



# Grunddatentypen, Ausdrücke und Variablen

## Typkonversion, Überprüfen und Auswerten von Ausdrücken

---

Annabelle Klarl

Zentralübung zur Vorlesung

„Einführung in die Informatik: Programmierung und Softwareentwicklung“

<http://www.pst.ifi.lmu.de/Lehre/wise-15-16/infoeinf>



Action required now



1. Smartphone: installiere die App "socrative student" **oder**  
Laptop: öffne im Browser [b.socrative.com/login/student](https://b.socrative.com/login/student)
2. Betrete den Raum **InfoEinf.**
3. Beantworte die erste Frage sofort!



## Grunddatentypen in Java

- Ganze Zahlen: **byte**, **short**, **int**, **long** mit `+`, `-`, `*`, `/`, `%`, `<`, `<=`, `>`, `>=`, `==`, `!=`  
z.B. 165
- Gleitpunktzahlen: **float**, **double** mit `+`, `-`, `*`, `/`, `%`, `<`, `<=`, `>`, `>=`, `==`, `!=`  
z.B. 1.65
- Zeichen: **char**  
z.B. 'A'
- Zeichenketten: `String` mit `+`  
z.B. "Annabelle"
- Wahrheitswerte : **boolean** mit `!`, `&&`, `&`, `||`, `|`  
z.B. `true` und `false`



## Grunddatentypen: Typkonversion (I)

= Werte eines Datentyps in einen anderen Datentyp umwandeln

### 1. Implizite oder **automatische Typkonversion** zum größeren Typ

`byte < short < int < long < float < double`

z.B. `165 - 1.5` ist automatisch vom Typ `double`



## Grunddatentypen: Typkonversion (II)

= Werte eines Datentyps in einen anderen Datentyp umwandeln

2. Explizite Typkonversion oder **Type Casting**:  
Erzwingen der Typkonversion durch Voranstellen von `(type)`

z.B. `(int) 1.65` erhält explizit den Typ `int`

Nachkommaanteil passt nicht  
in den Wertebereich des Datentyps `int`

=> Nachkommastellen werden abgeschnitten: Informationsverlust



## Aufgabe 1: Typkonversion (I)

Ein netter Bankangestellter verspricht Ihnen für Ihr Sparkonto einen Zinssatz von 25%. Er berechnet dabei folgendermaßen den Zins, den Sie bekommen werden:

```
double haben = 2000;  
double zins = haben * (1/4);
```



Was ist der Wert des  
Java-Ausdrucks  $2000 * (1/4)$  ?

Sie wollen natürlich sofort zuschlagen. Warum sollten Sie sich das **nochmal genauer überlegen** und dem Bankangestellten einen Gegenvorschlag machen?



## Aufgabe 1: Typkonversion (II)

Ein netter Bankangestellter verspricht Ihnen für Ihr Sparkonto einen Zinssatz von 25%. Er berechnet dabei folgendermaßen den Zins, den Sie bekommen werden:

```
double haben = 2000;  
double zins = haben * (1/4);
```

Vom Typ `int`,  
d.h. Nachkommastellen werden abgeschnitten:  $1/4$  ( $=0.25$ )  $=0$

Sie wollen natürlich sofort zuschlagen. Warum sollten Sie sich das **nochmal genauer überlegen** und dem Bankangestellten einen Gegenvorschlag machen?



## Aufgabe 1: Typkonversion (III)

Ein netter Bankangestellter verspricht Ihnen für Ihr Sparkonto einen Zinssatz von 25%. Er berechnet dabei folgendermaßen den Zins, den Sie bekommen werden:

```
double haben = 2000;  
double zins = haben * (1.0/4.0);
```

Vom Typ `double`,  
d.h.:  $1.0/4.0 = 0.25$

Sie wollen natürlich sofort zuschlagen. Warum sollten Sie sich das nochmal genauer überlegen und dem Bankangestellten einen **Gegenvorschlag** machen?





