

8. Ausdrücke, Operatoren (einfache Typen)

Teil 2

Java-Beispiel:
Unicode.java

Typumwandlung Cast-Operator

Unterschiedliche Typen

Zuweisung

```
int a;
short b;

a = b; ✓
b = a;
```

Ausdruck

```
float y;
double x;

x = x + y; ✓
```

Parameter Deklaration:

```
public static int fakultaet (int n) { ... }
```

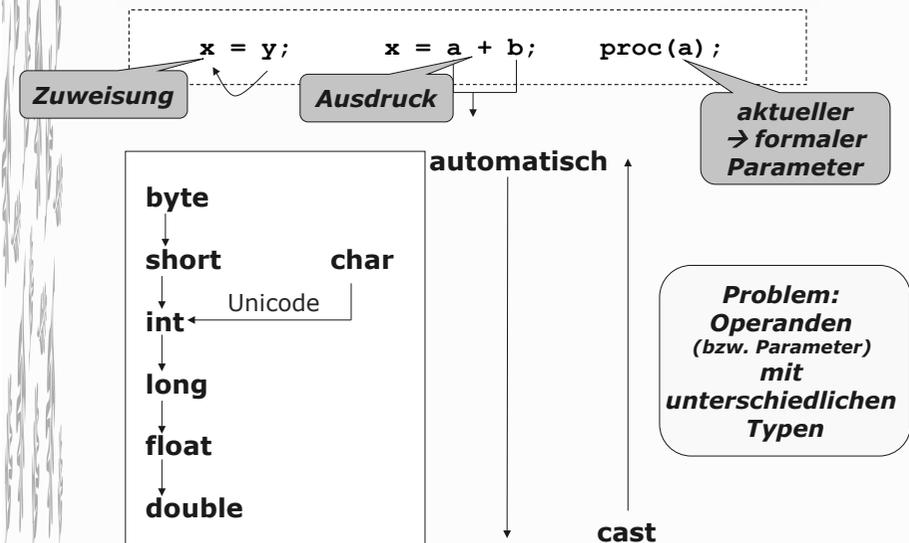
Aufruf: int x; short a; long b;

```
x = fakultaet (a); ✓
x = fakultaet (b);
```

Was ist erlaubt?

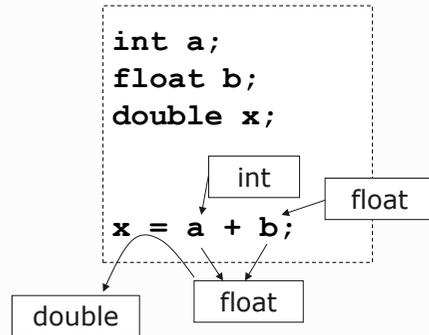
Wo muss man nachhelfen?

Typumwandlung: Übersicht



Automatische Typumwandlung

Automatische Umwandlung in den größeren Typ:
Kein Informationsverlust

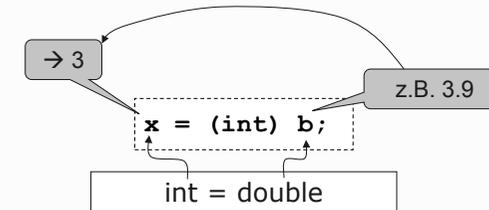


Explizite Typumwandlung: Type-Cast-Operator

Explizite Umwandlung vom größeren in den
kleineren Typ:
Eventuell Informationsverlust

(type) Ausdruck

Wandelt Ausdruck in einen Ausdruck vom Typ type um



Ausgabe der Unicodezeichen 0020 - 00FF im Console-Fenster (ASCII, LATIN 1)

```
// Windows: Ausgabe von Unicode-Zeichen im Console-Fenster erfordert die Codepage 1252.
// Erfolgt durch das Kommando: 'chcp 1252' im DOS-Fenster.
// Zusätzlich im DOS-Fenster die Schriftart 'Lucida Console' wählen.

import java.awt.*;

class Unicode {

    public static void main (String [] args) {

        for (char code = '\u0020'; code <= '\u00FF'; code = (char) (code + 1)) {
            String fill = "";
            if (code < '\u0064')
                fill = " "; // zweistellige Dezimalzahlen rechtsbueendig

            if (code >= '\u007F' & code <= '\u009F')
                // '?' fuer nichtdruckbares Zeichen
                System.out.print
                    (fill + Integer.toString(code) + " " + '?' + " ");
            else
                System.out.print
                    (fill + Integer.toString(code) + " " + Character.toString(code) + " ");

            if (code%10 == 0)
                System.out.println();
        }
        System.out.println();
    }
}
```

