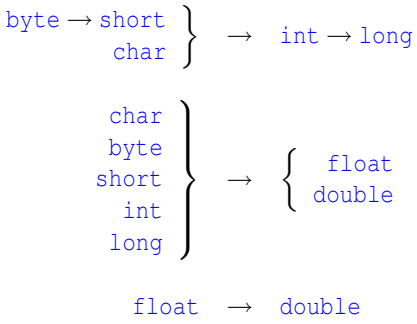
Javakurs SS03

„widening “



Primitive Datentypen

Typ	Bits	Werte
Ganze Zahlen		
byte	8*	-128 bis 127
short	16*	-32768 bis 32767
int	32	-2^{31} bis $2^{31}-1$
long	64	-2^{63} bis $2^{63}-1$
Fließkommazahlen (IEEE 754 single/double precision floating points)		
float	32	ca. $-3,4*10^{38}$ bis ca. $+3,4*10^{38}$ Betragsmin. $\neq 0$ ca. $\pm 1,4*10^{-45}$
double	64	ca. $-3,4*10^{38}$ bis ca. $+3,4*10^{38}$ Betragsmin. $\neq 0$ ca. $\pm 4,9*10^{-324}$
Logikwerte		
boolean	1*	true oder false
Buchstaben (Unicode)		
char	16	\u0000 bis \uFFFF

* Logische Größe des Datentyps.

Operatoren

Symbol	Operandentyp	Assoz.	Priorität	Name
`++`	arithm.	R	1	Prä-/Postinkrement (unär)
`--`	arithm.	R	1	Prä-/Postdekrement (unär)
∼	integral	R	1	bitweises Komplement (unär)
`!`	boolean	R	1	Not (unär)
`(`*type*`)`	bel.	R	1	Cast
`*`	arithm.	L	2	Multiplikation
`/`	arithm.	L	2	Division
`%`	arithm.	L	2	Modulo
`+`	arithm.	R	2	pos. Vorzeichen (unär)
`-`	arithm.	R	2	neg. Vorzeichen (unär)
`+`	arithm.	L	3	Addition
`-`	arithm.	L	3	Subtraktion
`+`	Strings	L	3	Stringverkettung
`<<`	integral	L	4	Linksshift
`>>`	integral	L	4	arithm. Rechtsshift
`>>>`	integral	L	4	logischer Rechtsshift
`<`	arithm.	L	5	kleiner
`>`	arithm.	L	5	größer
`>=`	arithm.	L	5	größergleich
`instanceof`	Objekt, Typ	L	5	Typüberprüfung
`==`	bel.	L	6	gleich
`!=`	bel.	L	6	ungleich
`&`	integral	L	7	bitweises Und
`&`	boolean	L	7	boolsches Und
∧	integral	L	8	bitweises XOR
∧	booleanl	L	8	boolsches XOR
`|`	integral	L	9	bitweises Oder
`|`	boolean	L	9	boolsches Oder
`&&`	boolean	L	10	bedingtes Und
`||`	boolean	L	11	bedingtes Oder
`? :`	bool., bel., bel.	R	12	Ternärer Auswahloperator
`=`	Var., bel.	R	13	Zuweisung
`*=`, `/=`, `%=`, `+=`, `-=`, `>>=`, `>>=`, `>>>=`, `&=`, ∧`=`, `|=`	Var, s.o.	R	13	Zuweisung mit Operation

Literale

- Integer
  - dezimal, Bsp.: `23`, `-15`
  - oktal, beginnt mit `0`, Bsp.: `023`, `-017`
  - hexadezimal, beginnt mit `0x`, Bsp.: `0x17`, `-0xF`
- Long: wie Integer, aber mit angehängtem `l`/`L`, Bsp: `23L`
- Double: Dezimalpunkt, Exponent `e`/`E` oder angehängtes `d`/`D`, Bsp: `1.39e-47`
- Float: angehängtes `f`/`F`, Dezimalpunkt/Exponent optional, Bsp: `1.39e-47f`
- Char: Zeichen in einfachen Anführungszeichen
  - druckbare Zeichen, Bsp.: `'K'`
  - Escapesequenzen

    `'\b'` backspace
    `'\t'` (horizontaler) Tabulator
    `'\n'` Zeilenvorschub (newline)
    `'\f'` Seitenvorschub (form feed)
    `'\r'` Wagenrücklauf (carriage return)
    `'\''` Anführungszeichen
    `'\"'` doppeltes Anführungszeichen
    `'\\'` backslash
  - Oktalcodes `'\000'` bis `'\377'`
  - Hexcode `'\u0000'` bis `'\uFFFF'`
- Boolean: `true`, `false`
- String: Zeichenfolge in doppelten Anführungszeichen
- Objektreferenz: `null`

Schlüsselwörter und vordef. Bezeichner

`abstract`	`default`	`goto`*	`operator`*	`synchronized`
`boolean`	`do`	`if`	`outer`*	`this`
`break`	`double`	`implements`	`package`	`throw`
`byte`	`else`	`import`	`private`	`throws`
`byvalue`*	`extends`	`inner`*	`protected`	`transient`
`case`	`false`	`instanceof`	`public`	`true`
`cast`*	`final`	`int`	`rest`*	`try`
`catch`	`finally`	`interface`	`return`	`var`*
`char`	`float`	`long`	`short`	`void`
`class`	`for`	`native`	`static`	`volatile`
`const`*	`future`*	`new`	`super`	`while`
`continue`	`generic`*	`null`	`switch`	