Wert-Datentypen (Grunddatentypen, elementare Datentypen)

Mit Wertdatentypen kann ein Computer besonders effizient umgehen, weil sie seiner internen Registerstruktur angepaßt sind. Wertdatentypen werden nicht (wie Objekte) über Referenzen angesprochen, sondern die Speicherzellen enthalten direkt den binär codierten Wert. Die üblichen Java-Datentypen entsprechen weitgehend den elementaren Datentypen in C und C++, nämlich Ganzzahlen, Gleitkommazahlen, Buchstaben. Allerdings sind der jeweilige Wertebereich und die Speichergröße anders festgelegt und gelten wegen der Plattformunabhängigkeit von Java für alle Laufzeitumgebungen und Betriebssysteme.

Obwohl ein Wertdatentyp mit einer bestimmten Speicherstelle innerhalb der Java Virtual Machine verknüpft ist kann die Speicheradresse nicht ausgegeben werden, weil es in Java keinen Adress-Operator gibt, der dafür nötig wäre. Intern kennt die JVM die Speicherstellen und verfügt aber über alle nötigen Informationen zum Speicherbedarf und zur Art des dort gespeicherten Wertes.

Variablen und Konstanten eines Wertdatentyps werden wie in C durch die Nennung des Datentyps und des Namens erzeugt. Der Zusatz **`final`** kennzeichnet einen Datentyp als konstant (unveränderlich). Eine Konstante darf im Unterschied zu C und C++ nachträglich initialisiert werden – allerdings natürlich nur ein einziges Mal.

```
int variable;                   // Variable
final int konstante;            // Konstante

variable  = 2;          // ok
konstante = 3;          // ok
konstante = 4;          // Compilerfehler!
```

Art:	Typ:	Speicher:	Wertebereich:	.
boolean	Wahrheitswert	1 Byte	true	false
char	Buchstabe	2 Byte		
byte	Ganzzahl	1 Byte	-128	+127
short	Ganzzahl	2 Byte	-2^{15}	$+2^{15}-1$
int	Ganzzahl	4 Byte	-2^{31}	$+2^{31}-1$
long	Ganzzahl	8 Byte	-2^{63}	$+2^{63}-1$
float	Gleitkomma	4 Byte	$-1.4*10^{-45}$	$+3.4*10^{38}$
double	Gleitkomma	8 Byte	$-4.9*10^{-324}$	$+1.8*10^{308}$

Der Datentyp **boolean** kann nur den Wert **true** oder **false** annehmen. Eine Zuweisung von Zahlenwerten (wie in C) ist nicht gestattet. Der Wert von true und false ist eine Konstante, die in Java fest eingebaut ist und nicht durch einen Zahlenwert ersetzt werden darf. Bei allen Anweisungen oder Operationen, die als Ergebnis einen Wahrheitswert haben (`if(…)`, `for(…)`, `while(…)`), ist zu beachten, dass das Ergebnis nicht auf eine Zahl (wie in C: 0 oder 1) geprüft werden kann, sondern nur auf **false** oder **true**.

```
if ( (a < b) == true)...
if ( (a >= b) == false)...
```

Der Datentyp **char** ist immer vom Typ Unicode und kann somit alle internationalen Buchstaben (griechisch, türkisch ...) darstellen. Achtung: dies gilt nicht für die Bildschirmausgabe in konsolenbasierten Programmen.

Der Datentyp **byte** wurde für kleine Zahlenwerte, die wenig Speicher beanspruchen, eingeführt.

Die anderen Zahlentypen (**int**, **double**, **float**...) entsprechen den von C/C++ bekannten Zahlentypen.

In Java sind alle Zahlentypen immer vorzeichenbehaftet, umfassen also den negativen und positiven Zahlenbereich. Es gibt keine vorzeichenlose Datentypen und daher auch keinen Modifizierer wie in C / C++ (**unsigned**).

Hüllklassen für Wertdatentypen

Es wurde bereits darauf hingewiesen, dass die Übergabe eines Wert-Datentyps an eine Methode automatisch **call-by-value** erfolgt und vom Programm nicht beeinflußt werden kann. Wenn ein Wertdatentyp unbedingt per-reference übergeben werden muss, dann kann eine der speziellen Hüllklassen („Wrapper“) verwendet werden, die ihn in einen Objektdatentyp verwandeln. Allerdings sind diese Objekte nach erfolgter Zuweisung nicht veränderbar, eignen sich also nur bedingt zur Übergabe an Methoden.

Hüllklassen gibt es für jeden der genannten Zahlentypen:

```
byte b    = 1;
short s   = 2;
int i     = 3;
long l    = 4;
float f   = 5.1f;
double d1 = 6.3;
double d2 = 3.6;

Byte b1    = new Byte (b);
Short s1   = new Short(s);
Integer i1 = new Integer(i);
Long l1    = new Long(l);
Float f1   = new Float(f);
Double d3  = new Double(d1);
Double d4  = new Double(d2);
```

Auf Objekte der Hüllklassen können die arithmetischen Operatoren (+ - * / %) NICHT angewendet werden. Sie besitzen hierfür spezielle Methoden, die auch diverse Umwandlungen erlauben: in einen String, in einen Zahlenwert, Wertvergleich usw.

Darüber hinaus gibt es spezielle Klassen für finanzmathematische Berechnungen mit hoher Genauigkeit. Die Klassen **`BigInteger`** und **`BigDecimal`** erstellen Objektdatentypen, die über vielerlei Attribute und Methoden verfügen. Hierzu gehören auch alle algebraischen und mathematischen Operationen.