



Java Einführung

VARIABLEN und DATENTYPEN

Kapitel 2

Inhalt dieser Einheit

- Variablen (Sinn und Aufgabe)
- Bezeichner
- Datentypen, Deklaration und Operationen
- Typenumwandlung (implizit/explicit)

Variablen

- Daten werden in **Variablen** gespeichert. Variablen sind aus der Mathematik bekannt: $y=5+x$
- Variablen haben einen **Namen** (Bezeichner) und einen **Wert**.
Bsp. x kann den Wert 8 haben.
- Variablen sind Behälter, die jeweils nur einen Wert eines bestimmten **Datentyp** beinhalten können.
Bsp. x ist vom Datentyp 'ganze Zahl' (*Integer*), die Variable kann daher keine Zeichenkette enthalten.
- Variablen müssen vor der Verwendung **deklariert** werden, dabei wird der Name und der Datentyp festgelegt.

Bezeichner

- **Bezeichner** benennen Variablen, Klassen, Instanzen, Methoden,
 - *beginnen* mit Buchstaben, “_“ oder “\$“,
 - anschließend *folgen* Buchstaben, Ziffern, Unterstrich, Dollar und/oder Unicode-Zeichen.
- **Zur besseren Lesbarkeit** wird eine einheitliche Form der Benennung der Klassen, Objekte, Methoden und Variablen empfohlen. Besteht ein Bezeichner aus mehreren Worten werden Sie folgendermaßen zusammengesrieben: `nameOfTheUser`

Bezeichner (forts.)

- **Variablennamen (und Instanznamen):** kleiner Anfangsbuchstabe. z.B.: `zaehler` oder `meinZaehler`
- **Konstantennamen:** komplett groß z.B.: `MAXINT`
- **Klassennamen:** ein Hauptwort, den ersten Buchstaben einer Klasse groß schreibt. z.B.: `MeineErsteKlasse`
- **Methodennamen:** sind Verben, beginnen mit Kleinbuchstaben. z.B.: `getNameFromUser ()`

Datentypen und Werte

- Es gibt unterschiedliche Kategorien (Datentypen) von Werten:
 - Ganze Zahlen: 2, 4754, -50
 - Gleitkomma (Reelle-) Zahlen: -3.4, 34.98, 34.83945
 - Wahrheitswerte: Richtig/Falsch
 - Zeichen: 'a', 'b'
 - Zeichenketten: "Hello World!"
- Unterschiedliche Datentypen benötigen unterschiedlich viel Speicherplatz; z.B. Wahrheitswerte 1 Bit oder ASCII-Zeichen 7 Bit
- Grund für den Einsatz von Datentypen:
 - Überprüfung sinnvoller Operationen (verhindert Addition "hallo"+3)
 - Reservierung passenden Speicherplatzes

Primitive Datentypen

In Java *eingebaute* Datentypen:

- Ganze Zahlen: `byte`, `short`, `int`, `long`
- Gleitkomma (Reelle-) Zahlen: `float`, `double`
- Zeichen: `char`
- Wahrheitswerte: `boolean` (`true/false`)

Die Datentypen und ihre Eigenschaften (Größe, Genauigkeit) hängen nicht (wie bei anderen Programmiersprachen) vom jeweiligen Computersystem ab, sondern sind in Java einheitlich festgelegt.

Primitive Datentypen (forts.)

Typ	von	bis	
byte	-128	127	Ganzzahlen
short	-32768	32767	
int	-2.147.483.648	2147483647	
long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	
char	0	65535	Unicode
boolean	true	false	Wahrheitswerte
float	$\pm 1,40239846 \cdot 10^{-45}$	$\pm 3,40282347 \cdot 10^{38}$	Gleitkom- mazahlen
double	$\pm 4,94065645841246544 \cdot 10^{-324}$	$\pm 1,79769313486231570 \cdot 10^{308}$	

Ganze Zahlen

(Primitive Datentypen)

- Datentypen `byte`, `short`, **`int`**, `long`
- Operationen (Operanden und Ergebniswert sind ganze Zahlen)
 - + Addition
 - Subtraktion
 - * Multiplikation
 - / Ganzzahldivision
 - % Restbildung (Modulo-Funktion)
- Beispiele: `1` , `-45`

weitere Operatoren folgen...

Gleitkommazahlen

(Primitive Datentypen)

- Datentypen `float`, `double`
- Operationen: (Operanden und Ergebniswert sind Gleitkommazahlen)
 - + Addition
 - Subtraktion
 - * Multiplikation
 - / Division
- Beispiel: `2.77`, `1.0` (Achtung `.` als Komma!)



weitere Operatoren folgen...

