

EINI LogWing/WiMa

**Einführung in die Informatik für
Naturwissenschaftler und Ingenieure**

Vorlesung 2 SWS WS 17/18

Dr. Lars Hildebrand
Fakultät für Informatik – Technische Universität Dortmund
lars.hildebrand@tu-dortmund.de
<http://ls14-www.cs.tu-dortmund.de>

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- **Prolog**
- Variablen
- Zuweisung
- Datentypen

▶ Kapitel 3: Basiskonstrukte imperativer (und objektorientierter) Programmiersprachen

▶ Unterlagen

- ▶ Echte, Klaus und Michael Goedicke: *Lehrbuch der Programmierung mit Java*. Heidelberg: dpunkt-Verl, 2000. (→ ZB)
- ▶ Gumm, Heinz-Peter und Manfred Sommer: *Einführung in die Informatik*, 10. Auflage. München: De Gruyter, 2012. (Kap. 2) (→ Volltext aus Uninetz)

Übersicht

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- **Prolog**
- Variablen
- Zuweisung
- Datentypen

- ▶ Variablen
- ▶ Zuweisungen
- ▶ (Einfache) Datentypen und Operationen
 - ▶ Zahlen (`integer`, `byte`, `short`, `long`, `float`, `double`)
 - ▶ Wahrheitswerte (`boolean`)
 - ▶ Zeichen (`char`)
 - ▶ Zeichenketten (`String`)
 - ▶ Typkompatibilität
- ▶ Kontrollstrukturen
 - ▶ Sequentielle Komposition, Sequenz
 - ▶ Alternative, Fallunterscheidung
 - ▶ Schleife, Wiederholung, Iteration
- ▶ Verfeinerung
 - ▶ Unterprogramme, Prozeduren, Funktionen
 - ▶ Blockstrukturierung
 - ▶ **Rekursion**

Variable: Mathematische Sichtweise

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- **Variablen**
- Zuweisung
- Datentypen

- ▶ Variable als **Stellvertreter** für einen unbekanntes Wert, z.B. Lösung linearer Gleichungssysteme:

$$\underline{a = b + 3}, \quad \underline{b = 2 * c}, \quad \underline{c = 13}$$

- ▶ Variable in Verbindung mit **Gleichungen** erlaubt Ersetzung/ Einsetzen gleichwertiger Beschreibung für eine Variable, z.B.:

$$a = (2 * 13) + 3 = 29$$

- ▶ Gleichungen erlauben **Operationen**, die die Lösung unverändert lassen, z .B.:

$$\begin{aligned} & a = b + 3 \\ \Rightarrow & \underline{a - 3 = b} \\ \Rightarrow & a - b = 3 \end{aligned}$$

❖ **Achtung! In Programmiersprachen werden „Variable“ und „=“ ganz anders verstanden und behandelt!**



Variable: Sichtweise in der Informatik!

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- **Variablen**
- Zuweisung
- Datentypen

▶ In der Informatik steht eine Variable für einen **Speicherplatz**, der eine bestimmte Art von Datum aufnehmen kann.

▶ **Deklaration**: Programmiersprachen verlangen i.d.R., dass die Art des Datums festgelegt wird.

▶ Eine **Variable hat** einen (Daten-)Typ.

▶ **Einfache Datentypen** werden in jeder Sprache bereitgestellt

- Wahrheitswerte: boolean
- Zeichen: char
- Ganze Zahlen: byte, short, int, long
- Fließkommazahlen: float, double

▶ Die **Deklaration** erfolgt textuell **vor** der Verwendung einer Variablen und durch Angabe des Typs, des Namens/Bezeichners und ggfs. eines initialen Wertes.

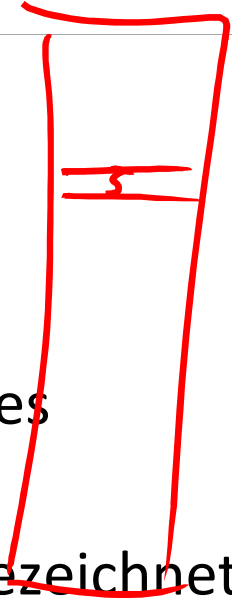
Variable: Sichtweise in der Informatik II

▶ Beispiel

```
int i = 5;
```

```
int i;  
int i = 5;
```

↔



- ▶ deklariert einen Speicherplatz zur Aufnahme eines ganzzahligen Wertes „int“,
- ▶ dieser Speicherplatz wird im folgenden mit „i“ bezeichnet und
- ▶ der Wert „5“ wird initial gespeichert.

