

# Operatoren

# Operatoren



# Arithmetische Operatoren

## Grundrechenarten

- + Addition
- Subtraktion
- \* Multiplikation
- / Division

## Modulo-Operator

- % bestimmt den Rest

Involviert sind immer zwei Operanden.

Es gelten die selben Regeln wie in der Mathematik:

- Punktrechnung vor Strichrechnung
- Durch Klammersetzung ( ) wird der Ausdruck in den Klammern zuerst ausgewertet

# Arithmetische Operatoren

## Beispiele

|  |                      |                                |
|--|----------------------|--------------------------------|
| <code>int resultat = 3 + 4 * 5;</code>   | <code>// = 23</code> | Punkt vor Strich               |
| <code>int resultat = (3 + 4) * 5;</code> | <code>// = 35</code> | Klammer zuerst                 |
| <code>int resultat = 14 / 5;</code>      | <code>// = 2</code>  | Kommastellen entfallen bei int |
| <code>int resultat = 14 % 5;</code>      | <code>// = 4</code>  | $14 / 5 = 2$ Rest 4            |

# Arithmetische Operatoren: Verkürzt geschrieben

Da oft der Wert einer Variable mit **sich selbst berechnet** wird, gibt es dafür eine verkürzte Schreibweise mit folgenden Operatoren:

- `+=` Addition mit sich selbst
- `-=` Subtraktion mit sich selbst
- `/=` Division mit sich selbst
- `*=` Multiplikation mit sich selbst
- `%=` Rest von Division mit sich selbst

## In gut Deutsch

Ein Wert wird mit sich selber berechnet

## In Java

```
int i = 15; // Startwert von i ist 15
int k = 3;  // Startwert von k ist 3
i += 5;     // i = i + 5; → i = 20
i -= 7;     // i = i - 7; → i = 16
i /= k;     // i = i / k; → i = 5
i *= 7;     // i = i * 7; → i = 35
i %= 4;     // i = i % 4; → i = 3
```

# Unäre Operatoren: `++` *und* `--`

- besitzen kein Gleichheitszeichen
- beziehen sich **auf sich selbst**
- also **nur den "linken" Teil** einer Operation
- werden ausschliesslich zum Hoch- und Runterzählen verwendet
  - `++` Inkrementierung
  - `--` Dekrementierung

## In gut Deutsch

In Eineschritten sich selbst Hoch- oder Runterzählen

## In Java

```
int i = 3;           // Startwert für i ist 3
double d = 1.5;      // Startwert für d ist 1.5

i++; // i = i + 1    → i = 4
d--; // d = d - 1.0  → d = 0.5
```

# Vergleichsoperatoren

- Mit Vergleichsoperatoren lassen sich Ausdrücke formulieren, welche einen Rückgabewert vom Typ `boolean` liefern.
- Eine Variable vom Typ `boolean` kann die Werte `true` oder `false` annehmen.
- Durch Vergleichsoperatoren können Wahrheitswerte ermittelt werden
- Diese können für Entscheidungen im Programmverlauf verwendet werden.
  - `if`, `while`
- Alle Vergleichsoperatoren:
  - `==`, `!=`, `<`, `>`, `<=`, `>=`

# Vergleichsoperatoren: *Tabelle*

Gegeben: `int a = 2; int b = 3;`

| Operator           | Beschreibung                                  | Beispiel               | Resultat           |
|--------------------|---|------------------------|--------------------|
| <code>==</code>    | überprüft auf <u>Gleichheit</u>               | <code>a == b</code>    | <code>false</code> |
| <code>!=</code>    | überprüft auf <u>Ungleichheit</u>             | <code>a != b</code>    | <code>true</code>  |
| <code>&gt;</code>  | ist linker Operand <u>grösser</u>             | <code>a &gt; b</code>  | <code>false</code> |
| <code>&lt;=</code> | ist linker Operand <u>grösser oder gleich</u> | <code>a &lt;= b</code> | <code>true</code>  |
| <code>&lt;</code>  | ist linker Operand <u>kleiner</u>             | <code>a &lt; b</code>  | <code>true</code>  |
| <code>&gt;=</code> | ist linker Operand <u>kleiner oder gleich</u> | <code>a &gt;= b</code> | <code>false</code> |

\* nur bei primitiven Datentypen. Nicht bei `String` !



# Logische Operatoren: *Kombination von bool'shen Ausdrücken*

Gegeben: `boolean a = true; boolean b = false;`

| Operator                | Beschreibung  | Beispiel                    | Resultat           |
|-------------------------|---|-----------------------------|--------------------|
| <code>&amp;&amp;</code> | UND: <b>beide</b> Ausdrücke sind <code>true</code>          | <code>a &amp;&amp; b</code> | <code>false</code> |
| <code>  </code>         | ODER: <b>mindistens ein</b> Ausdruck ist <code>true</code>  | <code>a    b</code>         | <code>true</code>  |
| <code>^</code>          | XOR: <b>genau einer</b> der Ausdrücke ist <code>true</code> | <code>a ^ b</code>          | <code>true</code>  |
| <code>!</code>          | NOT: wandelt ein <code>boolean</code> ins Gegenteil um      | <code>!b</code>             | <code>true</code>  |