



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Programmierkurs Java

Operatoren, Ausdrücke
und Anweisungen

Raphael Allner
Institut für Telematik
22. Oktober 2019

1. Operatoren

- Variablen befüllen mit dem Zuweisungsoperator
- Arithmetische, Vergleichs-, logische und Bitweise Operatoren

2. Ausdrücke und Anweisungen

- Satzteile und ganze Sätze
- Blockanweisungen

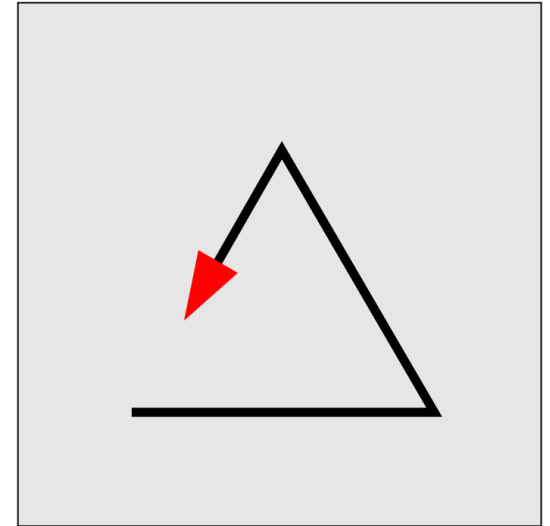
3. Ein- und Ausgaben

- `System.out`
- `System.in`

Beispiel




```
1  double laenge = 4;  
2  int volleUmdrehung = 360  
3  int drittel = volleUmdrehung / 3;  
4  boolean fertig = false;  
5  
6  BeginneZeichnen();  
7  BewegeVorwärts(laenge / 3);  
8  DreheNachLinks(drittel);  
9  BewegeVorwärts(laenge / 3);  
10 DreheNachLinks(drittel);  
11 BewegeVorwärts((laenge / 3) / 2);  
12 DreheNachLinks(drittel);  
13 BeendeZeichnen();  
14  
15 fertig = true;
```



Übungs- oder Hausaufgabe



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

1. Laden Sie die Datei **OperatorenUndAusdruecke.java** aus dem Moodle herunter
2. Verfolgen Sie die Vorlesung um die Aufgaben zu erfüllen
3. Achten Sie auf die Ausrufezeichen auf den Folien
 - 
 - Hier sind Informationen zur Aufgabenerfüllung zu finden
4. Überprüfen Sie Ihren Code zwischen den Aufgaben
 1. Starten Sie ein Terminal / Kommandozeilentool
 2. Kompilieren mit **javac OperatorenUndAusdruecke.java**
 3. Zum Ausführen in der JVM: **java OperatorenUndAusdruecke**

Operatoren, Ausdrücke und Anweisungen

Operatoren

Was können wir mit Variablen und deren Werten anfangen?

- Werte **zuweisen**
- Werte **vergleichen**
- Werte **addieren**
- Werte **subtrahieren**
- usw.

Dazu werden **Operatoren** eingesetzt

- Siehe auch:
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>

Operatoren

Ausdrücke



Ein **Ausdruck** (engl. expression) besteht aus:

- **Operanden** (Variablen, Literale, oder Methoden) und
- **Operatoren** (verknüpfen Operanden, berechnen neuen Wert)
- Liefert **einen Wert** eines **bestimmten Datentyps**
要确定一个数据类型

Beispiele:

1	1	// Wert: 1,	Typ: int
2	42 * -18	// Wert: -756,	Typ: int
3	1 + 2.0	// Wert: 3.0,	Typ: double
4	3.098 * 12.7 + 8 / 7.0	// Wert: 40.48...,	Typ: double
5	(char)('a' + 7)	// Wert: 'h',	Typ: char
6	(4.0 / 3) * Math.PI * Math.pow(radius, 3);		
7		// Wert: 57.90...,	Typ: double
8	a + b	// Wert: a + b	Typ: ?

