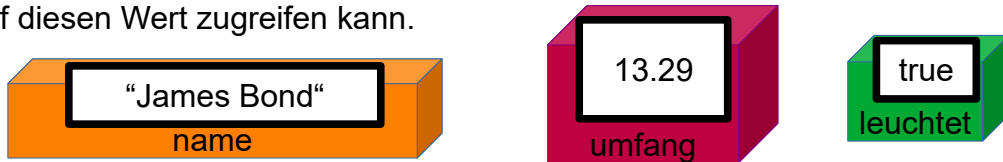
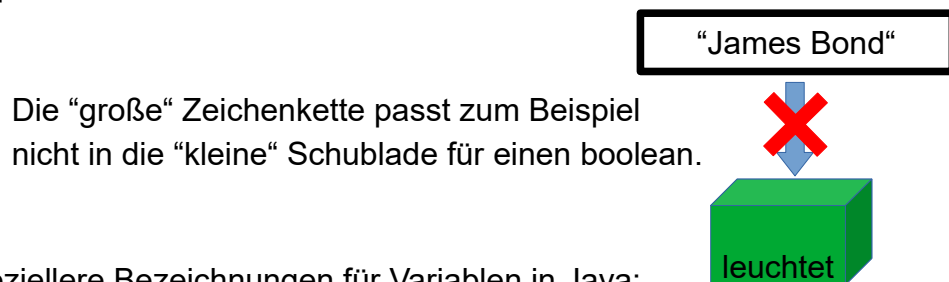


# Variablen in Java

**Variablen** sind (in Java) **Platzhalter** für einen Wert. Dabei bekommt jeder Platzhalter bzw. Variable einen **Datentyp** und einen **Namen** zugewiesen, damit man weiß, wie man wieder auf diesen Wert zugreifen kann.



Du kannst dir Variablen wie eine Art **Schublade oder Box** vorstellen, die durch einen Namen gekennzeichnet ist. In diese Schublade kann immer **nur genau ein Wert** (auf einem Zettel) gleichzeitig abgelegt werden. Die **Form und Größe** der Schublade würde dem Datentyp entsprechen.



Wir nutzen drei speziellere Bezeichnungen für Variablen in Java:

- **Attribute**
- **Übergabeparameter**
- **lokale Variablen**

Diese sind aber dennoch alle Variablen, die auch genau gleich funktionieren. Sie unterscheiden sich nur in ihrer "Lebensdauer". » [siehe Lebensdauer von Variablen am Ende des Skripts](#)

## Deklaration einer Variablen

Bevor eine Variable genutzt werden kann, muss diese erstmals **einmalig deklariert** (= **Festlegung des Datentyps**) werden. Das bedeutet, bevor ein Wert in die Schublade gelegt werden kann, muss die Schublade mit der entsprechenden Form und Größe (→ Datentyp) gebaut werden. **Dies funktioniert bei allen Variablen gleich!**

**Attribute** werden **außerhalb von Methoden** deklariert.

```
public class GEHEIMAGENT{  
    int agentennummer;  
    public GEHEIMAGENT(){
```

**Übergabeparameter** werden in der **Methodensignatur** deklariert.

```
public void kontaktiereAgent(int agentennummer){
```

**Lokale Variablen** können an einer beliebigen Stelle **innerhalb von Methoden** deklariert werden.

```
public void kontaktiereAgent(int agentennummer){  
    int telefonnummer;
```

## Initialisierung von Variablen


Damit die Variable benutzt werden kann, muss diese initialisiert werden. Die **Initialisierung** einer Variablen ist das **erstmalige Zuweisen eines Werts**. Bezogen auf das Modell wäre es der erste Zettel, der in der Schublade abgelegt wird, mit welchem man anschließend arbeiten kann. Vorher hat die Variable in Java den „Nullwert“ (alle Bits sind 0 → Interpretation je nach Datentyp). Die Initialisierung erfolgt bei den Variablentypen **unterschiedlich**:

**Attribute** werden im **Konstruktor** initialisiert.

```
int agentennummer;  
public GEHEIMAGENT(){  
    agentennummer = 47;
```

Durch den **Methodenaufruf** mit einem Wert für den Übergabeparameter wird der **Übergabeparameter** initialisiert.

```
Bond.kontaktiereAgent(47);
```



```
public void kontaktiereAgent(int agentennummer){
```

**Lokale Variablen** werden in der Regel **direkt nach** der Deklaration initialisiert.

```
public void kontaktiereAgent(int agentennummer){  
    int telefonnummer;  
    telefonnummer = 12345;
```


## Benutzung von Variablen

Sobald die Variablen deklariert und initialisiert sind, kann man mit ihnen bzw. mit ihren zugewiesenen Werten arbeiten. Die Art der Benutzung der Variablen ist auch abhängig vom Datentyp (→ mit Zahlen kann man rechnen und mit Zeichenketten nicht).

