

# 5.1 Basistypen

## Primitive Datentypen

- ▶ Zu jedem Basistypen gibt es eine Menge möglicher **Werte**.
- ▶ Jeder Wert eines Basistyps benötigt den gleichen **Platz**, um ihn im Rechner zu repräsentieren.
- ▶ Der Platz wird in **Bit** gemessen.

Wie viele Werte kann man mit  $n$  Bit darstellen?

# Primitive Datentypen – Ganze Zahlen

Es gibt **vier** Sorten ganzer Zahlen:

<i>Typ</i>	<i>Platz</i>	<i>kleinster Wert</i>	<i>größter Wert</i>
byte	8	-128	127
short	16	-32 768	32 767
int	32	-2 147 483 648	2 147 483 647
long	64	-9 223 372 036 854 775 808	9 223 372 036 854 775 807

Die Benutzung kleinerer Typen wie **byte** oder **short** spart Platz.

# Primitive Datentypen – Ganze Zahlen

## Literale:

- ▶ dezimale Notation
- ▶ hexadezimale Notation (Präfix `0x` oder `0X`)
- ▶ oktale Notation (Präfix `0`)
- ▶ binäre Notation (Präfix `0b` oder `0B`)
- ▶ Suffix `l` ☹️ oder `L` für `long`
- ▶ `'_'` um Ziffern zu gruppieren

## Beispiele

- ▶ `192, 0b11000000, 0xC0, 0300` sind alle gleich
- ▶ `20_000L, 0xABFF_0078L`
- ▶ `09, 0x_FF` sind ungültig

# Primitive Datentypen – Ganze Zahlen

## Literale:

- ▶ dezimale Notation
- ▶ hexadezimale Notation (Präfix `0x` oder `0X`)
- ▶ oktale Notation (Präfix `0`)
- ▶ binäre Notation (Präfix `0b` oder `0B`)
- ▶ Suffix `l` ☹️ oder `L` für `long`
- ▶ `'_'` um Ziffern zu gruppieren

## Beispiele

- ▶ `192`, `0b11000000`, `0xC0`, `0300` sind alle gleich
- ▶ `20_000L`, `0xABFF_0078L`
- ▶ `09`, `0xFF` sind ungültig

# Primitive Datentypen – Ganze Zahlen

**Achtung:** Java warnt nicht vor Überlauf/Unterlauf!!!

**Beispiel:**

```
1 int x = 2147483647; // groesstes int
2 x = x + 1;
3 write(x);
```

liefert: -2147483648

# Primitive Datentypen – Gleitkommazahlen

Es gibt **zwei** Sorten von Gleitkommazahlen:

<i>Typ</i>	<i>Platz</i>	<i>kleinster Wert</i>	<i>größter Wert</i>	<i>signifikante Stellen</i>
float	32	ca. $-3.4 \cdot 10^{38}$	ca. $3.4 \cdot 10^{38}$	ca. 7
double	64	ca. $-1.7 \cdot 10^{308}$	ca. $1.7 \cdot 10^{308}$	ca. 15

$$x = s \cdot m \cdot 2^e \quad \text{mit } 1 \leq m < 2$$

- ▶ Vorzeichen  $s$ : 1 bit
- ▶ reduzierte Mantisse  $m - 1$ : 23 bit (float), 52 bit (double)
- ▶ Exponent  $e$ : 8 bit (float), 11 bit (double)

# Primitive Datentypen – Gleitkommazahlen

## Literale:

- ▶ dezimale Notation.
- ▶ dezimale Exponentialschreibweise  
(e, E für Exponent)
- ▶ hexadezimale Exponentialschreibweise. (Präfix 0x oder 0X,  
p oder P für Exponent)
- ▶ Suffix f oder F für float, Suffix d oder D für double  
(default is double)

## Beispiele

- ▶ 640.5F == 0x50.1p3f
- ▶ 3.1415 == 314.15E-2
- ▶ 0x1e3\_dp0, 1e3d
- ▶ 0x1e3d, 1e3\_d, 0x50.1

# Primitive Datentypen – Gleitkommazahlen

- ▶ Überlauf/Unterlauf bei Berechnungen liefert **Infinity**, bzw. **-Infinity**
- ▶ Division Null durch Null, Wurzel aus einer negativen Zahl etc. liefert **NaN**

## Weitere Basistypen

<i>Typ</i>	<i>Platz</i>	<i>Werte</i>
boolean	1	true, false
char	16	alle(?) Unicode-Zeichen

Unicode ist ein Zeichensatz, der alle irgendwo auf der Welt gängigen Alphabete umfasst, also zum Beispiel:

- ▶ die Zeichen unserer Tastatur (inklusive Umlaute);
- ▶ die chinesischen Schriftzeichen;
- ▶ die ägyptischen Hieroglyphen ...

### Literale:

- ▶ char-Literale schreibt man in Hochkommas: 'A', '\u00ED', ';', '\n'.
- ▶ boolean-Literale sind true und false.