Ausdrücke und Anweisungen

Ausdrücke, Beispiel



```
1 public class Ausdruecke {  
2     public static void main(String[] args) {  
3         System.out.println(1);  
4         System.out.println(42 * -18);  
5         System.out.println(1 + 2.0);  
6         System.out.println(3.098 * 12.7 + 8 / 7.0);  
7         System.out.println((char)('a' + 7));  
8         System.out.println((4.0 / 3) * Math.PI * Math.pow(2.4, 3));  
9     }  
10 }
```

A screenshot of a Windows PowerShell terminal window. The prompt is 'PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke>'. The user enters 'javac .\Ausdruecke.java'. The prompt changes to 'PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke>'. The user enters 'java Ausdruecke'. The output is displayed on the next lines: '1', '-756', '3.0', '40.48745714285714', 'h', '57.90583579096705'. The prompt returns to 'PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke>'.

```
Windows PowerShell  
PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> javac .\Ausdruecke  
.java  
PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> java Ausdruecke  
1  
-756  
3.0  
40.48745714285714  
h  
57.90583579096705  
PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> _
```


Operatoren

Übersicht



Bezeichnung	Syntax
Postfix	expr++ expr--
Unär	++expr --expr +expr -expr ~ !
Multiplikativ	* / %
Additiv	+ -
Shift	<< >> >>>
Vergleich	< > <= >= instanceof
Gleichheit	== !=

Bezeichnung	Syntax
Bitweises AND	&
Bitweises XOR	^
Bitweises OR	
Logisches AND	&&
Logisches OR	
Ternärer Operator	? :
Zuweisung	= += -= *= /= %= &= ^= = <<= >>= >>>=

sortiert nach Vorrang

Operatoren

Übersicht



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

算法	增量	配置	对比	逻辑	位元
Arithmetik	Inkrement Dekrement	Zuweisung	Vergleich	Logik	Bitweise
-a a + b a - b a * b a / b a % b a & b	++a --a a++ a--	a = b a += b a -= b a *= b a /= b a %= b a &= b	a == b a != b a > b a >= b a < b a <= b	!a a && b a b	~a a & b a b a ^ b a << b a >> b a >>> b

Operatoren

Zahlenarithmetik



Arithmetische Operatoren geben einen Zahlenwert zurück

- + (Addition)
- (Subtraktion)
- * (Multiplikation)
- / (Division)
- % (Modulo-Operator)

Syntax: $\langle \text{WertA} \rangle + \langle \text{WertB} \rangle$

Semantik: Gibt das Ergebnis der Addition von $\langle \text{WertA} \rangle$ & $\langle \text{WertB} \rangle$ zurück

Beispiel:

- `System.out.println(3 + 9.5);` // 12.5, double
- `System.out.println(10 % 8);` // 2, int

Operatoren

Zahlenarithmetik – durch null



Schon die Schulmathematik sagt:

„Teile niemals durch 0!“

Schüler teilt durch 0.

In Java:

- **Ganzzahl**division: Arithmetic Exception -> Ende des Programmablaufs
- **Fließkommazahl**division: Ergebnis +/- unendlich
- $0.0/0.0$ = Sonderwert Not a Number (NaN)

Operatoren

Inkrement und Dekrement



Inkrement und Dekrement-Operatoren

- Geben jeweils einen Zahlenwert zurück
- Nicht frei von **Nebeneffekten** 次要因素

		Entspricht	Bezeichnung	Erklärung	Beispiel in Java-Code
Inkrement	<code>i++</code>	<code>i = i + 1;</code>	Postinkrement	Nachherige Werterhöhung.	<code>int i = 5;</code> <code>int c = i++; // c = 5</code>
	<code>++i</code>	<code>i = i + 1;</code>	Präinkrement	Vorherige Werterhöhung	<code>int i = 5;</code> <code>int d = ++i; // d = 6</code>
Dekrement	<code>i--</code>	<code>i = i - 1;</code>	Postdekrement	Nachherige Wertverkleinerung.	<code>int i = 5;</code> <code>int e = i--; // e = 5</code>
	<code>--i</code>	<code>i = i - 1;</code>	Prädekrement	Vorherige Wertverkleinerung	<code>int i = 5;</code> <code>int f = --i; // f = 4</code>



Java ist statisch typisiert:

Das **Ergebnis einer Berechnung** hat immer **einen bestimmten Datentyp** 计算结果只有一个数据类型

Beispiele:

1 + 1

- Ergebnis 2
- Datentyp **int**

1.0 + 1

- Ergebnis 2
- Datentyp **double**

Wir erinnern uns:

