Variablen in Java

1 Definition

Eine Variable ist eine Speicherzelle im Hauptspeicher, an der ein Programm Werte speichern kann.
Um diese Speicherzelle im Programm komfortabel ansprechen zu können, wird sie mit einem Namen versehen, den der Programmierer (relativ) frei wählen kann.

Java ist eine "*typsichere*" Sprache. Das bedeutet, dass zum Compile-Zeitpunkt eine Typüberprüfung vorgenommen wird, die sicherstellt, dass jede Variable

- einen ***Datentyp*** besitzt, der festlegt, welche Art von Werten die Variable aufnehmen kann, z.B. eine ganze Zahl, eine Kommazahl oder Text
- einen ***Namen*** oder ***Bezeichner*** hat, unter dem sie im Verlauf des Programms angesprochen wird und
- einen ***Wert*** enthält, der ihr vor der ersten Verwendung zugewiesen werden muss (=Initialisierung)

Die Festlegung von Datentyp und Bezeichner bezeichnet man als *Deklaration* einer Variablen. Erst nach der Deklaration kann die Variable im Programm verwendet werden.

2 Deklaration von Variablen

2.1 Bezeichner, Name der Variablen

Ein Variablenname (Bezeichner)
- muss mit Buchstaben, $ oder _ beginnen
- besteht ab dem zweiten Zeichen aus einer beliebigen Kombination von Zahlen, Buchstaben und sonstigen Zeichen
  (alle Unicode-Zeichen möglich *außer*:
  Leerzeichen, Steuerzeichen, Operatorenzeichen (z.B. + - / * % ...)
  Klammern (( ) [ ] { }), Interpunktionszeichen (. , ; : ...)
  von Java genutzte Sonderzeichen (‘$’, ”, ’)
  Wegen evtl. englischen Softwarewerkzeugen sind Umlaute und ß nicht empfehlenswert)
- darf fast beliebig lang sein (255)

Schlüsselwörter sind als Variablennamen unzulässig
Der Programmtext wird in Java mit speziellen Wörtern, den Schlüsselwörtern formuliert. Durch die Schlüsselwörter werden Datendefinitionen und Funktionalitäten beschrieben. In Java sind sämtliche Schlüsselwörter reserviert, d. h., dass die Schlüsselwörter nicht als Variablennamen zulässig sind.

Schlüsselwörter in Java
abstract, else, int , static, boolean, extends, interface, super, break, false, long, switch, byte, final, native, synchronized, byvalue, finally, new, this, case, float, null, throw, cast, for, operator, throws, catch, future, outer, transient, char, generic, package, true, class, goto, private, try, const, if, protected, var, continue, implements, public, void, default, import, rest, volatile, do, inner, return, while, double, instanceof, short (Eine verbindliche Liste findet sich in der jeweils aktuellen “Java Language Specification” auf www.oracle.com)

2.2 Datentypen

Es werden grundsätzlich 2 Arten von Datentypen unterschieden:
- einfache Datentypen (Primitive Types) und
- Referenz- oder Objektdatentypen (Reference Types)

Zu den einfachen Datentypen siehe Informationsblatt "Datentypen". Die Referenzdatentypen werden – mit wenigen Ausnahmen, z.B. String - im Zusammenhang mit der objektorientierten Programmierung besprochen.

2.3 Beispiele

Die allgemeine Formulierung der Variablendeklaration in Java in der Backus-Naur-Form[1] (die Angaben in { } können beliebig oft wiederholt):

<Datentyp> <Variablenname> {, <Variablenname>};

Beispiele:
int iZahl;
int iZahl1, iZahl2, iZahl3;

3 Wertzuweisung an Variablen

Eine (Wert-)Zuweisung ist eine Anweisung, bei der eine Variable einen anderen Wert erhält.[2]
Die Zuweisung wird durch den ***Zuweisungsoperator =*** (Gleichheitszeichen) vorgenommen und verläuft immer von der rechten Seite des Zuweisungsoperators nach links.

Dementsprechend unterscheidet man zwischen dem "***L-Value***" einer Variablen und dem "***R-Value***":
der L-Value ist immer die Adresse im Hauptspeicher, also der Variablenname, der R-Value der Inhalt.

Also gilt allgemein:

<Variablenname> = <Ausdruck>;

<Ausdruck> ist hier ein Konstrukt, das einen Wert liefert, also z.B. ein arithmetischer Ausdruck, ein Modul-/Methodenaufruf oder ein konstanter Wert.

Es dürfen nur solche Werte zugewiesen werden, die dem vereinbarten Datentyp entsprechen bzw. *zuweisungskompatibel* sind (siehe Informationsblatt "Rechnen in Java").