Unicode.java

Ausgabe des Programms

```
fill
32 33 ! 34 " 35 # 36 $ 37 % 38 & 39 ' 40 (
41 ) 42 * 43 + 44 , 45 - 46 . 47 / 48 0 49 1 50 2
51 3 52 4 53 5 54 6 55 7 56 8 57 9 58 : 59 ; 60 <
61 = 62 > 63 ? 64 @ 65 A 66 B 67 C 68 D 69 E 70 F
71 G 72 H 73 I 74 J 75 K 76 L 77 M 78 N 79 O 80 P
81 Q 82 R 83 S 84 T 85 U 86 V 87 W 88 X 89 Y 90 Z
91 [ 92 \ 93 ] 94 ^ 95 _ 96 ` 97 a 98 b 99 c 100 d
101 e 102 f 103 g 104 h 105 i 106 j 107 k 108 l 109 m 110 n
111 o 112 p 113 q 114 r 115 s 116 t 117 u 118 v 119 w 120 x
121 y 122 z 123 { 124 | 125 } 126 ~ 127 ? 128 ? 129 ? 130 ?
131 ? 132 ? 133 ? 134 ? 135 ? 136 ? 137 ? 138 ? 139 ? 140 ?
141 ? 142 ? 143 ? 144 ? 145 ? 146 ? 147 ? 148 ? 149 ? 150 ?
151 ? 152 ? 153 ? 154 ? 155 ? 156 ? 157 ? 158 ? 159 ? 160 ?
161 i 162 ¢ 163 £ 164 ¤ 165 ¥ 166 ¦ 167 § 168 ¨ 169 © 170 ª
171 « 172 ¬ 173 ® 174 ¯ 175 ¯ 176 ¯ 177 ± 178 º 179 » 180 ¯
181 µ 182 ¶ 183 · 184 ¸ 185 ¹ 186 º 187 » 188 ¼ 189 ½ 190 ¾
191 ¿ 192 À 193 Á 194 Â 195 Ã 196 Ä 197 Å 198 Æ 199 Ç 200 È
201 É 202 Ê 203 Ë 204 Ì 205 Í 206 Î 207 Ï 208 Ð 209 Ñ 210 Ò
211 Ó 212 Ô 213 Õ 214 Ö 215 × 216 Ø 217 Ù 218 Ú 219 Û 220 Ü
221 Ý 222 Þ 223 ß 224 à 225 á 226 â 227 ã 228 ä 229 å 230 æ
231 ç 232 è 233 é 234 ê 235 ë 236 ì 237 í 238 î 239 ï 240 ð
241 ñ 242 ò 243 ó 244 ô 245 õ 246 ö 247 ÷ 248 ø 249 ù 250 ú
251 û 252 ü 253 ý 254 þ 255 ÿ
```

? nicht druckbar

ASCII-Code

0000 – 001F:
nicht druckbar

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Unicode: Basic Latin (ASCII) und Latin-1

Zu lesen:

- Alles hexadezimal

- z.B. Spalte 004, Zeile 1

→ Gesamt 2 Byte 0041

→ 'A' repräsentiert

Gedruckter
Bereich

C0 Controls and Basic Latin								C1 Controls and Latin-1 Supplement							
000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
0	NUX	DLE	SP	0	@	P	p	0	XXX	DCS	SP	À	Ð	à	ð
1	SOH	DC1	!	1	A	Q	a	1	XXX	PU1	±	Á	Ñ	á	ñ
2	STX	DC2	"	2	B	R	b	2	BPH	PU2	²	Â	Ò	â	ò
3	ETX	DC3	#	3	C	S	s	3	NBH	STS	£	³	Ã	Ó	ã
4	DOT	DC4	\$	4	D	T	t	4	IND	DCH	¤	´	Ä	Ö	ä
5	ENQ	NAK	%	5	E	U	e	5	NEL	MW	¥	µ	Å	Ø	å
6	ACK	SYN	&	6	F	V	f	6	ESAP	SPA	¦	¶	Æ	Ö	æ
7	BEL	ETB	'	7	G	W	w	7	ESAL	EPA	§	·	Ç	×	ç
8	BS	CAN	(8	H	X	x	8	HTS	SOS	¨	¸	È	Ø	è
9	HT	EM)	9	I	Y	y	9	HTX	XXX	©	¹	É	Ù	é
A	LF	SUB	*	:	J	Z	z	A	VTS	SCI	ª	º	Ê	Ú	ê
B	VT	BSC	+	;	K	[{	B	PLD	CSI	«	»	Ë	Û	ë
C	FF	PS	<	L	\			C	PLU	ST	¼	½	Ì	Ü	ì
D	CR	RS	=	M]	m	}	D	RI	OSC	¾	⅓	Í	Ý	í
E	SO	RS	.	>	N	^	~	E	SS2	PM	¾	¼	Î	Þ	î
F	SI	US	/	?	O	_	oj	F	SS3	APC	¾	¼	Ï	ß	ï

