



Parameter

openHPI-Java-Team
Hasso-Plattner-Institut



Datentyp Parameterbezeichner

```
1  void speak(String input){  
2      System.out.println(input);  
3  }
```

Parameter

- Variablen von Methoden, die beim Methodenaufruf befüllt werden
- Syntax:
 <Typ><Methodenbezeichner>(<Datentyp><Parameterbezeichner>)
- Zugriff innerhalb der Methode über den Parameterbezeichner



Aufruf von Methoden mit Parametern

```
1 void speak(String input){  
2     System.out.println(input);  
3 }  
4 void greet(){  
5     speak("Hallo");  
6 }
```

Parameter

Methodenname

Argument

- Syntax: <Methodenname>(<Argument>);
- Argumente werden beim Methodenaufruf übergeben
- **Parameter:** Bei der Methodendefinition
- **Argument:** Übergabewert beim Aufruf der Methode
- **Achtung!** Datentypen müssen übereinstimmen



Parameter

```
1 void speak(String input){  
2     System.out.println(input);  
3 }  
4 void greet(){  
5     String text = "Hallo Paco";  
6     speak(text);  
7 }
```

- Argument kann Wert oder eine andere Variable sein
- Variablenname (`text`) muss nicht gleich dem Parameternamen in der Methodendefinition (`input`) sein
- **Achtung!** Datentypen müssen übereinstimmen



Mehrere Parameter

```
1 double add(int a, int b, double c){  
2     return a+b+c;  
3 }  
4 void calculate(){  
5     System.out.println(add(3,4,1.1));  
6 }
```

Parameter

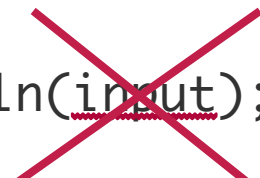
- Übergeben von mehreren Parametern möglich
 - Werden durch Kommata getrennt
- Können unterschiedliche Datentypen haben
- **Achtung!** Reihenfolge muss exakt gleich sein
- Anzahl der Argumente muss der Anzahl der Parameter entsprechen
- Leere Klammern: Methode nimmt keine Argumente



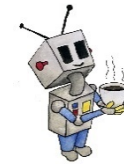
- Auch "Scope" genannt
- Schlüsselwörter **überall** verwendbar
- Selbstgewählte Bezeichner **nicht überall** verwendbar
- Parameter nur innerhalb von Methode bekannt
- Grundsätzlich gilt: geschweifte Klammern begrenzen einen Gültigkeitsbereich



```
1 void speak(String input){  
2     System.out.println(input);  
3 }  
4 void say(){  
5     System.out.println(input);  
6 }
```



- Parameter nur innerhalb der Methode definiert

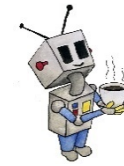


Gültigkeitsbereich: Parameter

```
1 void speak(String input){  
2                       
3     System.out.println(input);  
4 }  
5 void greet() {  
6     String input = "Hallo";  
7 }
```

- Parameter nur innerhalb der Methode `speak(...)` gültig
- `input` ist außerhalb von `speak(...)` nicht bekannt
- gleicher Bezeichner kann außerhalb der Methode (z.B. in `greet()`) an andere Variable vergeben werden

Gültigkeitsbereiche: ein verwirrendes Beispiel



```
1 class Name {  
2     void name(String name){  
3         name = name + "name";  
4         System.out.println(name);  
5     }  
6     public static void main(String[] argv){  
7         Name name = new Name();  
8         String namenname = "name";  
9         name.name(namenname);  
10    }  
11 }
```

- Valider Code (leider)
- Als Hilfe für Verständlichkeit individuelle Bezeichner nehmen