### Wertzuweisung von Variablen

Variablen kann ein neuer Wert mittels **=** zugewiesen werden.

Dies funktioniert bei allen „Variablentypen“ und Datentypen gleich.



```
String deckname;  
double erfolgsquote;  
public void methode(boolean istUndercover){  
    int telefonnummer;  
  
    deckname = "James Bond";  
    erfolgsquote = 0.99;  
    istUndercover = false;  
    telefonnummer = 12345;
```

## Rechnen mit Zahlenvariablen

Mit Zahlen (Datentyp **int** und **double**) kann gerechnet werden. Das **Ergebnis** der Rechnung kann **wieder einer Variablen zugewiesen** werden. Die Operatoren für das Rechnen sind **+**, **-**, **\***, **/**, **%**. **Modulo**-Rechnung (Divisionsrest) ist bei einer fünfte Grundrechenart der Informatik:  $17 \% 5 = 2$     $33 \% 3 = 0$     $9 \% 2 = 1$

```
ergebnis = 5-7;      ergebnis = v1*v2;      v1 = v1+v2;
```

Es wird immer zuerst die Rechnung auf der rechten Seite ausgeführt und **anschließend** der berechnete Wert der ausgeführte Variablen auf der linken Seite zugewiesen.

Mit Variablen muss häufig und viel gerechnet werden. Daher gibt es in Java Kurzschreibweisen, um schneller programmieren zu können.

Im folgenden Beispiel wird die Variable **int z** um den Wert **a** verändert:

	Standard	Kurzversion	Veränderung um genau 1
Addition	<code>z = z + a;</code>	<code>z += a;</code>	<code>z++;</code>
Subtraktion	<code>z = z - a;</code>	<code>z -= a;</code>	<code>z--;</code>
Multiplikation	<code>z = z * a;</code>	<code>z *= a;</code>	
Division	<code>z = z / a;</code>	<code>z /= a;</code>	
Modulo	<code>z = z % a;</code>	<code>z %= a;</code>	

## Ausgabe von Variablenwerten

Alle Variablen jedes Datentyps können mit dem Befehl **System.out.println( Variable );** auf der **Konsole** ausgegeben werden.

```
System.out.println(deckname);  
System.out.println(erfolgsquote);  
System.out.println(istUndercover);  
System.out.println(telefonnummer);
```



BlueJ: Konsole - Einführung  
Optionen  
James Bond  
0.99  
false  
12345

## Zeichenketten verbinden

Zeichenketten können mittels **+** verbunden werden.

```
vorname = "James";  
nachname = "Bond";  
vollername = vorname+" "+nachname;  
System.out.println(vollername);
```



BlueJ: Konsole - Einführung  
Optionen  
James Bond

# Lebensdauer von Variablen

Damit ein Computer nicht im laufenden Betrieb immer weiter zugemüllt wird bis kein Speicherplatz mehr vorhanden ist, müssen zwischendrin immer wieder nicht mehr benötigte Variablen oder Objekte gelöscht werden. Zum Beispiel beim Schließen eines Programms, können alle Objekte und Variablen des Programms wieder vom Arbeitsspeicher gelöscht werden. Um diesen Vorgang müssen wir uns in Java nicht selbst kümmern (in anderen Programmiersprachen schon). Der **Garbage Collector** in Java erkennt, wann Objekte und Variablen nicht mehr benötigt werden und löscht diese dann automatisch.

- **Attribute** sind die Eigenschaften eines Objekts. Solange dieses Objekt existiert muss auch das Attribut existieren. Sobald das Objekt gelöscht wird, dann wird auch Attribut mitgelöscht.
- Ein **Übergabeparameter** wird für die Wertübergabe bei einem Methodenaufruf genutzt. Somit kann dieser gelöscht werden, sobald die Methode fertig ausgeführt wurde.
- Eine **lokale Variable** wird immer dann gelöscht, wenn der Code-Block (z. B. Methodenrumpf oder Schleifenrumpf) in dem sie erstellt wurde fertig ausgeführt wurde.