Unicode: Basic Latin (ASCII) und Latin-1

Zu lesen:

- Alles hexadezimal

- z.B. Spalte 004, Zeile 1

→ Gesamt 2 Byte 0041

→ 'A' repräsentiert

0000 – 001F:
nicht druckbar

007F – 009F:
nicht druckbar

C0 Controls and Basic Latin								C1 Controls and Latin-1 Supplement							
000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
0	NUX	DLE	SP	0	@	P	p	0	XXX	DCS	SP	À	Ð	à	ð
1	SOH	DC1	!	1	A	Q	a	1	XXX	PU1	±	Á	Ñ	á	ñ
2	STX	DC2	"	2	B	R	b	2	BPH	PU2	²	Â	Ò	â	ò
3	ETX	DC3	#	3	C	S	s	3	NBH	STS	£	³	Ã	Ó	ã
4	DOT	DC4	\$	4	D	T	t	4	IND	DCH	¤	´	Ä	Ö	ä
5	ENQ	NAK	%	5	E	U	e	5	NEL	MW	¥	µ	Å	Ø	å
6	ACK	SYN	&	6	F	V	f	6	ESAP	SPA	¦	¶	Æ	Ö	æ
7	BEL	ETB	'	7	G	W	w	7	ESAL	EPA	§	·	Ç	×	ç
8	BS	CAN	(8	H	X	x	8	HTS	SOS	¨	¸	È	Ø	è
9	HT	EM)	9	I	Y	y	9	HTX	XXX	©	¹	É	Ù	é
A	LF	SUB	*	:	J	Z	z	A	VTS	SCI	ª	º	Ê	Ú	ê
B	VT	BSC	+	;	K	[{	B	PLD	CSI	«	»	Ë	Û	ë
C	FF	PS	<	L	\			C	PLU	ST	¼	½	Ì	Ü	ì
D	CR	RS	=	M]	m	}	D	RI	OSC	¾	⅓	Í	Ý	í
E	SO	RS	.	>	N	^	~	E	SS2	PM	¾	¼	Î	Þ	î
F	SI	US	/	?	O	_	oj	F	SS3	APC	¾	¼	Ï	ß	ï

Wo Typumwandlungen – automatisch oder explizit?

```
public static void main (String [] args) {
    for (char code = '\u0020' ; code <= '\u00FF' ; code = (char)(code + 1)) {
        String fill = "";
        if (code < '\u0064')
            fill = " ";
        // zweistellige Dezimalzahlen rechtsbueendig

        if (code >= '\u007F' && code <= '\u009F')
            // '?' fuer nichtdruckbares Zeichen
            System.out.print
                (fill + Integer.toString(code) + " " + '?' + " ");
        else
            System.out.print
                (fill + Integer.toString(code) + " " + Character.toString(code) + " ");

        if (code%10 == 0)
            System.out.println();
    }
    System.out.println();
}
```

0000 .. 001F:
Nicht-druckbare
Steuerzeichen

identisch:
char a = 'A';
char a = '\u0041';

Integer.toString

Explizite Umwandlung in Typ String
(in den Typ String wird nicht automatisch
und auch nicht mit "cast" umgewandelt)

`Integer.toString(code)`

API:

`java.lang`
`class Integer`

`static String toString(int i)`
Returns a String object
representing the specified integer

z.B. `Integer.toString(42)` → "42"
`Character.toString('a')` → "a"

Ausgabe der Unicodezeichen 0020 - 00FF im Console-Fenster

```
for (char code = '\u0020' ; code <= '\u00FF' ;
     Zweifache Typumwandlung code = (char)(code + 1)) {
    .....
    if (code >= '\u007F' && code <= '\u009F')
        // '?' fuer nichtdruckbares Zeichen
        System.out.print(fill + Integer.toString(code)
                        + " " + '?' + " ");
    else
        Zahl als Zeichenkette
        System.out.print(fill + Parameterübergabe: Integer.toString(code)
                        + " " + Zeichen als Zahl: Character.toString(code) + " ");

        Zeichen als Zeichenkette
    if (code%10 == 0)
        System.out.println();
}

char a = 'A';
char a = '\u0041';
```

Operatorenübersicht:

Vorrangregeln
Operandentyp

Operator	Typisierung	Assoziativität	Bezeichnung
Gruppe 1			
++	N	R	Inkrement
--	N	R	Dekrement
+	N	R	Unäres Plus
-	N	R	Unäres Minus
~	I	R	Einerkomplement
!	L	R	Negation
(type)	A	R	Type-Cast
Gruppe 2			
*	N,N	L	Multiplikation
/	N,N	L	Division
%	N,N	L	Modulo
Gruppe 3			
+	N,N	L	Addition
-	N,N	L	Subtraktion
+	S,S	L	Stringverkettung
Gruppe 4			
<<	I,I	L	Linksschieben
>>	I,I	L	Rechtsschieben
>>>	I,I	L	Rechtsschieben mit Nulllexpansion