Auch Literale sind immer von einem bestimmten Datentyp

Weitere Beispiele:

```
int i = 2 * 1;
```

- Zwei Operatoren (= und *)
- Ergebnis der Multiplikation: 2, Datentyp **int**;

2.0 * 1; Ergebnis: 2.0, Datentyp **double**

3 / 2; Ergebnis: 1, Datentyp **int**

3 / 2.0; Ergebnis: 1.5, Datentyp **double**



Operatoren

Zuweisungsoperator (=)



Der **Zuweisungsoperator** = weist einer Variable einen Wert zu

Syntax: **<Variable>** = <WertDesAusdrucks>;

Semantik: (Ergebnis-)Wert der rechten Seite wird der linken Seite zugewiesen. <WertDesAusdrucks> wird **vor** der Zuweisung berechnet / ausgewertet.

Beispiele:

```
1 int myInt = 10;  
2 myInt = 2 + 9;           // 11)  
3 myInt = 12 * myInt / 2;  // 12 * 11 / 2 = 66)
```

- Wird die Variable links und rechts des Zuweisungsoperators verwendet, so hat sie rechts noch den „alten“ Wert
等号左右的变量和数据应是同一类型

Operatoren

Zuweisungsoperator (=)



Der **Zuweisungsoperator** = weist einer Variable einen Wert zu

Syntax: **<Variable>** = <WertDesAusdrucks>;

Semantik: (Ergebnis-)Wert der rechten Seite wird der linken Seite zugewiesen. <WertDesAusdrucks> wird **vor** der Zuweisung berechnet / ausgewertet.

Was wird hier ausgegeben?

```
1 double d = 10 / 3;  
2 System.out.println(d);
```

Ausgabe: 3.0

- Da Literale vom Datentyp **int** sind
- Richtig wäre: **double** d = 10.0 / 3.0.; bzw. **double** d = 10. / 3.;
- **Ergebnis bleibt ungenau:** 3.33333333333333335 → Siehe **BigDecimal**

Operatoren

Zuweisungsoperator (=): Beispiele



```
1 public class Zuweisung {  
2     public static void main(String[] args) {  
3         char    c = 'f';  
4         byte    b = 10;  
5         short   s = 20;  
6         int     i = 30;  
7         long    l = 111111111;  
8         float   f = 10.001f;  
9         double  d = 10.001;  
10        boolean bo = false;  
11  
12        bo = true;  
13        bo = i;  
14        f = i;  
15        i = f;  
16    }  
17 }
```

Terminal

```
File Edit View Search Terminal Help  
$javac Zuweisung.java  
Zuweisung.java:13: error: incompatible types: int cannot be converted to  
boolean  
        bo = i;  
        ^  
Zuweisung.java:15: error: incompatible types: possible lossy conversion  
from float to int  
        i = f;  
        ^  
2 errors  
$
```

Explizite Typumwandlung (Cast)

- Java handhabt Datentypen sehr strikt
- Typkonvertierungen müssen explizit angegeben werden

设置一个变量，如果数据不属于这个类型，在这之前加一个括号再加上数据

Syntax: **(<Datentyp>)**<Wert/Variable>

Semantik: Gib <Wert/Variable> im angegebenen Datentyp **<Datentyp>** zurück

Falsch:

```
1 int i = 3.7 * 6;           // Rechter Ausdruck: Typ double
2 int i = (int) 3.7 * 6;    // Ungenaueres Ergebnis: 3 * 6 = 18
```

Richtig:

```
1 int i = (int)(3.7 * 6);    // Ergebnis: 3.7 * 6 = 22.2 → 22
```

Operatoren

Typumwandlung



Man unterscheidet zwischen:

- 拓展 ■ **Widening** (Erweitern): Konvertierung von Variablen von einem Typ mit **kleinerem** auf einen mit **größerem Wertebereich**
数据类型的转化从小值域到大值域
- 缩小 ■ **Narrowing** (Eingrenzen): Konvertierung von Variablen von einem Typ mit **größerem** auf einen mit **kleinerem Wertebereich**
数据类型的转化从大值域到小值域
- Gefahr von Datenverlust bzw. Umwandlungsfehlern

→		byte	short	int	long	float	double
Ursprünglicher Datentyp	byte		w	w	w	w	w
	short	n		w	w	w	w
	int	n	n		w	w	w
	long	n	n	n		w	w
	float	n	n	n	n		w
	double	n	n	n	n	n	