▶ daher: Deklaration und Initialisierung vor Verwendung einer Variablen

▶ Konventionen:

- ▶ Variablen beginnen mit einem Kleinbuchstaben.
- ▶ Variablenbezeichner sollten aussagekräftig sein.

- Prolog
- Variablen
- Zuweisung
- Datentypen

▶ Semikolon = Sequenz

- ▶ „einfachste“ Kontrollstruktur
- ▶ gehört nicht zu einer Anweisung
- ▶ eigenständige Kontrollstruktur
- ▶ Trennzeichen zwischen Anweisungen: ;
- ▶ Zuweisungen können aneinandergereiht werden, z.B.:

$a = 3 ; b = a + 4 ;$

$a = 3 ;$
 $b = a + 4 ;$

In diesem Kapitel:

- Prolog
- **Variablen**
- Zuweisung
- Datentypen

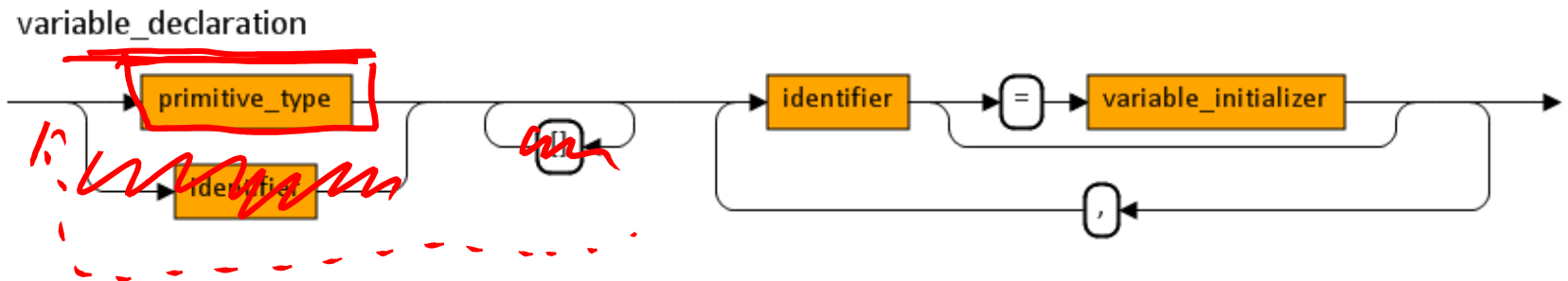
Syntaxdiagramm: Deklaration von Variablen

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

Variablen können direkt in der Deklaration mit einem Wert initialisiert werden:



gültige Deklarationen

```
int i;  
int i = 5;  
byte i, j;  
short i = 5, j;  
int i; byte j;
```

ungültige Deklarationen

```
int i  
int i; = 5  
byte i, byte j;  
small i, k = 5;  
int i, byte j;
```

Handwritten red notes:
A red scribble is over the second line. A red arrow points to the second line. A red scribble is over the third line. A red scribble is over the fourth line. A red scribble is over the fifth line. A red scribble is over the sixth line.

In diesem Kapitel:

- Prolog
- **Variablen**
- Zuweisung
- Datentypen

Einfache Datentypen

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- Datentypen

- ▶ Eine Programmiersprache stellt einen Vorrat an einfachen Datentypen bereit, z.B.:

- ▶ **Integer**

- ▶ Wertebereich (typ. 4 Byte): $-2^{31} \dots 0 \dots 2^{31}-1$
- ▶ Schlüsselwort: int
- ▶ Operationen: +, -, *, /, %
- ▶ Vergleiche: ==, !=, >, >=, <, <=
- ▶ vordefinierte Methoden, z.B. : Math.min, Math.max, Math.abs
- ▶ Konstante: z.B. 123, aber auch Integer.MAX_VALUE, MIN_VALUE

$$5 / 2 = 2$$
$$5 \% 2 = 1$$

- ▶ analog: **byte**, **short**, **long**
 - ▶ aber: Unterschiede im Wertebereich und Speichergröße