Typisierung

Typisierung: **N, I, L, S, R, P, A**
 Numerisch, Integral (ganzzahlig),
 Logisch, String, Referenz, Primitiv,
 Alle Typen

→ Typen der Operanden

linke Seite einer Zuweisung: Variable **V**

Operator	Typisierung	Assoziativität	Bezeichnung
Typen der Operanden			
Gruppe 5			
	N,N	L	Kleiner
<=	N,N	L	Kleiner gleich
>	N,N	L	Größer
>=	N,N	L	Größer gleich
instanceof	R,R	L	Klassenzugehörigkeit
Gruppe 6			
==	P,P	L	Gleich
!=	P,P	L	Ungleich
==	R,R	L	Referenzgleichheit
!=	R,R	L	Referenzungleichheit
Gruppe 7			
&	I,I	L	Bitw. UND
&	L,L	L	Log. UND
Gruppe 8			
^	I,I	L	Bitw. XOR
^	L,L	L	Log. XOR
Gruppe 9			
	I,I	L	Bitw. ODER
	L,L	L	Log. ODER
Gruppe 10			
&&	L,L	L	Log. UND, Short-Cut

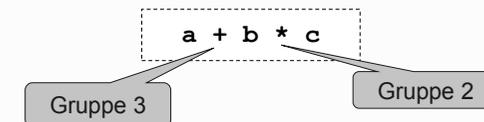
Vorrang (Priorität, Bindungsstärke)

Operator	Typisierung	Assoziativität	Bezeichnung
Gruppe 11			
	L,L	L	Log. ODER, Short-Cut
Gruppe 12			
? :	L,A,A	R	Bedingte Auswertung
Gruppe 13			
=	V,A	R	Zuweisung
+=	V,N	R	Additionszuweisung
-=	V,N	R	Subtraktionszuweisung
*=	V,N	R	Multiplikationszuweisung
/=	V,N	R	Divisionszuweisung
%=	V,N	R	Restwertzuweisung
&=	V,N	R	Bitw.-UND-Zuweisung
	V,L	R	Log.-UND-Zuweisung
=	V,N	R	Bitw.-ODER-Zuweisung
	V,L	R	Log.-ODER-Zuweisung
^=	V,N	R	Bitw.-XOR-Zuweisung
	V,L	R	Log.-XOR-Zuweisung
<<=	V,I	R	Linksschiebezuweisung
>>=	V,I	R	Rechtsschiebezuweisung
>>>=	V,I	R	Rechtsschiebezuweisung mit Nullexpansion

Vorrangregeln (1):

Vorrang = Priorität = Bindungsstärke

- Tabelle: niedrigere Gruppe mit höchster Priorität



→ also: $a + (b * c)$

Vergleiche: (int) 3.1 + 3.9 und (int) (3.1 + 3.9)

Operator	Typisierung	Assoziativität	Bezeichnung
Gruppe 1			
++	N	R	Inkrement
--	N	R	Dekrement
+	N	R	Unäres Plus
-	N	R	Unäres Minus
~	I	R	Einerkomplement
!	L	R	Negation
(type)	A	R	Type-Cast
Gruppe 2			
*	N,N	L	Multiplikation
/	N,N	L	Division
%	N,N	L	Modulo
Gruppe 3			
+	N,N	L	Addition
-	N,N	L	Subtraktion
+	S,S	L	Stringverkettung
Gruppe 4			
<<	I,I	L	Linksschieben
>>	I,I	L	Rechtsschieben
>>>	I,I	L	Rechtsschieben mit Nullexpansion

Vorrangregeln (2):

- Für eine einzelne Operation: Vorrang nach Assoziativität

- Beispiele für Linksassoziativität L
(Zusammenfassung von links)

`a - b - c` wie `(a - b) - c`
`a - (b - c)` wäre falsch

`a == b == c` wie `(a == b) == c`
true/false

Typ von a, b, c:
int möglich ? `if (a == b == 10) ... ?`

Vorrangregeln (3):

- Für eine einzelne Operation: Vorrang nach Assoziativität

- Beispiel für Rechtsassoziativität R
(Zusammenfassung von rechts)

`a = b = c` wie `a = (b = c)`

danach: a, b mit dem Wert von c

vermeiden !!

Aufgabe: Überprüfen

```
jahr % 4 == 0
&& jahr % 100 != 0
|| jahr % 400 == 0
```

bzw.

```
jahr % 4 == 0 && jahr % 100 != 0 || jahr % 400 == 0
```

dasselbe wie ...

```
(((jahr % 4) == 0)
&& ((jahr % 100) != 0))
|| ((jahr % 400) == 0)
```