Zeichen

(Primitive Datentypen)

- Datentyp `char`
- Operationen: (Operanden und Ergebniswert sind Zeichen oder Zeichenketten)
 - + (Verkettung)
- Beispiel: ``a`` , ``z``
(Achtung einfaches Anführungszeichen!)

weitere mögliche Werte für Character-Datentypen auf der nächsten Folie...

Zeichen (forts.)

- Tastatur-, Unicode oder Oktal-Zeichen in einfachen Hochkomma
 - Tastaturzeichen 'Z'
 - Unicode-Zeichen '\u005A'
 - Oktal-Zeichen '\132'
- Sonderzeichen (Auswahl der Escapesequenzen)
 - '\b' Backspace
 - '\"' Doppeltes Hochkomma
 - '\n' linefeed
 - '\\' Backslash
 - '\r' carriage return
 - '\'' Einfaches Hochkomma
 - '\t' Tabulator

Wahrheitswerte

(Primitive Datentypen)

- Datentyp `boolean`
- Operationen: Ergebniswert sind wieder Wahrheitswerte

`a && b` (logisches UND: a ist wahr und b ist wahr)

`a || b` (logisches ODER: a ist wahr oder b ist wahr)

`!a` (logisches NICHT: ist wahr wenn a falsch ist)

- Mögliche Werte: `true`, `false`

Wahrheitswerte (forts.)

Folgende Operatoren ergeben Wahrheitswerte:

< (kleiner)

<= (kleiner oder gleich)

== (gleich)

!= (ungleich)

>= (größer oder gleich)

> (größer)



z.B. Vergleich von zwei Integer-Variablen

```
int a = 1;
```

```
int b = 2;
```

```
boolean bool = a <= b;
```

Ergebniswert bool: true

Variablendeklarationen

- Variablen haben einen **Namen** und sind Behälter für **Werte** eines bestimmten **Datentyps**
- Damit Variablen verwendet werden können, müssen sie zuerst **deklariert** werden.

z.B.:

```
int i;
```

```
short x, y;
```

Deklaration u. Initialisierung

- Deklaration: *Datentyp name;*
z.B.: `int` abgaben;
`boolean` an;
- mehrere Variable vom selben Typ:
Datentyp name1, name2, name3;
z.B.: `int` gehalt, lohn, provision;
- Wertzuweisung: *variable = wert;*
z.B.: `gehalt = 3000; provision = 2000;`
`an = true;`

Deklaration u. Initialisierung (forts.)

- Deklaration mit Initialisierung

```
int gehalt = 3000;  
int provision = 100;  
double x = 0.000001;  
char zeichen = 'a';  
boolean aus = false;
```

- Zugriff auf Variablenwerte

```
gehalt = gehalt + provision;  
System.out.println("Gehalt: "+gehalt);
```

Automatische Initialisierung von primitiven Datentypen

- Wird eine Variable bei der Deklaration nicht Initialisiert (Bsp. `int i;`), erhält sie automatisch folgenden Wert:
 - Zahlen: `0` bzw. `0.0`
 - Wahrheitswerte: `false`
 - Zeichen/Referenzen*: `null` (bedeutet: leer)
- **besser:** explizite Initialisierung
 - `int provision = 0;`

*Referenzen werden erst später behandelt!

Datentypen-Beispiel

```
class ArithmeticDemo {  
    public static void main(String[] args) {  
        int i = 37;  
        int j = 42;  
        double x = 27.475;  
        double y = 7.22;  
  
        System.out.println("Variablenwerte...");  
        System.out.println(" i = " + i);  
        System.out.println(" j = " + j);  
        System.out.println(" x = " + x);  
        System.out.println(" y = " + y);  
  
        System.out.println("Addieren...");  
        System.out.println(" i + j = " + (i + j));  
        System.out.println(" x + y = " + (x + y));  
    }  
}
```

Variablen-
deklarationen

Ausgabe der
Variablenwerte

Addition

Datentypen-Beispiel Teil2

```
System.out.println("Subtrahieren...");
System.out.println(" i - j = " + (i - j));
System.out.println(" x - y = " + (x - y));

System.out.println("Multiplizieren...");
System.out.println(" i * j = " + (i * j));
System.out.println(" x * y = " + (x * y));

System.out.println("Dividieren...");
System.out.println(" i / j = " + (i / j));
System.out.println(" x / y = " + (x / y));

System.out.println("Restwertbestimmung...");
System.out.println(" i % j = " + (i % j));
System.out.println(" j % i = " + (j % i));

System.out.println("Vermischte Typen...");
System.out.println(" j + y = " + (j + y));
System.out.println(" i * x = " + (i * x));
}}
```

