PGdP

Parameter

Casts

Vererbung

Polymorphie I

P-Aufgaben

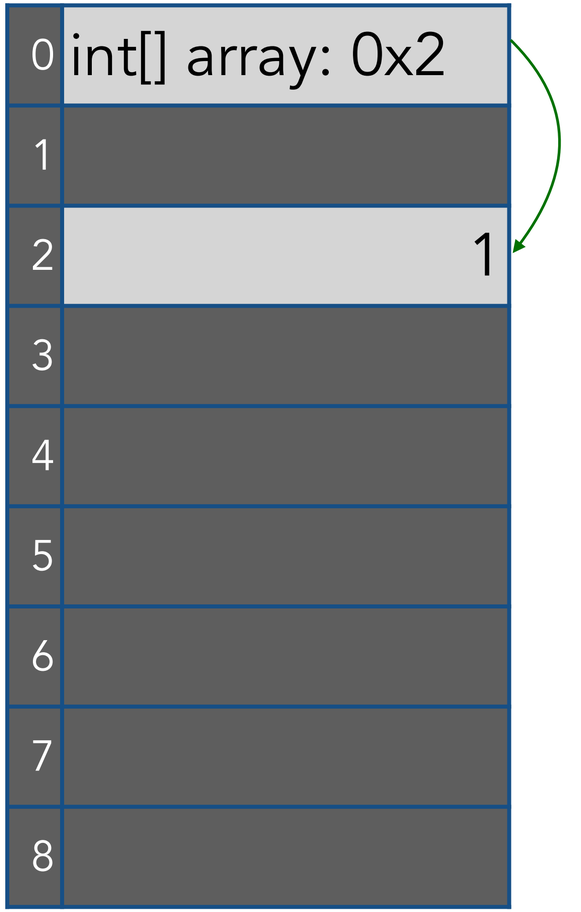
Folien: go.tum.de/904005

Tobias Schamel

Parameter

Java ist 'Call-by-Value' > Kopie des Wertes wird übergeben

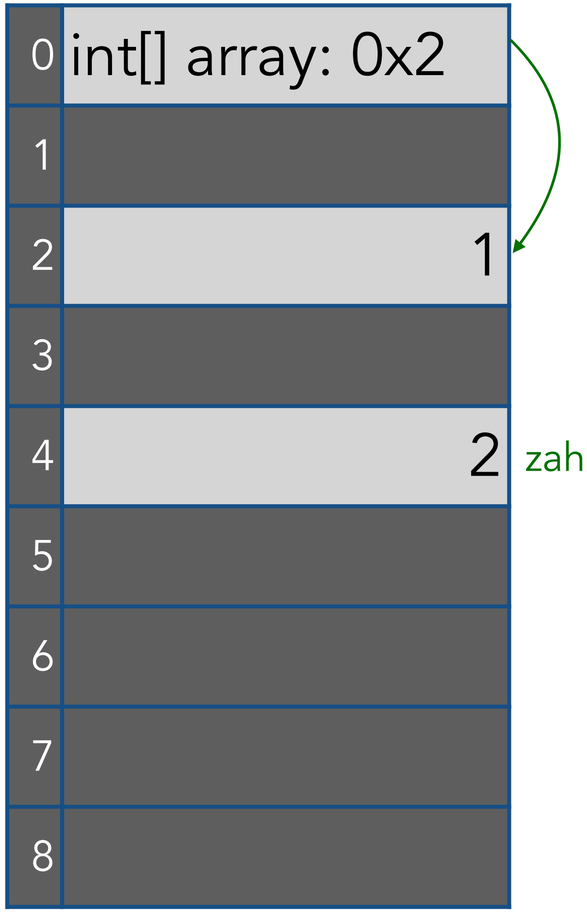
```
1   int[] array = {1};
2   int zahl = 2;
3   public static void
4    changeWert(int[] a, int b) {
5     a[0] = 0;
6     b = 0;
7   }
8
9   changeWert(array, zahl);
10  write("Array: " + array[0]
11   + ", Zahl: " + zahl;
12
```