- ▶ des Weiteren:

- ▶ **float**, **double** für Gleitpunktzahlen

- ▶ ...

$$5.0 / 2.0 = 2.5$$

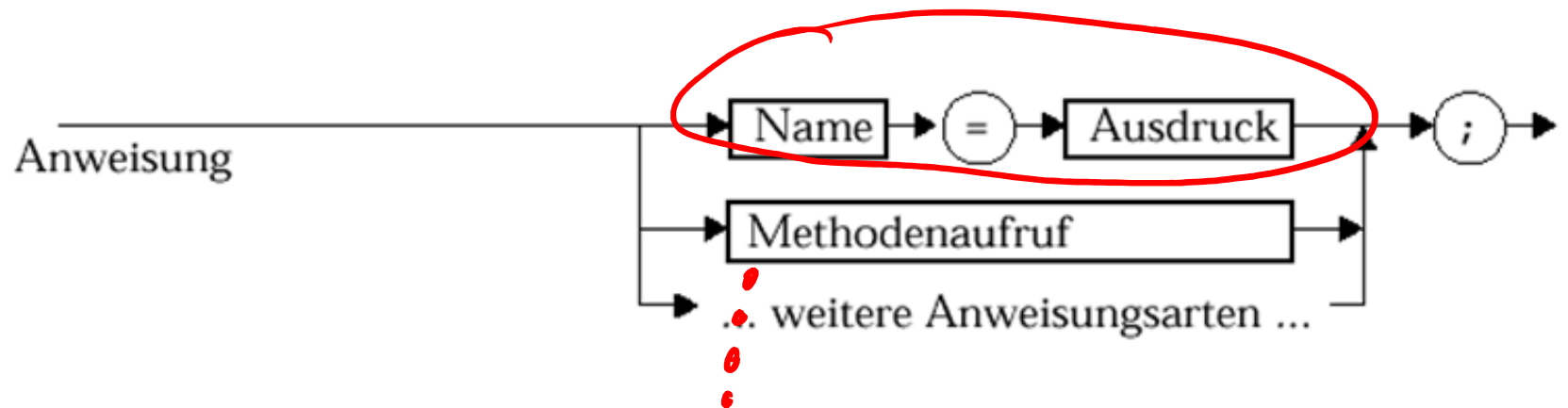
Syntaxdiagramm: Zuweisungen von Werten an eine Variablen

- ▶ Konstanten
- ▶ Ausdrücke
- ▶ Rückgabewerte von Methoden
- ▶ ...

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen



Echtle/Goedicke, Heidelberg: *Abb. 2-2 (Ausschnitt)*, S. 29 © dpunkt 2000.

In diesem Kapitel:

- Prolog
- **Variablen**
- Zuweisung
- Datentypen



Variablen

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- **Variablen**
- Zuweisung
- Datentypen

Artikel im EINI-Wiki:

- **Variable**
- **Deklaration**
 - Deklaration einer Variablen
- **Integer**
- **Basisvokabular**
- **(Codekonventionen)**

Variablen werden Werte zugewiesen I

Beispiel:

▶ Zuweisung einer Konstanten: $x = 10;$

▶ Zuweisung des Resultates eines arithmetischen Ausdrucks:

$y = 23 * x + 3 * 7 * (5 + 6);$

▶ Die beiden Anweisungen oben sind Zuweisungen an Variablen, d.h. die durch sie repräsentierten Speicherplätze haben am Ende der Ausführung einen **neuen** Wert.

▶ Der alte Wert ist **unwiederbringlich** verloren!

