Das erste Programm-Teil 3

Mit der Klasse „Scanner“ sind wir nun in der Lage, unsere Variablen mittels Eingabe zu definieren. Durch Ausprobieren verschiedener Werte verstehen wir auch die entsprechenden Reaktionen unserer Programme. Wenden wir uns den verschiedenen Variablentypen zu. Kennengelernt haben wir die Variablentypen:

int

double

Wir kennen deren Wertebereich, ihren Syntax und ihre Verwendung. Wann welcher Typ zu welchem Zeitpunkt zur Anwendung kommt, bleibt jedem Programmierer selbst überlassen, solange die unterschiedlichen Typen ihren Zweck erfüllen. Mit dem Variablentyp

byte

können wir Ganzzahlen im Bereich von -128 bis 127 speichern und darstellen, wobei dabei nur 1 Byte vom Speicherplatz reserviert wird. Zur Darstellung von Würfelaugen, der Anzahl von Karten, Monaten, Wochen oder Tagen, ist dieser Variablentyp völlig ausreichend. Geben wir also in unser Programm eine byte-Variable ein. Das geschieht ab Zeile 27:

```java
1: //Hier - ganz oben - importieren wir uns die Klasse "Scanner"
2: import java.util.Scanner;
3: //erstellen einer Klasse "VariablenEingabe"
4: public class VariablenEingabe {
5:
6: //erstellen der "main"-Methode
7:   public static void main (String[] args){
8:   //Deklarierung unserer Variablen, hier vom Typ Double
9:   //Wir können mehrere Variablen gleichen Typus in einer Zeile deklarieren
10:  //Bitte mit , (Komma) trennen
11:  double zahl1, zahl2, zahl3;
12:  //die Scannervariable generieren
13:  //Aufruf Scanner - Variablenname = neue (new) Scannervariable eingeben
14:  Scanner variScan = new Scanner (System.in);
15:  //Ausgabe der Eingabeaufforderung
16:  System.out.println("Bitte 1.Zahl eingeben: ");
17:  //Eingabewert in zahl1 speichern
18:  zahl1=variScan.nextDouble();
19:  //Ausgabe der 2. Eingabeaufforderung
20:  System.out.println("Bitte 2.Zahl eingeben: ");
21:  //Eingabewert in der Variable zahl2 speichern
22:  zahl2=variScan.nextDouble();
23:  //zahl3 soll das Ergebnis der Multiplikation von zahl1 und zahl2 sein
24:  zahl3=zahl1*zahl2;
25:  //Ausgabe der Berechnung
26:  System.out.println("Ergebnis der Multiplikation: "+zahl3);
27:  //deklarieren zweier byte-Variablen
28:  byte a_byte, b_byte;
29:  //Aufruf Scanner - Variablenname = neue (new) Scannervariable
30:  Scanner byteScan = new Scanner (System.in);
31:  //Eingabeaufforderung
32:  System.out.println("Bitte byte.Variable eingeben: ");
33:  //Eingabe in a_byte speichern
34:  a_byte=byteScan.nextByte();
35:  //Eingabe
36:  System.out.println("Bitte die 2. byte-Varaiable eingeben: ");
37:  //Eingabe in b_byte speichern
38:  b_byte=byteScan.nextByte();
39:  //Ausgabe von a_byte und b_byte
40:  System.out.println("Ihre Eingaben: "+a_byte +" "+b_byte);
41:  }
42: }
```

Listing 3

Zeile 28

Deklaration der beiden int-Variablen

Zeile 30

Aufruf von „Scanner“, Deklaration der Variable „byteScan“

Zeile 32

Eingabeaufforderung für die erste Variable

Zeile 34

Eingabewert in „a_byte“ speichern

Zeile 36

Eingabe der zweiten Variablen

Zeile 38

Eingabewert in „b_byte“ speichern

Zeile 40

Ausgabe der beiden Variablen.

Der Variablentyp

short

stellt ebenfalls Ganzzahlen dar. Im Zahlenbereich von -32768 bis 32767 belegt dieser Typ 16 Bit – also 2 Byte. Und mit

long

deklarieren wir Ganzzahlen im Bereich -9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807 wofür und Speicher in Höhe von 64 Bit, sprich 8 Byte in Rechnung gestellt wird.

Der Variablentyp

float

ergänzt unsere Möglichkeiten der Darstellung von Zahlen im Bereich von $-3{,}40282347*10^{38}$ bis $3{,}40282347*10^{38}$. Hier, wir schon bei **double** werden die Zahlen als Fließkommazahlen gespeichert.

Also sortiert sieht es so aus:

Ganzzahlen

byte	**8 Bit**	**-128**	**+127**
short	**16 Bit**	**-32768**	**+32767**
int	**32 Bit**	**-2.147.483.648**	**+2.147.483.647**
long	**64 Bit**	**-9.223.372.036.854.775.808**	**+9.223.372.036.854.775.807**

Fließkommazahlen

float	**32 Bit**	$\mathbf{-3{,}40282347*10^{38}}$	$\mathbf{+3{,}40282347*10^{38}}$
double	**64 Bit**	$\mathbf{-1{,}7976931348623157*10^{308}}$	$\mathbf{-1{,}7976931348623157*10^{308}}$

Abschließend seien hier noch die Variablentypen

char

und

boolean

aufgeführt, die Auflistung der Variablentypen vervollständigen sollen. Ich werde auf diese Variablen noch gesondert eingehen.

char	**16 Bit**	**16-Bit-Unicode**
boolean	**1 Bit**	**true, false (wahr/falsch)**

Soweit alles klar; oder?