Parameter

Java ist 'Call-by-Value' > Kopie des Wertes wird übergeben

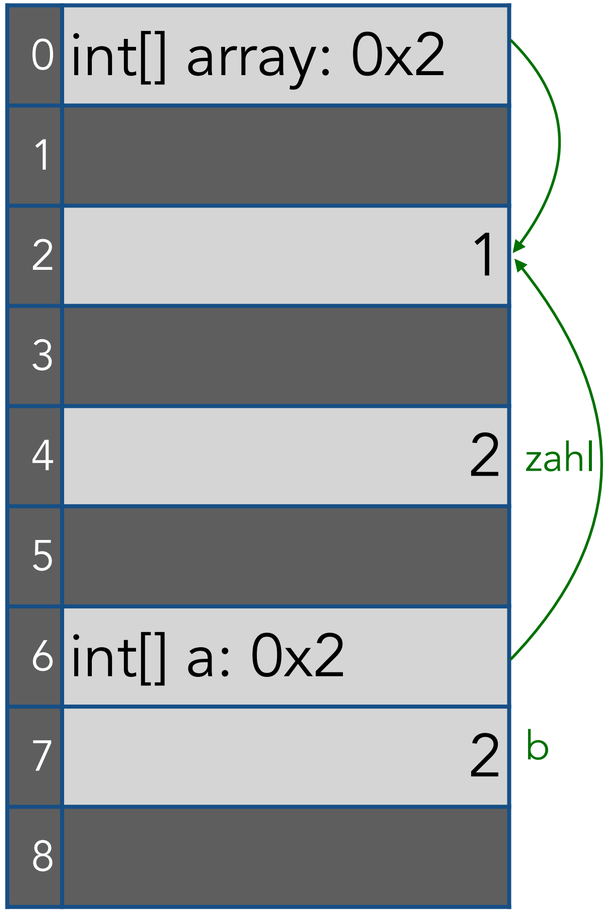
```
1   int[] array = {1};
2   int zahl = 2;
3   public static void
4    changeWert(int[] a, int b) {
5     a[0] = 0;
6     b = 0;
7   }
8
9   changeWert(array, zahl);
10  write("Array: " + array[0]
11   + ", Zahl: " + zahl;
12
```



Parameter

Java ist 'Call-by-Value' > Kopie des Wertes wird übergeben

```
1   int[] array = {1};
2   int zahl = 2;
3   public static void
4    changeWert(int[] a, int b) {
5     a[0] = 0;
6     b = 0;
7   }
8
9   changeWert(array, zahl);
10  write("Array: " + array[0]
11   + ", Zahl: " + zahl;
12
```

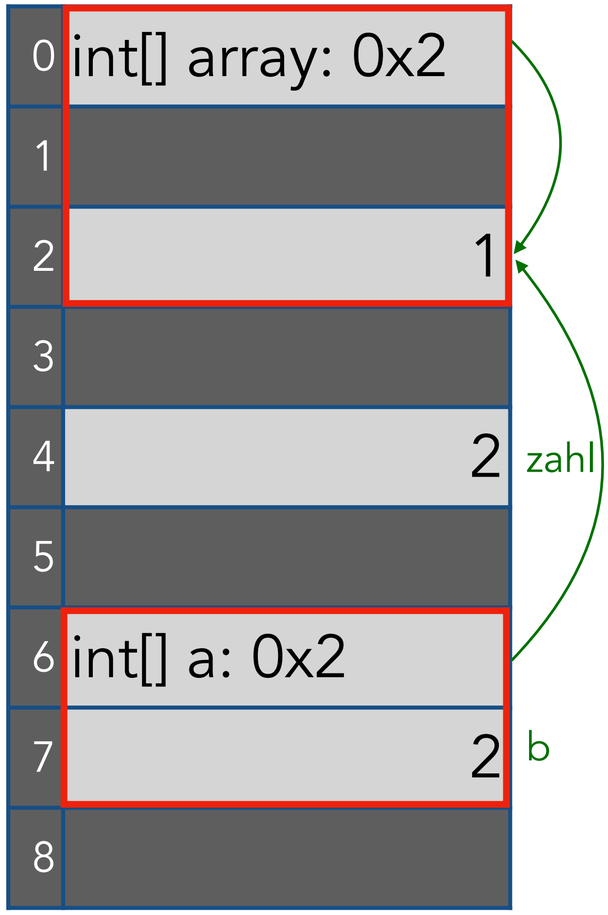


Parameter

Java ist 'Call-b … rtes wird übergeben

Von den beiden Parametern wird eine Kopie erstellt.

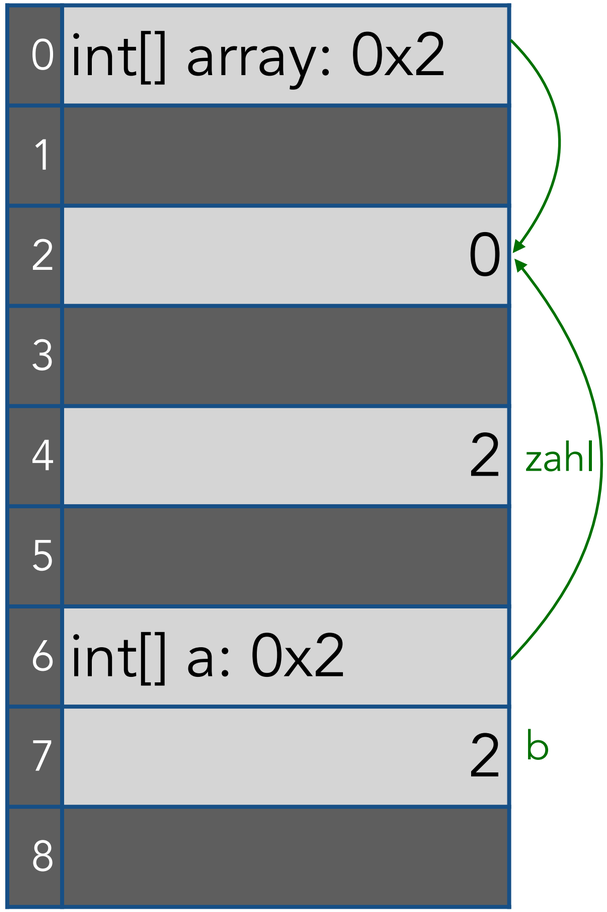
```
1   int[] a
2   int zah
3   public
4     changeWert(int[] a, int b) {
5       a[0] = 0;
6       b = 0;
7   }
8
9   changeWert(array, zahl);
10  write("Array: " + array[0]
11    + ", Zahl: " + zahl;
12
```