Variablen werden Werte zugewiesen II

Beispiel:

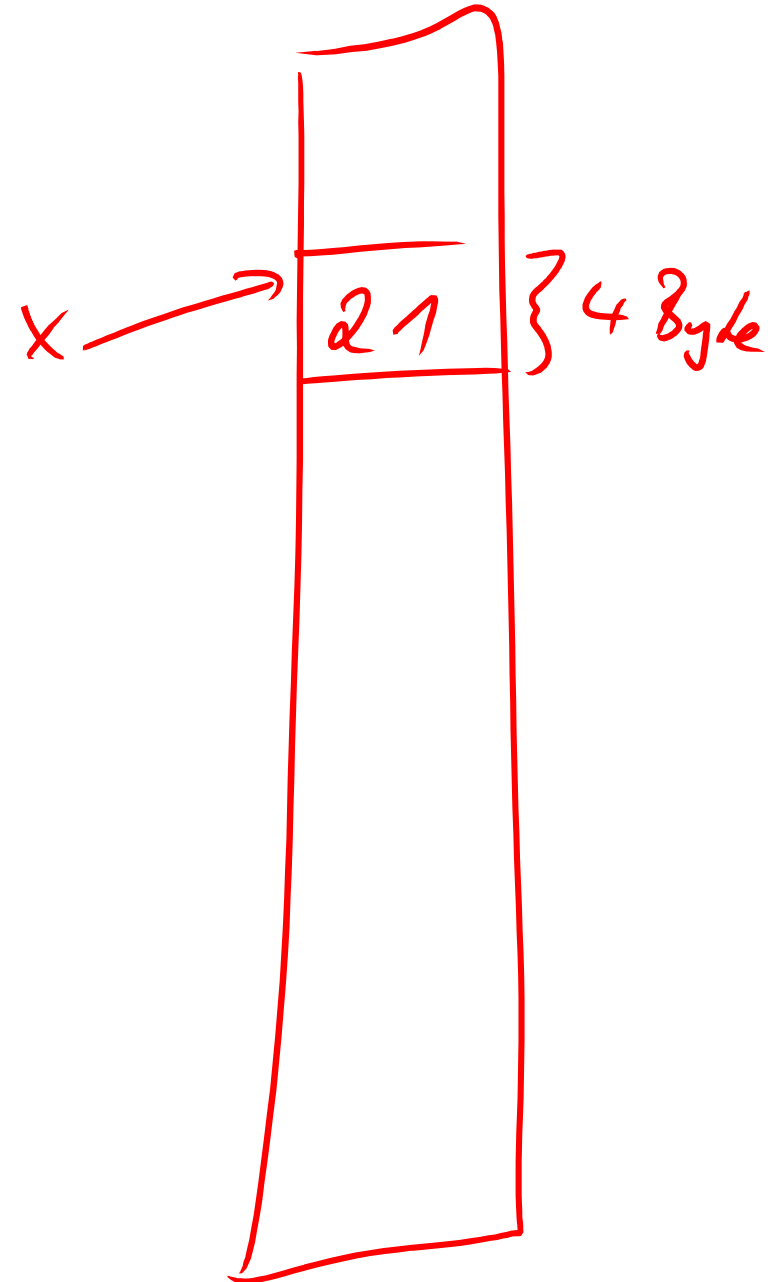
```
int x;
```

```
x = 20;
```

20

```
x = x + 1;
```

21



EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- **Zuweisung**
- Datentypen

Zuweisung im Detail I

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- **Zuweisung**
- Datentypen

- ▶ Durch eine Zuweisung wird ein errechneter Wert in einer Variablen abgelegt.

$$a = 5 + 3$$

- ▶ Der grundsätzliche Aufbau:
Variablenname = Ausdruck
b = 5 * 27;

~~a = 5 + 3~~

- ▶ Die **Verwendung des** gespeicherten **Wertes** geschieht **durch** die Angabe des **Variablennamens** in einem Ausdruck:

$$a = b * 8;$$

- ❖ **Achtung:**

Die Verwendung von Variablennamen auf der linken und rechten Seite eines Ausdrucks hat daher eine unterschiedliche Bedeutung!

Zuweisung im Detail II

Der Ablauf der Zuweisung

linkeSeite = rechteSeite ;

besteht aus insgesamt drei Schritten:

1. Die **linke Seite** der Zuweisung wird **aufgelöst**.
 - ▶ ... hier der Variablenname
2. Die **rechte Seite** (Ausdruck) wird **ausgewertet**.
 - ▶ ... Regeln zu Operatorreihenfolgen etc. werden beachtet
3. Ist die Auswertung der rechten Seite typkompatibel zu der Variablen oder kann automatisch angepasst werden, ist die **Zuweisung erlaubt** und wird kompiliert.

