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:

Typ	*Platz*	*kleinster Wert*	*größter Wert*
`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

Verwenden Sie niemals `l` als `long`-Suffix, da dieses leicht mit `1` verwechselt werden kann.

_ darf nur **zwischen** Ziffern stehen, d.h. weder am Anfang noch am Ende.

Übung:
Geben Sie eine reguläre Grammatik an, die diese Regeln abbildet

Beispiele

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

- In realem Java kann man bei der Deklaration einer Variablen ihr direkt einen ersten Wert zuweisen (Initialisierung).
- Man kann sie sogar (statt am Anfang des Programms) erst an der Stelle deklarieren, an der man sie braucht!

Primitive Datentypen – Gleitkommazahlen

Es gibt zwei Sorten von Gleitkommazahlen:

Typ	*Platz*	*kleinster Wert*	*größter Wert*	*signifikante Stellen*
`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

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

- 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)

Mantisse und Exponent sind dezimal; Basis für Exponent ist 10;

- hexadezimale Exponentialschreibweise. (Präfix `0x` oder `0X`, `p` oder `P` für Exponent)

Mantisse ist hexadezimal; Exponent ist dezimal und muß vorhanden sein; Basis für Exponent ist 2;

- Suffix `f` oder `F` für `float`, Suffix `d` oder `D` für `double` (default is `double`)

In der hexadezimalen Notation, gibt der Exponent die Anzahl der Bitpositionen an, um die das Komma verschoben wird.

Wenn der Exponent in der hexadeximalen Notation, hexadezimal wäre, wüßten wir nicht, ob ein finales 'f' zum Exponenten gehört, oder ein `float`-Suffix sein soll.

`0x1e3d` ist ein `int` und keine Gleitkommazahl

`1e3_d` ist ungültig, da '_' nicht zwischen 2 Ziffern steht (`d` ist keine Ziffer sondern das `double`-Suffix)

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

Typ	*Platz*	*Werte*
`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 Hochkomma: `'A'`, `'\u00ED'`,`';'`, `'\n'`.
- `boolean`-Literale sind `true` und `false`.

Die ursprüngliche Idee war, dass `char` alle Unicodezeichen enthält. Nach der Einführung von Java, hat sich der Unicodestandard geändert. Deshalb kann ein `char` nur Zeichen der sogenannten Basic Multilingual Plane speichern. Andere Unicodezeichen werden über Strings codiert.