Subtraktion

Multiplikation

Division

Rest

gemischte
Datentypen

Ausgabe des Programms

Variablenwerte...

i = 37
j = 42
x = 27.475
y = 7.22

Addieren...

i + j = 79
x + y = 34.695

Subtrahieren...

i - j = -5
x - y = 20.255

Multiplizieren...

i * j = 1554
x * y = 198.37

Dividieren...

i / j = 0
x / y = 3.8054

Restwertbestimmung...

i % j = 37
j % i = 5

Vermischte Typen...

j + y = 49.22
i * x = 1016.58

Konstanten

- Werte sollen während des gesamten Programmablaufes gleich bleiben.
- Konstante ist kein weiterer Datentyp.
- Deklaration: Sie werden mit dem Schlüsselwort `final` deklariert.

```
final Datentyp NAME = Wert;
```

```
z.B.: final double PI = 3.1416;
```

Typumwandlungen

- Java ist **streng typisiert**, Variablen dürfen nur Werte vom deklarierten Typ zugewiesen werden.
- Vergleichsoperatoren vergleichen nur Ausdrücke gleichen Typs.
- Es gibt **implizite** und **explizite Typumwandlung**

```
int ganz = 3;
```

```
double komma = ganz; //implizit
```

```
float komma2 = (float) ganz; //explizit
```

Implizite Typumwandlung

- In manchen Fällen wird automatisch vom Compiler umgewandelt (**implizite Typumwandlung**)
 - byte, short => int => long
 - int => float
 - float => double
 - char => int
- Beispiel:

```
short s = 3;           // s=3
int i = s;            // i=3
float g = 3 + i;      // g=6.0
int b = 'a';         // b=97*
int i = 'z' - 'a';    // i=25*
```

*Ascii-Werte a=97, z=122

Implizite Typvergrößerungen

Java nimmt diese Konversionen, falls nötig, bei Zuweisungen, Methoden- und Konstruktoraufrufen und bei der Auswertung von Ausdrücken implizit vor.

Zuweisungskompatibilität:

`byte -> short -> int -> long -> float -> double`

Kein Informationsverlust!

Ausnahme beim Übergang zu den Gleitkommazahlen:

bei `long -> float` wird **Genauigkeit verloren**

Explizite Typumwandlung (Cast)

Explizite Angabe des umzuwandelnden Typs

```
double d = 7.99;
```

```
int i = (int) d;  /*i =7, da keine  
Nachkommastellen beim Datentyp int */
```

```
int zahl = 33000;
```

```
short s = (short) zahl; //s=-32536
```

```
/*Zahl außerhalb des Wertebereichs
```

```
33000 binär = 1000 0000 1110 1000
```

```
Die erste Ziffer wird nun als Minuszeichen
```

```
interpretiert und ergibt den Wert -32536
```

```
im 2er-Komplement (-32768 + 232)*/
```

ACHTUNG!

Die Typverkleinerung kann zu falschen Werten führen, wenn der Wertebereich des Zieldatentyps nicht ausreicht, den Wert darzustellen.

Exkurs: Zeichenketten (Strings)

- **Achtung:** Zeichenkette ist keine primitiven Datentyp sondern ein Objekt-Datentyp*
- Operationen: Verkettung, Ergebniswert ist wieder eine Zeichenkette
+ (Verkettung)
- Mögliche Werte:
z.B. "Hallo World" = Kombinationen vom Typ char
- Initialisierung

```
String s = "A short way to initialize";
```



```
(oder* String id = new String( "The long way" ); )
```

* Wird später genau behandelt

Exkurs: Unicode

	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
1	SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072

0000	NUL	<control> = NULL
0001	SOH	<control> = START OF HEADING
0002	STX	<control> = START OF TEXT
...		
0060	`	GRAVE ACCENT • this is a spacing character → 02CB ` modifier letter grave accent → 0300 ◌ combining grave accent → 2035 ` reversed prime
0061	a	LATIN SMALL LETTER A
0062	b	LATIN SMALL LETTER B
...		

Entspricht im unteren Bereich ASCII

Bsp: `char c = '\u0060';`

Nach dieser Einheit sollten Sie ...



- Die verschiedenen Arten von Variablen kennen.
- alle primitiven Datentypen deren Initialisierung und gängigen Operationen.
- Variablen verschiedenen Typs definieren, deren Typ umwandeln, und damit rechnen können.
- Die Konventionen zur Bezeichnung von Variablen, Methoden und Klassen kennen.