Parameter

Java ist 'Call-by-Value' > Kopie des Wertes wird übergeben

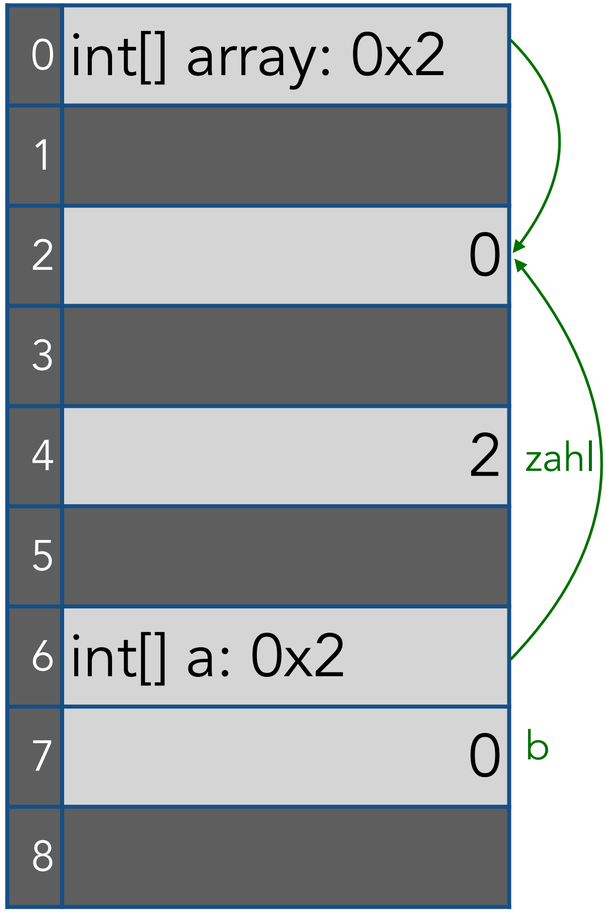
```
1   int[] array = {1};
2   int zahl = 2;
3   public static void
4    changeWert(int[] a, int b) {
5     a[0] = 0;
6     b = 0;
7   }
8
9   changeWert(array, zahl);
10  write("Array: " + array[0]
11   + ", Zahl: " + zahl;
12
```



Parameter

Java ist 'Call-by-Value' > Kopie des Wertes wird übergeben

```
1   int[] array = {1};
2   int zahl = 2;
3   public static void
4    changeWert(int[] a, int b) {
5     a[0] = 0;
6     b = 0;
7   }
8
9   changeWert(array, zahl);
10  write("Array: " + array[0]
11   + ", Zahl: " + zahl;
12
```



