      public static void main(String[] args) {
3
4          Tier kuh     = new Tier();
5          kuh.art      = "Fleckvieh";
6          kuh.gewicht  = 750;
7          kuh.farbe    = "Weichselbraun";
8
9          System.out.println("Tieranzahl: " + Tier.anzahlTiere);
10     }
11 }
```

Zugriff auf Instanzvariable

Zugriff auf Klassenvariable

Klassen- vs. Instanz**methoden**

- Wird bei der Deklaration einer Methode der Modifier **`static`** verwendet, handelt es sich um eine **Klassenmethode**
- Ohne die Angabe des **`static`**-Modifiers handelt es sich um eine **Instanzmethode**
- **Klassenmethoden** können **nur** auf **Klassenvariablen** und **Klassenmethoden** zugreifen (**nicht** auf Instanzvariablen oder -methoden)
- **Instanzmethoden** können sowohl auf **Klassen**- als auch auf **Instanzvariablen** und -methoden zugreifen

Klassen- vs. Instanz**methoden**

```
1  public class Tier {
2      public String art;
3      public double gewicht;
4      public String farbe;
5
6      public static int anzahlTiere;
7
8      public double getGewicht() {
9          return gewicht;
10     }
11
12     public int getAnzahlTiereInstanceMethod() {
13         return anzahlTiere;
14     }
15
16     public static int getAnzahlTiere() {
17         return anzahlTiere;
18     }
19 }
```

Instanzmethoden

Klassenmethode

Klassen- vs. Instanz**methoden**

- Auf **Klassenmethoden** wird über den **Klassennamen** zugegriffen
- Auf **Instanzmethoden** wird über den **Instanznamen** zugegriffen

```
1  public class Main {
2    public static void main(String[] args) {
3
4      Tier kuh     = new Tier();
5      kuh.art      = "Fleckvieh";
6      kuh.gewicht  = 750;
7      kuh.farbe    = "Weichselbraun";
8
9      System.out.println("Tieranzahl: " + Tier.getAnzahlTiere());
10     System.out.println("Gewicht: " + kuh.getGewicht());
11   }
12 }
```

Zugriff auf Klassenmethode

Zugriff auf Instanzmethode

This-Operator

This-Operator

- Mithilfe des **`this`**-Operators kann innerhalb einer Instanz-Methode (oder eines Konstruktors) auf die Attribute und Methoden einer Instanz zugegriffen werden
- Durch diesen Operator können lokale Variablen und Instanzvariablen mit gleichem Namen voneinander unterschieden werden

This-Operator

```
1  public class Tier {
2
3      private String art;
4
5      public Tier(String art) {
6          this.art = art;
7      }
8  }
```

Instanzvariable mit dem Namen **art**

Lokale Variable **art**

this-Operator stellt den Bezug zur Instanzvariable her

Konstruktor

Konstruktor

- Spezielle Methode, die bei der Initialisierung eines Objekts (bzw. einer Instanz) aufgerufen wird.
- In Java sind Konstruktoren Methoden ohne Rückgabewert (vgl. void)
- Der Name des Konstruktors **muss** mit dem Namen der Klasse übereinstimmen
- Konstruktoren können beliebig viele Parameter haben
- Konstruktoren werden aufgerufen, sobald eine neue Instanz einer Klasse mit dem **`new`**-Operator erstellt wird

Konstruktor: Beispiel

```
1  public class Auto {
2
3      public String   name;
4      public int      leistung;
5      public double   verbrauch;
6      public String   farbe;
7
8      public Auto() {
9          name        = "BMW";
10         leistung    = 200;
11         verbrauch   = 7.7;
12         farbe       = "Pink";
13     }
14 }
```

Konstruktor: Aufruf

```
1 public class Main {
2
3     public static void main(String[] args) {
4
5         Auto auto1  = new Auto();
6     }
7 }
```