Beispiele:
```
int     iZahl;
char    cZeichen;
double  dWert;

iZahl = 10;
iZahl = 5 * 9;
```

[1] Die Backus-Naur-Form dient zur Beschreibung von formalen Sprachen, wie z.B. Programmiersprachen
[2] Hardwaremäßig betrachtet handelt es sich um einen Speichervorgang. In der Speicherzellen wird ein Wert eingespeichert. Im Struktogramm verwenden wir dafür das Zeichen ← (Das verwendete Zeichen '=' in Java ist leider etwas irreführend)

```
iZahl = iZahl + 1;

cZeichen = 'U';
dWert = 23.98567;
```

falsch:

```
dWert    = 5,789;          // Dezimaltrennzeichen im englischen ist .
cZeichen = 'abc';          // char kann nur 1 Zeichen aufnehmen
cZeichen = "xyz";          // ... und auch keinen String
```

Einer Variablen kann auch direkt bei der Deklaration ein Anfangswert (Initialwert) zugewiesen werden:

Beispiele:

```
int iZahl  = 3;
int iZahl2 = 4*7, iZahl3 = 4;
```

Gültig (aber nicht gut für die Lesbarkeit) sind auch ***verkettete Zuweisungen*** der Form

```
iZahl = iZahl2 = 4*7;
```

wobei der Ausdruck ganz rechts durchgereicht wird. Es ist also jeweils auf einen gültigen L-Value zu achten:

```
iZahl = iZahl2 + 1 = 4*7;   // wäre also ungültig.
```

Achtung:
Der Zuweisungsoperator darf nicht mit dem mathematischen Gleichheitszeichen verwechselt werden!

4 Gültigkeitsbereich von Variablen

Der Gültigkeitsbereich einer Variablen bezieht sich immer auf den Anweisungsblock, in dem die Variable deklariert wurde. Sie verliert also mit der schließenden Klammer des Blocks **}** ihre Gültigkeit und damit ihren Wert.

Beispiel:

```
public class VarGueltigkeit
{
  public static void main( String[] args)
  {
    {                                   // Beginn des Anweisungsblocks
      String sText = "Hallo Welt";
    }                                   // Ende des Anweisungsblocks

    System.out.println(sText);          // sText ist hier nicht mehr bekannt:
                                        // →Kompilierfehler !
  }
}
```

Gültig ist dagegen:

```
public class VarGueltigkeit
{
  public static void main( String[] args)
  {                                    // Beginn des Anweisungsblocks
    String sText;
    {
      sText = "Hallo Welt"
    }
    System.out.println(sText);

  }                                    // Ende des Anweisungsblocks
}
```

5 Symbolische Konstanten und Literale

Symbolische Konstanten sind eine Sonderform der Variablen. Sie erhalten einen festen Wert, der – im Gegensatz zu Variablen – nicht mehr geändert werden kann.

Symbolische Konstanten werden wie Variablen mit Datentyp und Namen deklariert und erhalten dann einen Wert.
Konstanten werden mit dem Schlüsselwort **final** versehen.
Konstanten werden üblicherweise groß geschrieben.

Allgemein:

final <Datentyp> <Bezeichner> = <Ausdruck>;

Beispiel:
final double MWST = 0.19;

double dNetto = 3.19;
double dBrutto = dNetto * (1 + MWST);

Literale sind unveränderliche, ***namenlose*** Darstellungen eines Wertes. Sie besitzen also einen konstanten Wert, man kann aber nicht über einen Namen auf sie zugreifen. (Englisch „litteraly“ bedeutet „wortwörtlich“)

Beispiele:

int iZahl = 100; // 100 ist das Literal (der konstante Wert)

Ganzzahl-Literale sind vom Typ int, wenn ihnen nicht das Suffix L oder l angehängt ist. Dann sind sie vom Typ long. (D.h. 100 und 100L sind zwar mathematisch gesehen der gleiche Wert, programmtechnisch gesehen werden sie aber unterschiedlich gespeichert)

Fließkomma-Literale sind vom Typ double, wenn ihnen nicht das Suffix F oder f angehängt ist. Dann sind sie vom Typ float.

Fließkomma-Literale werden in Dezimalnotation geschrieben. Sie bestehen aus dem Vorkommateil, einem Dezimalpunkt, einem Nachkommateil, einem Exponenten und einem Suffix. Exponent und Suffix sind optional. Der Exponent wird mit einem E oder e eingeleitet. Der Vorkommateil oder der Nachkommateil können weggelassen werden.

Beispiele:

```
double dZahl;

dZahl = 4.16;
dZahl = .6;
dZahl = 6E-2;    // entspricht der Zahl 6 * 10^-2 , d.h. das E bzw e steht für die Basis 10
dZahl = 4.;

float fZahl = 3.123f;
```