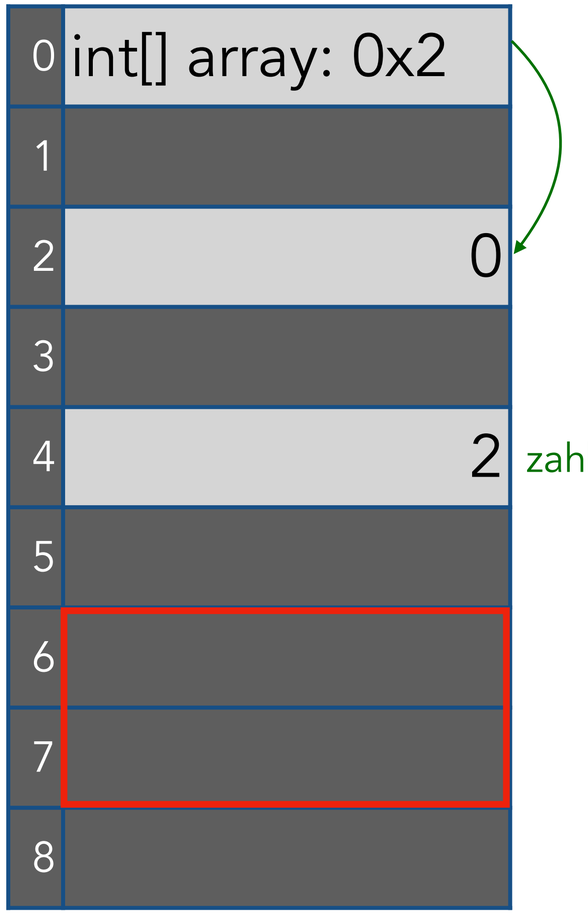
Parameter

Java ist 'Call-b … rtes wird übergeben

Beim Verlassen der Methoden verschwinden die Variablen/Parameter aus dem Block!

```
1   int[] a
2   int zah
3   public
4    change                    {
5     a[0]
6     b = 0
7   }
8
9   changeWert(array, zahl);
10  write("Array: " + array[0]
11   + ", Zahl: " + zahl;
12
```

```
Array: 0, Zahl: 2
```

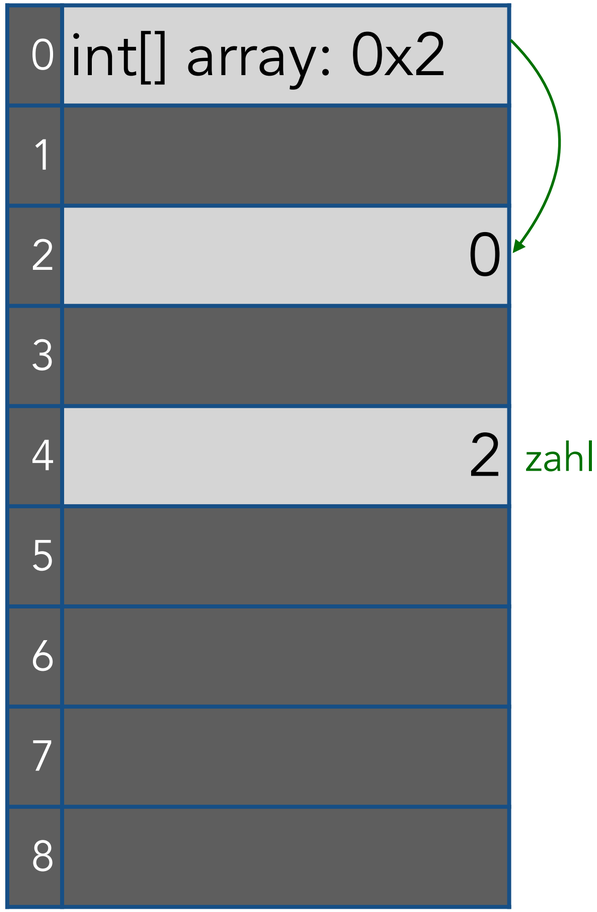


Parameter

Java ist 'Call-by-Value' > Kopie des Wertes wird übergeben

```
1   int[] array = {1};
2   int zahl = 2;
3   public static void
4    changeWert(int[] a, int b) {
5     a[0] = 0;
6     b = 0;
7   }
8
9   changeWert(array, zahl);
10  write("Array: " + array[0]
11   + ", Zahl: " + zahl;
12
```

```
Array: 0, Zahl: 2
```



Parameter

Java ist 'Call-by-Value' > Kopie des Wertes wird übergeben

- Innerhalb der Methode wird eine **Kopie eines primitiven Datentypen** verändert > keine Veränderung des Originals
- Innerhalb einer Methode wird mithilfe einer **Kopie der Referenz auf einen komplexen Datentyp** die Stelle im Speicher verändert, an der sich das Original befindet > Veränderung des Originals

Casts

Letzte Woche:

```
encrypted[i] = (char)     (current+key);
```

expliziter Typecast zu char — char — int

Java führt implizite Typecasts durch, um einen Datentypen festzulegen, mit dem gerechnet werden kann.

z.B. **`int`** + **`short`** $\Longrightarrow$ **`int`**

Verwendung von Casts

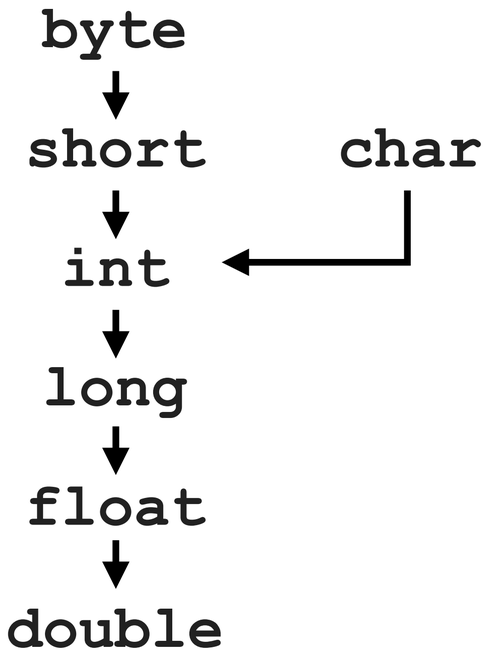
```
(<datentyp>) (<expr>)
```

Implizit bei Zuweisung

```
<datentyp> <varname> = (<datentyp>) (<expr>)
```

Casts

Implizite Typecasts (Hierarchie):



Achtung beim Rechnen: Rundungsfehler!