## Ausdrücke: Präzedenzen (I)

Woher wissen wir, wie man  $2 * 5 + 10$  berechnet?

- Gilt  $2 * 5 + 10 = 2 * (5 + 10)$  oder
- Gilt  $2 * 5 + 10 = (2 * 5) + 10$ ?

Die mathematischen Operatoren haben eine feste Reihenfolge, in der sie ausgewertet werden:

- Potenzrechnung vor Punktrechnung
- Punktrechnung vor Strichrechnung („Punkt vor Strich“)...

Auch in Programmiersprachen gibt es eine solche Reihenfolge, besser bekannt als **Präzedenz** (=Bindungsstärke) **eines Operators**.



## Ausdrücke: Präzedenzen (II)

Der Operator mit der höchsten Präzedenz wird zuerst ausgewertet.

Operation	Präzedenz
!, unäres +-	14
(type)	13
*, /, %	12
binäres +-	11
>, >=, <, <=	9
==, !=	8
&	7
	6
&&	4
	3

Beispiel:

- $5-4 < 3 == \text{false}$  **ist**  
 $((5-4) < 3) == \text{false}$



Was ist der Wert von  
`!false && false?`



## Ausdrücke: Präzedenzen (II)

Der Operator mit der höchsten Präzedenz wird zuerst ausgewertet.

Operation	Präzedenz
!, unäres +-	14
(type)	13
*, /, %	12
binäres +-	11
>, >=, <, <=	9
==, !=	8
&	7
	6
&&	4
	3

Beispiel:

- $5-4 < 3 == \text{false}$  **ist**  
 $((5-4) < 3) == \text{false}$
- `!false && false`
  - `ist(!false) && false = false`
  - **ist nicht** `!(false && false) = true`



## Ausdrücke: Überprüfen von Korrektheit

### Vorgehensweise:

1. Den Ausdruck von **links nach rechts** durchgehen und **vollständig klammern** unter Berücksichtigung von Präzedenzen.
2. Den Ausdruck nochmals von links nach rechts durchgehen und unter Berücksichtigung der Klammern überprüfen, ob
  - a. der Ausdruck **gemäß der Regel für Expression** gebildet ist (*syntaktische Korrektheit*).
  - b. die Argumenttypen von **Operationen** zu den Typen der Ausdrücke, auf die die Operationen angewendet werden, passen (*Typkorrektheit*).



## Aufgabe 2: Überprüfen von Korrektheit

```

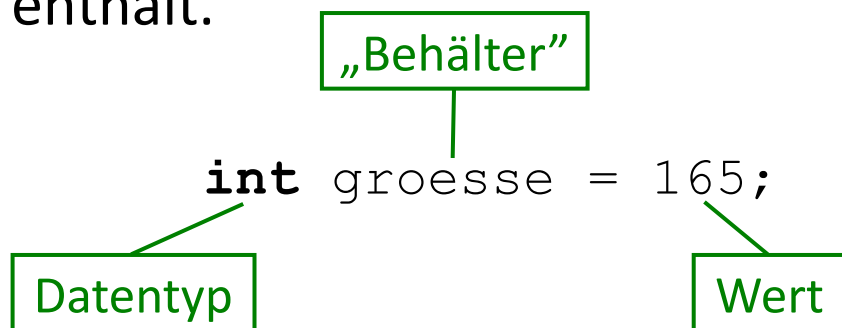
Expression = Variable | Value |
             Expression BinOp Expression |
             UnOp Expression |
             "(" Expression ")"
BinOp = "&" | "|" | "&&" | "||" | "+" | "-" | "*" | "/" |
        "%" | "==" | "!=" | ">" | ">=" | "<" | "<="
UnOp = "!" | "(" Type ")" | "-" | "+"
  
```

Ausdruck	Vollständig geklammert	Syn. K.	Typk.
<code>false == 5-4-3 &lt; 3</code>	<code>false == (((5-4)-3) &lt; 3)</code>	ja	ja
<code>7 &lt; false</code>	<code>7 &lt; false</code>	ja	nein, wg <
<code>3 &lt;&gt; 6</code>		nein	-



## Variablen in Java

Eine Variable ist ein „Behälter“, der zu jedem Zeitpunkt (während eines Programmlaufs) einen Wert eines bestimmten Datentyps enthält.