Operatoren

Zuweisungsoperator (=)



Der Zuweisungsoperator kann mit **Arithmetik**, **logischen** und **bitweisen** Operatoren kombiniert werden

- Spart Schreibarbeit, jedoch ggf. unübersichtlicher

Operator	Bezeichnung	Beispiel	Entspricht	
+=	Additionszuweisung	a += b	a = a + b	Arithmetik
-=	Subtraktionszuweisung	a -= b	a = a - b	
*=	Multiplikationszuweisung	a *= b	a = a * b	
/=	Divisionszuweisung	a /= b	a = a / b	
%=	Modulozuweisung	a %= b	a = a % b	
&=	Und-Zuweisung	a &= b	a = a & b	Logik
=	Oder-Zuweisung	a = b	a = a b	
^=	Exklusiv-Oder-Zuweisung	a ^= b	a = a ^ b	
<<=	Linksschiebe-Zuweisung	a <<= b	a = a << b	Bitweise
>>=	Rechtsschiebe-Zuweisung	a >>= b	a = a >> b	
>>>=	Rechtsschiebe-Zuweisung ohne Vorzeichen	a >>>= b	a = a >>> b	

Operatoren

Zahlenvergleiche



Vergleich-Operatoren geben **boolean** Wert zurück 比较符号在Boolean中的应用

- `==` (gleich)
- `!=` (ungleich)
- `>` (größer)
- `>=` (größer oder gleich)
- `<` (kleiner)
- `<=` (kleiner oder gleich)

Syntax: `<WertA> > <WertB>`

Semantik: Gib **true** zurück, wenn `<WertA>` **echtgrößer** als `<WertB>` ist, sonst **false**.

Beispiel:

```
1 int i = 0;  
2 int j = 10;  
3 System.out.println( i == j );  
4 System.out.println( j < 10 );
```

Ausgabe:
false
false



Operatoren

Logik - Boolesche Ausdrücke



Logik-Operatoren werten Boolesche-Ausdrücke zu **boolean**-Werten aus

&& (und-Verknüpfung)

逻辑符号在Boolean中的应用

|| (oder-Verknüpfung)

! (Negation)

Syntax: <BooAusdruckA> && <BooAusdruckB>

Semantik: Gibt **true** zurück, wenn <BooAusdruckA> **und** <BooAusdruckB> wahr sind, sonst **false**.

Beispiel:

```
1 System.out.println(  
2   ((false && true) || !(1 == 2))  
3 );
```

(false&&true) 是空集
!(1==2) 是true
true || 空集 是true

Ausgabe:
true



Operatoren

Bitweise Operatoren



Bitweise-Operatoren für Operationen auf Bit-Ebene eines Wertes

- & (logisches UND auf Bit-Ebene)
- | (logisches ODER auf Bit-Ebene)
- ^ (logisches EXKLUSIV-ODER auf Bit-Ebene)
- ~ (negieren alle Bits)
- >> (Rechts-Schiebe-Operator)
- >>> (Erweiterter Rechts-Schiebe-Operator)
- << (Links-Schiebe-Operator)

Syntax: `<Ergebnis> = <WertA> & <WertB>`

Semantik: **Auf Bit-Ebene** - Wenn ein Bit von <WertA> **und** sein Gegenstück bei <WertB> 1 sind, wird der Bit 1 im <Ergebnis>, sonst 0.

Beispiele: `0001 & 0010 = 0000;`
`0001 & 0011 = 0001;`

Operatoren, Ausdrücke und Anweisungen

Anweisungen

Ausdrücke und Anweisungen

Anweisungen (Statements)



Anweisungen

- ... sind einzelne Schritte, die das Programm ausführen soll
- Mehrere Anweisungen werden durch Semikolons ; getrennt (sequentielle Ausführung)

Anweisungen **müssen keinen Wert zurückgeben**

- **Ausdrücke** hingegen **schon**
- Jeder Ausdruck ist auch eine Anweisung
 - Wert des Ausdrucks wird ggf. ignoriert

Beispiel:

```
1 System.out.println("Hallo");
```

Ausgabe:
Hallo

Gibt Hallo auf der Konsole aus, berechnet aber keinen Wert

Ausdrücke und Anweisungen

Anweisungen (Statements)