Zum Rechnen benötigen wir also die verschiedensten Arten von Zahlen, die wir den entsprechenden Variablen zuordnen und in diesen speichern können. Die Entscheidung, welchen Typ wir wofür verwenden, bleibt, soweit mathematisch nicht vorgeschrieben, uns überlassen – der Gedanke an Speicherplatz sollte uns in der heutigen Zeit nicht mehr behindern. Im späteren Programm werden wir noch lernen, wie die einzelnen Variablen-Typen umzuwandeln sind, damit die verschiedenen Zahlentypen zu einem gemeinsamen Ergebnis geführt werden können.

Die String-Variable

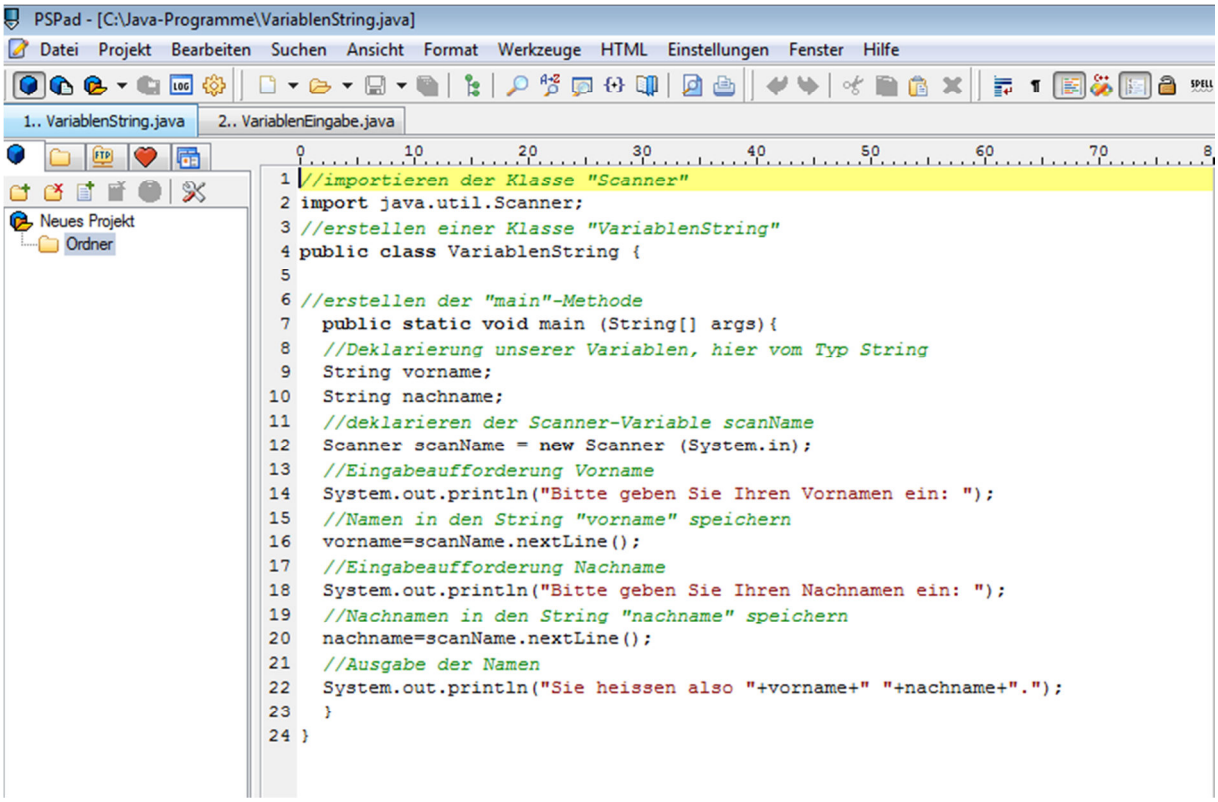
Sind die Inhalte unserer Variablen keine Zahlen sondern Zeichenketten, deklarieren wir eine

String

Variable. Strings nehmen beliebige Zeichenfolgen in sich auf und speichern diese ab.

String name;

deklariert einen String namens „name“.



Listing 4

Im Listing sind die Zeilen 1-8 keiner Erklärung nötig, die kennen wir bereits. Aber was haben wir geschrieben?

Zeilen 9 und 10

Deklarieren der beiden Variablen „vornamen und nachname“.

Zeile 12

Deklarieren der Scanner-Variable „scanName“.

Zeile 14

Eingabe der Zeichenfolge (unser Vorname).

Zeile 16

Schreibt die eingegebene Zeichenfolge in die Variable „vorname“ und speichert sie dort.

Zeile 18

Eingabe der Zeichenfolge (unser Nachname).

Zeile 20

Schreibt die eingegebene Zeichenfolge in die Variable „nachname“ und speichert sie dort.

Zeile 22

Ausgabe der Strings auf den Bildschirm.

Ende! Alles klar?

Bei der Zuweisung einer Zeichenfolge an einen String innerhalb des Quellcodes muss das so geschrieben werden:

String vorname;

vorname = „Marco“;

Natürlich nur in meinem Fall, oder Sie heißen ebenso.