`boolean`

Vererbung

- Gemeinsame Methoden und Attribute müssen nicht doppelt implementiert werden.
  > Zugriff auch über Unterklasse möglich
- Sie werden in einer gemeinsamen Oberklasse angelegt.

```
public class Auto extends Fahrzeug {
  //Implementierung
}
public abstract class Fahrzeug {
  //Implementierung
}
```

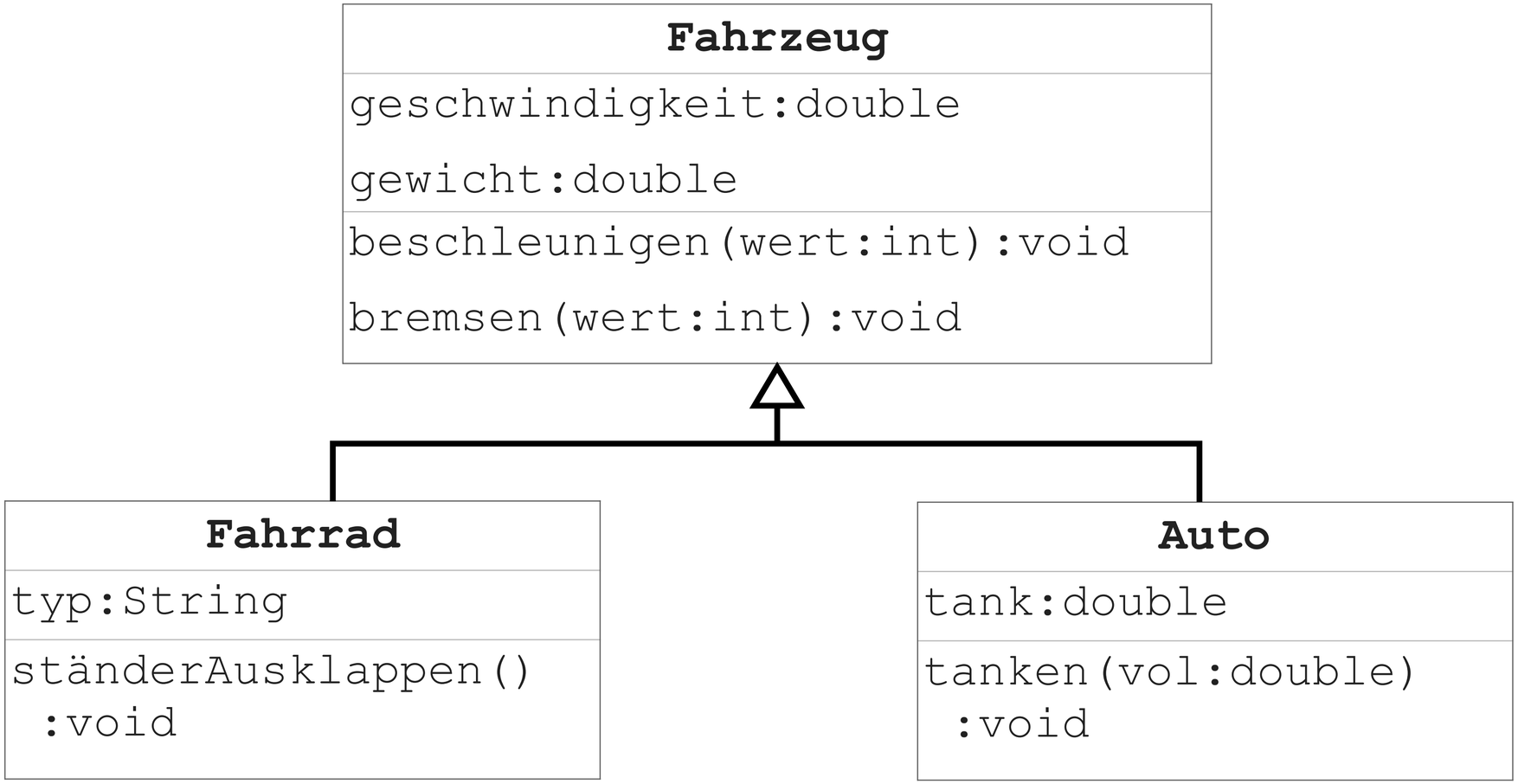
Vererbung

Abstraktion

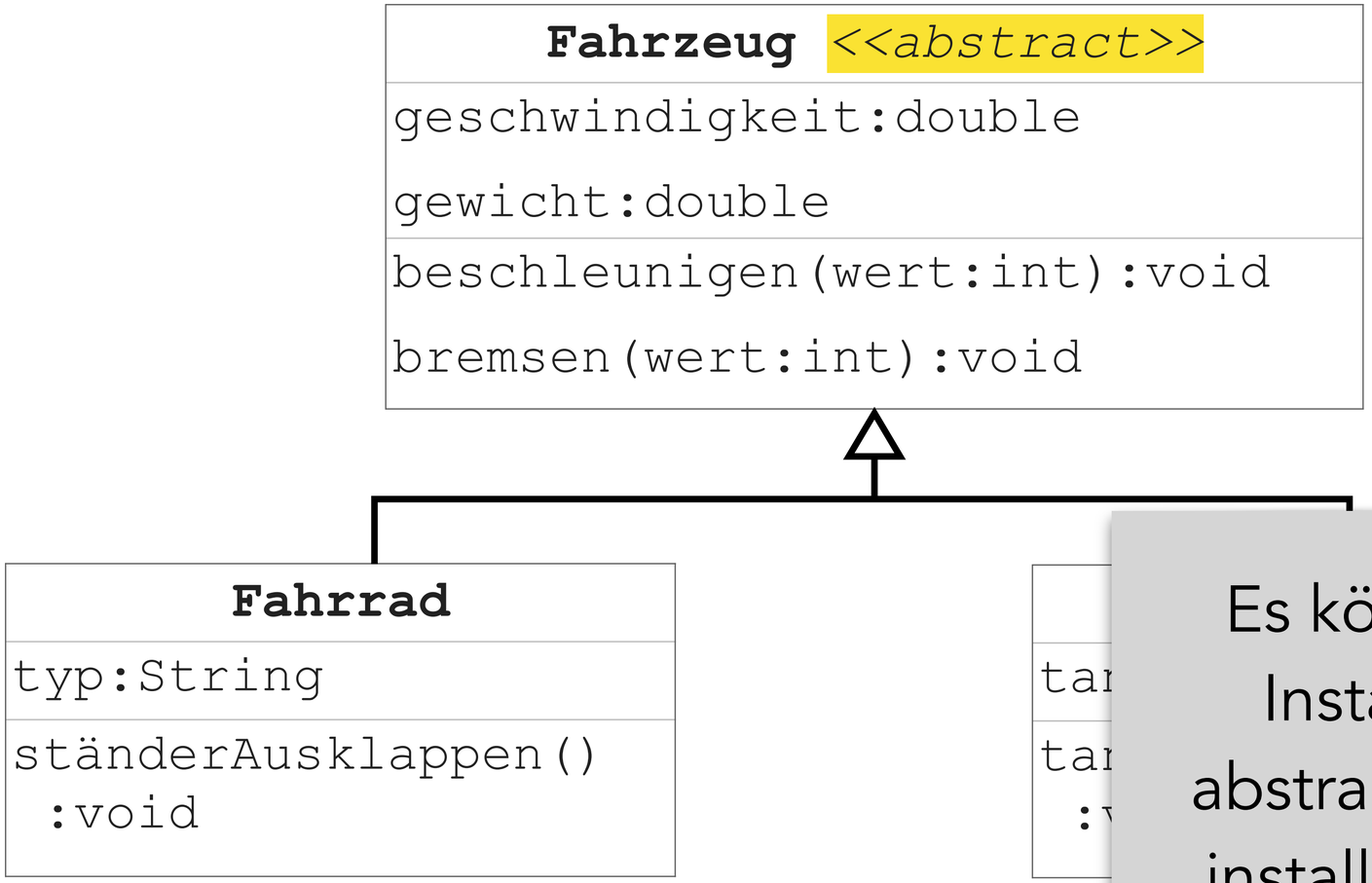
Vererbung

Fahrzeug
`geschwindigkeit:double` `gewicht:double`
`beschleunigen(wert:int):void` `bremsen(wert:int):void`

