

JAVA Programmierung

ECLIPSE & JAVA 8

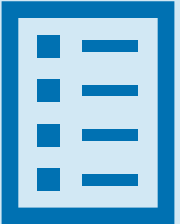


Themenübersicht

01

Konstruktoren

1. Generelles
2. Beispiele





01 Konstrukturen

1. *Generelles*
2. *Beispiele*



1.1

Konstrukturen

Generelles



1.1 Konstruktoren

• Generelles

Konstruktoren sind spezielle Methoden, die zum Erzeugen einer Instanz implizit oder explizit aufgerufen werden.

Der Konstruktor trägt den Namen der Klasse, und gibt keinen Rückgabebetyp an, da er offensichtlich eine Instanz der eigenen Klasse (erzeugt) zurückgibt.

```
public class Person{  
  
    private String vorName;  
    private String nachName;  
    private int alter;  
  
    //Konstruktor  
    public Person(String vorName, String nachName, int alter) {  
        this.vorName = vorName;  
        this.nachName = nachName;  
        this.alter = alter;  
    }  
}
```

1.1 Konstruktoren

- Generelles

Ein Konstruktor Aufruf erfolgt außerhalb der eigenen Klasse **immer** über das Schlüsselwort **new**, in Verbindung mit Klassennamen sowie den auch für Methoden üblichen Runden Klammern und eventuellen Parametern.

```
public class Person{  
  
    private String vorName;  
    private String nachName;  
    private int alter;  
  
    public Person(String vorName, String nachName, int alter) {  
        this.vorName = vorName;  
        this.nachName = nachName;  
        this.alter = alter;  
    }  
}
```

```
public class MyProgram {  
  
    public static void main(String[] args) {  
        //Eine Instanz mit Startzustand instanziiieren  
        Person p1 = new Person("Heinz", "Ulrich", 37);  
    }  
}
```


1.1 Konstruktoren

• Generelles

Innerhalb der eigenen Klasse ist es ebenso möglich den Konstruktor auszurufen. Hierfür wird allerdings, im Gegensatz zum Aufruf außerhalb der Klasse nicht der **new** Operator verwendet sondern **this()**. Hierbei gilt zu beachten das dies eine Weiterleitung an einen anderen Konstruktor darstellt. In unserem Beispiel wird der Konstruktor mit den Parametern aufgerufen.

Sollte kein Konstruktor definiert sein, erstellt der Compiler einen parameterlosen default Konstruktor. Dieser dient nur einem Zweck => Aufruf der Superklasse (Vaterklasse). **Hierzu später mehr!**

```
public class Person{  
  
    private String vorName;  
    private String nachName;  
    private int alter;  
  
    public Person(String vorName, String nachName, int alter) {  
        this.vorName = vorName;  
        this.nachName = nachName;  
        this.alter = alter;  
    }  
  
    public Person() {  
        this("Max", "Mustermann", 0);  
    }  
}
```



1.1 Konstruktoren

• Generelles

Da Konstruktoren dem gleichen Prinzip wie Methoden unterliegen, können diese auch Überladen werden. Hierfür ist eine unterschiedliche Anzahl an Parametern notwendig.

Bei gleicher Parameter Anzahl müssen sich die Typen der jeweiligen Parameter unterscheiden. Somit sich das Verhalten der Klasse sehr dynamisch auf etwaige unterschiedliche Anwendungsfälle anpassen.

```
public class Person{

    private String vorName;
    private String nachName;
    private int alter;

    //mit allen 3 Parametern
    public Person(String vorName, String nachName, int alter) {
        this.vorName = vorName;    //oder: setVorName(vorName);
        this.nachName = nachName;  //oder: setNachName(nachName);
        this.alter = alter;        //oder: setAlter(alter);
    }

    //Wenn nur Vor und Nachname vorhanden sind
    public Person(String vorName, String nachName) {
        this.vorName = vorName;
        this.nachName = nachName;
        this.alter = 0;
    }

    //Wenn man eine Dummy (den Max Mustermann) erstellen will
    public Person() {
        this.vorName = "Max";
        this.nachName = "Mustermann";
        this.alter = 0;
    }
}
```


1.1 Konstruktoren

• Generelles

Konstruktoren folgen einigen Regeln :

- Sie tragen immer den Namen der Klasse.
- Sie können überladen werden (mehrere Konstruktoren, s.o.).
- Sie deklarieren keinen Rückgabewert.
- Sollte innerhalb einer Klasse kein Konstruktor deklariert sein, wird immer ein „default “ Konstruktor vom Compiler erstellt => einen sogenannten „No args Konstruktor“.

1.2 Konstrukturen

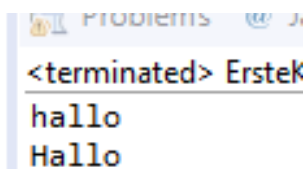
Beispiele



1.2 Konstruktoren

• Beispiele

```
4 public class ErsteKlasse {  
5     public static void main(String[] args) {  
6         Test test = new Test("hallo");  
7         // oder  
8         Test test2 = new Test();  
9     }  
10 }  
11  
12 class Test {  
13     private String x;  
14     public Test(String x) {  
15         this.x = x;  
16         System.out.println(this.x);  
17     }  
18     public Test() {  
19         this("Hallo");  
20     }  
21 }
```



Problems @ J
<terminated> ErsteK
hallo
Hallo

1.2 Konstruktoren

- Beispiele

```
}  
// die Konstruktoren des Baustein Projektes  
public Board(int kante) {  
    setKante(kante);  
    setSpielbrett();  
    fillBoardWithWater();  
}  
  
private Board() {  
    setKante(10);  
    setSpielbrett();  
    fillBoardWithWater();  
}
```



VIELEN DANK!