Spezielle Anweisung: **Blockanweisung**

- **Gruppiert Menge von Anweisungen** zu einer Anweisung

Syntax: `{ <keineOderMehrereAnweisungen> }`

Beispiele:

```
3  {}  
4  { int i = 1+2; System.out.println("Hallo") }
```

Inhalte eines Blocks werden zur besseren Lesbarkeit eingerückt:

```
5  {  
6      int i = 1 + 2;  
7      System.out.println("Hallo");  
8      i = i * 2;  
9  }
```

Operatoren, Ausdrücke und Anweisungen

Ein- und Ausgaben

Ein- und Ausgaben

Ausgeben von Zeichen



Ausgabe auf der Konsole 输出语句

- Zwei Ausgabe-„Ströme“: **Standard** (`System.out`) und **Fehler** (`System.err`)
- Statische Objekte `out` und `err` von der Klasse **PrintStream**
- Trennen von „echten“ Ausgabedaten und Fehlern
- Normal nicht unterscheidbar, können einzeln „umgeleitet“ werden

Zeilenumbrüche bei Standardausgaben: 标准输出

```
5 System.out.println("Hallo Welt"); //„Hallo Welt“ und Zeilenumbruch
6 System.out.print("Hallo Welt");  //„Hallo Welt“ (ohne Z.-Umbruch)
7 System.out.println();            //Beginnt nur eine neue Zeile
```

Fehlerausgabe: 报错输出

```
8 System.err.println("Das sollte nicht passieren");
```

Ein- und Ausgaben

Beispiel: System.out und System.err



```
1 public class SysOutErr {
2     public static void main(String[] args) {
3         System.out.println("Hallo Welt 1");
4         System.err.println("Das sollte nicht passieren");
5         System.out.print("Hallo Welt 2");
6         System.err.println("Und das auch nicht");
7         System.out.println();
8         System.err.println("Und das erst recht nicht");
9     }
10 }
```

Windows PowerShell

```
PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> javac .\SysOutErr.java
PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> java SysOutErr
Hallo Welt 1
Das sollte nicht passieren
Hallo Welt 2Und das auch nicht

Und das erst recht nicht
PS C:\dev\Git\Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> _
```

Ein- und Ausgaben

Exkurs: Einlesen von Zeichen



Passendes Gegenstück: `System.in` 输入语句

- Zeichen von der Tastatur einlesen

Beispiel: Ein Zeichen von Tastatur lesen

- `char c = (char)System.in.read();`
- Warum der Typ-Cast?
- Woher weiß man das?

Umfangreiche Dokumentation von Java

- Sogenannte Javadoc
- <https://docs.oracle.com/javase/10/docs/api/overview-summary.html>

Ein- und Ausgaben

Beispiel



```
1 public class Eingabe {  
2     public static void main(String[] args) throws java.io.IOException {  
3         System.out.println("Drücken Sie eine beliebige Taste (gefolgt von <Enter>):");  
4         char taste = (char)System.in.read();  
5         System.out.println("Sie haben die folgende Taste gedrückt: " + taste);  
6     }  
7 }
```

```
Windows PowerShell  
PS C:\dev\Git\_Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> javac .\Eingabe.java  
PS C:\dev\Git\_Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> java Eingabe  
Drücken Sie eine beliebige Taste (gefolgt von <Enter>):  
süßeKatzenbabys  
Sie haben die folgende Taste gedrückt: s  
PS C:\dev\Git\_Lehre\java-kurs\Folien\Java-Quellcode\Beispiele\3-Operatoren_und_Ausdruecke> _
```

Drücken Sie eine beliebige Taste (gefolgt von <Enter>):
süßeKatzenbabys
Sie haben die folgende Taste gedrückt: s

Übungs- oder Hausaufgabe

Nachbesprechung



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Siehe Lösungsvorschlag im Moodle

Ausgabe:

```
true  
istGleich und istKleiner sind true
```



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Kontakt

Raphael Allner, M. Sc.
Wissenschaftlicher Mitarbeiter
Institut für Telematik

Universität zu Lübeck
Ratzeburger Allee 160
23562 Lübeck

<https://www.itm.uni-luebeck.de/mitarbeitende/raphael-allner.html>