Vererbung

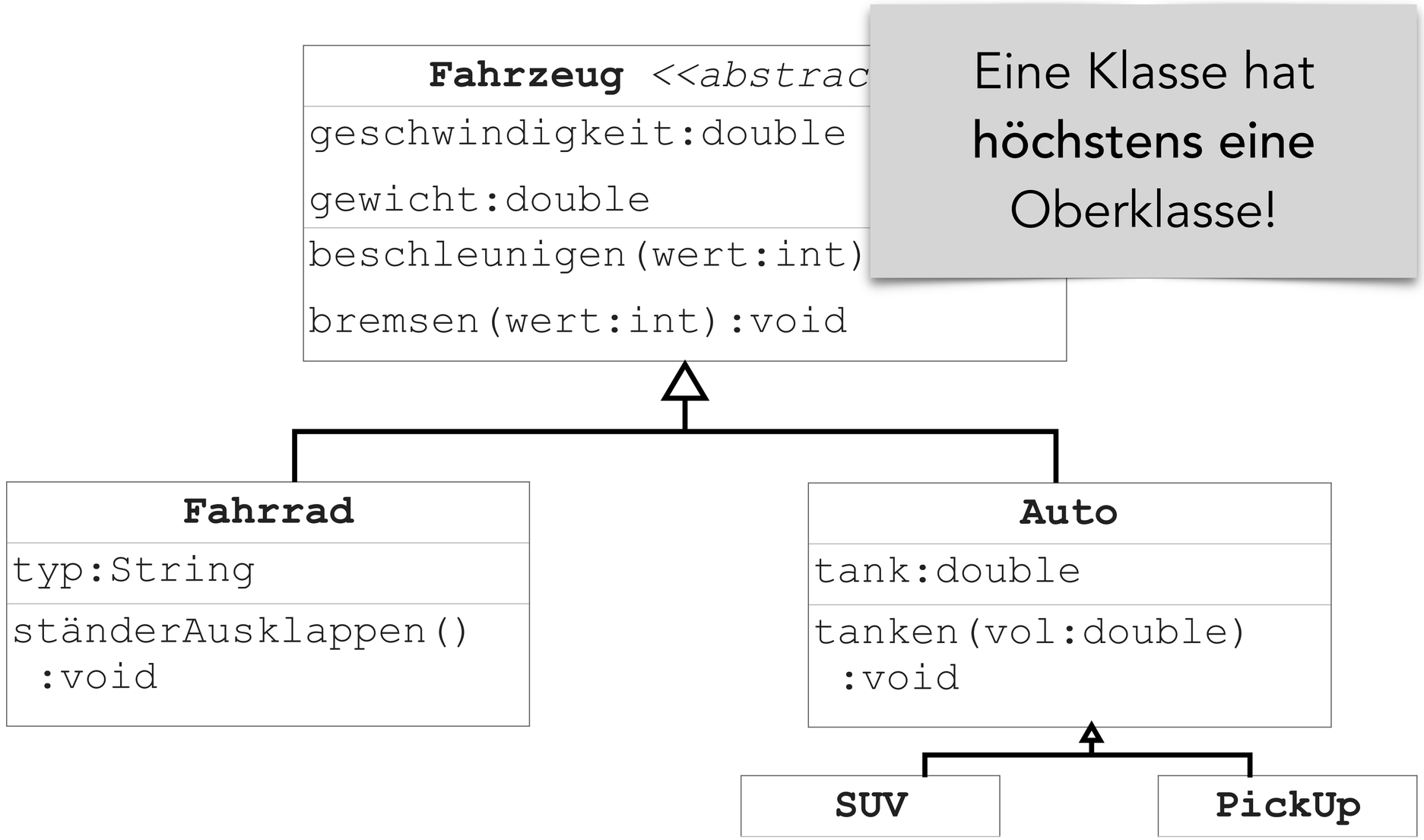

Fahrzeug
geschwindigkeit:double
gewicht:double
beschleunigen(wert:int):void
bremsen(wert:int):void
Fahrrad
typ:String
ständerAusklappen()
:void
Auto
tank:double
tanken(vol:double)
:void

Vererbung



Es können keine Instanzen von abstrakten Klassen installiert werden.

Vererbung



Eine Klasse hat **höchstens eine** Oberklasse!

Vererbung

Das Schlüsselwort *super* zum Zugriff auf Oberklasse:

```
public abstract class Fahrzeug {
  double geschwindigkeit, gewicht;
  public Fahrzeug(double gs, double gw) {
    this.geschwindigkeit = gs;
    this.gewicht = gw;
  }
  public void beschleunigen(double wert) {
    if(wert => 0) this.geschwindigkeit+=wert;
  }
}
```

Vererbung

Das Schlüsselwort *super* zum Zugriff auf Oberklasse:

```
public class Auto extends Fahrzeug {
   double tank;
   public Auto(double tank, double gs,
    double gw) {
      this.tank = tank;
      super(gs, gw);
   }
}
```

Aufruf des Konstruktors der Oberklasse

Mit super.*attribut* und super.*methode()* kann man auf Attribute & Methoden der Oberklasse zugreifen.

Vererbung

Überschreiben von Methoden

```
public class Auto extends Fahrzeug {
  //...
  public void beschleunigen(double wert) {
    if(wert >= this.tank && wert >= 0) {
      this.tank-=wert;
      this.geschwindigkeit+=wert;
    }
  }
}
```

'super' nicht nötig, da die Unterklasse (automatisch) über das Attribut verfügt

Für ein Auto a1 kann man mit

`a1.super.beschleunigen(x)`

die Methode der Oberklasse aufrufen.