Zustand  $\sigma$  nach obiger Deklaration

**textuell**

$\sigma = [(groesse, 165)]$

**grafisch**

groesse 165



## Ausdrücke & Variablen: Auswertung

Vorgehensweise gegeben ein Ausdruck und ein Zustand  $\sigma$ :

1. Den Ausdruck von **links nach rechts** durchgehen und **vollständig klammern** unter Berücksichtigung von Präzedenzen.
2. Den Ausdruck nochmals von links nach rechts durchgehen und unter Berücksichtigung der Klammern **auswerten**. Der Wert der Variablen ist dabei durch den **Zustand  $\sigma$**  bestimmt.



## Aufgabe 3a: Auswertung

Gegeben seien folgende Variablendeklarationen:

```
double fahrenheit = 40;  
double celsius = 4.44;
```

Welcher Zustand  $\sigma$  wird durch diese Deklarationen beschrieben?





## Aufgabe 3a: Auswertung

Gegeben seien folgende Variablendeklarationen:

```
double fahrenheit = 40; //automatische Typkonversion  
double celsius = 4.44;
```

Welcher Zustand  $\sigma$  wird durch diese Deklarationen beschrieben?

**textuell**

$\sigma = [(fahrenheit, 40.0), (celsius, 4.44)]$

**grafisch**

celsius

4.44

fahrenheit

40.0

Stack  $\sigma$  wächst von unten nach oben



## Aufgabe 3b: Auswertung

Werten Sie den Ausdruck `fahrenheit - 32 * 5/9` bezüglich des Zustands  $\sigma = [(fahrenheit, 40.0), (celsius, 4.44)]$  aus:



Was ist der Wert von

`(fahrenheit - 32 * 5/9)?`



## Aufgabe 3b: Auswertung

Werten Sie den Ausdruck `fahrenheit - 32 * 5/9` bezüglich des Zustands  $\sigma = [(\text{fahrenheit}, 40.0), (\text{celsius}, 4.44)]$  aus:

### 1. Vollständig klammern:

```
fahrenheit - ((32 * 5) / 9)
```

### 2. Von links nach rechts auswerten:

```
fahrenheit - ((32 * 5) / 9) = σ
```

```
40.0 - ((32 * 5) / 9) = σ
```

```
40.0 - (160/9) = σ
```

```
40.0 - 17 = σ
```

```
23.0
```

Vom Typ `int`,  
d.h.: `160/9` (=17.78) =17

Automatische Typkonversion zu `double`,  
d.h.: `40.0-17` =`40.0-17.0` = 23.0



## Aufgabe 3c: Auswertung

Werten Sie den Ausdruck  $(\text{fahrenheit} - 32) * 5/9$  bezüglich des Zustands  $\sigma = [(\text{fahrenheit}, 40.0), (\text{celsius}, 4.44)]$  aus:



Was ist der Wert von

$(\text{fahrenheit} - 32) * 5/9?$



## Aufgabe 3c: Auswertung

Werten Sie den Ausdruck  $(\text{fahrenheit} - 32) * 5/9$  bezüglich des Zustands  $\sigma = [(\text{fahrenheit}, 40.0), (\text{celsius}, 4.44)]$  aus:

### 1. Vollständig klammern:

```
((fahrenheit - 32) * 5) / 9
```

### 2. Von links nach rechts auswerten:

```
((fahrenheit - 32) * 5) / 9 = 6
```

```
((40.0 - 32) * 5) / 9 = 6
```

```
(8.0 * 5) / 9 = 6
```

```
40.0 / 9 = 6
```

```
4.4444...
```

Automatische Typkonversion zu `double`,