Polymorphie I

Eine Instanz einer Unterklasse kann auch im statischen Referenztyp der Oberklasse gespeichert werden.

```
Fahrzeug a1 = new Auto(0, 1000, 20);
Auto a2 = new Auto(0, 1200, 15);
```

Statischer Typ — Dynamischer Typ

Verfügbare Methoden & Attribute werden durch den statischen Typen eines Objekts bestimmt.

```
a1.tanken(20);  //Compilerfehler
a2.tanken(20);
```

P07.03

Fehler finden: In jeder Methode ist ein Fehler. Finde ihn!

kahoot.it

by Lukas Gerhardt

P07.01

Recap: Stack

- FIFO (First-In-First-Out)
- zwei Operationen

```
push(data); //auf den Stack legen
pop();  //vom Stack nehmen
```

P07.01

Stack, Überprüfung korrekter Klammerung.

cB ::= (cB)|[cB]|{cB}|cB|ϵ

```
([])(){([])}   ✓
([)]           ✗
```

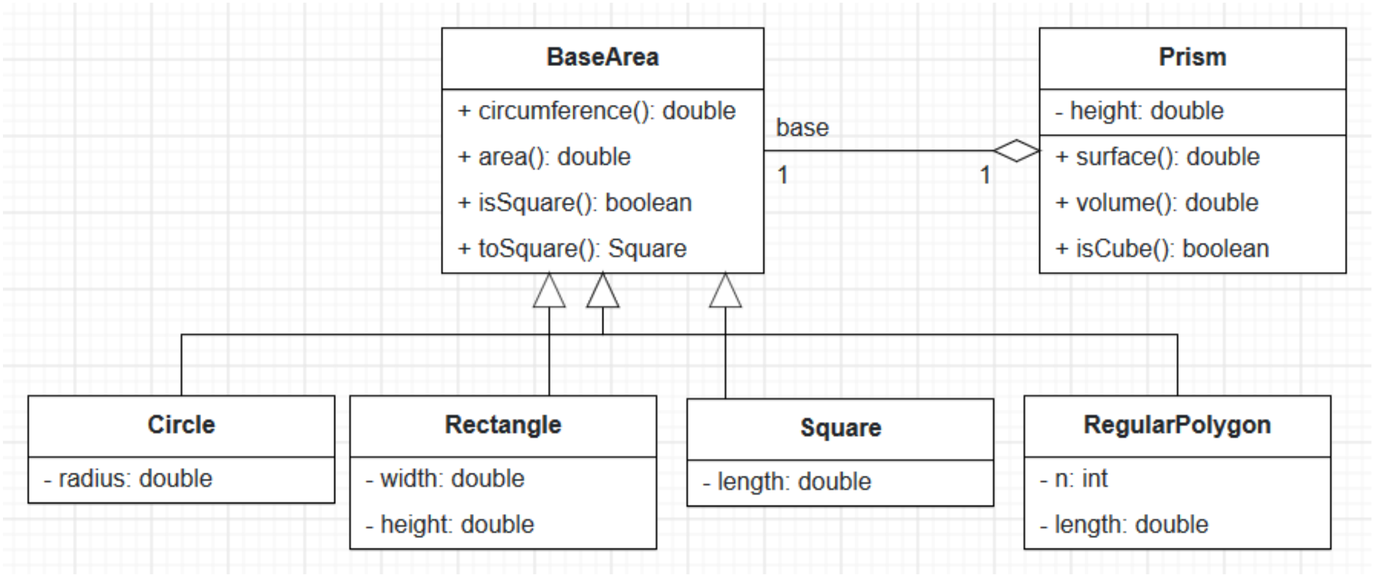
Idee: Stack zur Speicherung der aktuell geöffneten Klammern.

P07.02

Vererbung, Geometrische Formen

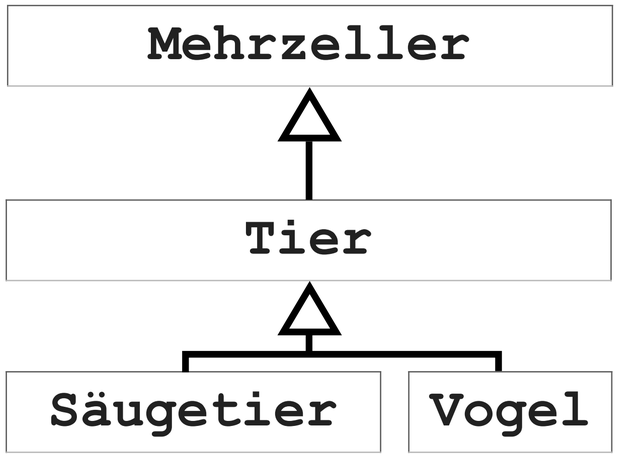
Fläche eines n-Eckes mit Seitenlänge a: $$\frac{n \cdot a^2}{4 \cdot \tan(\frac{\pi}{n})} = A_n$$

```
Math.PI; Math.tan(double angle);
```



P07.04

Terminkalender, Vererbungshierachien




```
public class M
```

Vogel verfügt auch über Attribute und Methoden aus Mehrzeller.

```
public class Tier extends
 Mehrzeller {}
public class Säugetier extends
 Tier {}
public class Vogel extends
 Tier {}
```

Mehrstufige hierarchische Verknüpfungen

P07.04

Terminkalender, Vererbungshierachien