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- **Zuweisung**
- Datentypen

Zuweisung im Detail III

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- **Zuweisung**
- Datentypen

Die genannten drei Schritte laufen nur dann ungestört ab, wenn keine Ausnahmen registriert werden!

- ▶ Die genannte Variable könnte nicht vorhanden sein:

```
int y;  
→ int x;  
→ y = 4;
```

- ▶ Die Auswertung der rechten Seite liefert einen Fehler:

```
float x;  
x = 5.0 / 0.0;
```

x = 5.0/0;

- ▶ Typ der Variablen und des Ergebnisses sind nicht typkompatibel:

```
→ int x;  
x = "Ich bin ein Name";
```


Zuweisung im Detail IV

Programmiersprachen erlauben oft eine Reihe abstruser Spezialkonstrukte, die als „Komfort“ verstanden werden ...



EINI LogWing /
WiMa

linkeSeite2 = (linkeSeite1=rechteSeite1) op rechteSeite2 ;

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

- ▶ Nur zum Verständnis (!) anderer Programme: Eine Zuweisung ist auch ein Ausdruck!

- ▶ Der zugewiesene Wert einer Zuweisung ist der „Wert“ einer Zuweisung als Ausdruck betrachtet:

```
int a = 1, b = 2, c, d;  
c = 3 * (d = a + b + 1);
```

*a = 5;
b = 5;
c = 3;
d = 4;*

- ▶ Mehrere Zuweisungen in einem Ausdruck werden von rechts nach links abgearbeitet:

```
a = b = c = d = 5;
```

```
a = (b = (c = (d = 5))) ;
```

In diesem Kapitel:

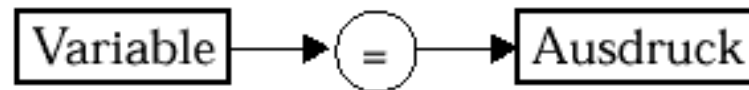
- Prolog
- Variablen
- **Zuweisung**
- Datentypen

Beispiel: Kurzformen

Kurzversionen für häufig vorkommende Zuweisungen

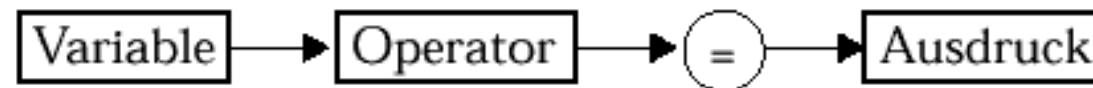
- ▶ Erhöhung einer Variablen um einen Wert
 - ▶ $a = a + 5;$ kann geschrieben werden $a += 5;$
- ▶ Verminderung einer Variablen um einen Wert
 - ▶ $a = a - 5;$ kann geschrieben werden $a -= 5;$

Zuweisung

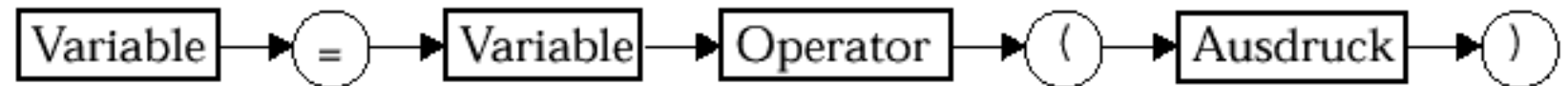


Summe += Wert!
Summe = Summe + Wert!

Spezielle Kurznotation einer Zuweisung



steht abkürzend für:



Echtle/Goedicke, Heidelberg: Abb. 2-6, S. 45 © dpunkt 2000.

$a ++;$ \Rightarrow $a = a + 1;$
 $a --;$ \Rightarrow $a = a - 1;$

- Prolog
- Variablen
- Zuweisung
- Datentypen

Zuweisung im Detail V

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- **Zuweisung**
- Datentypen

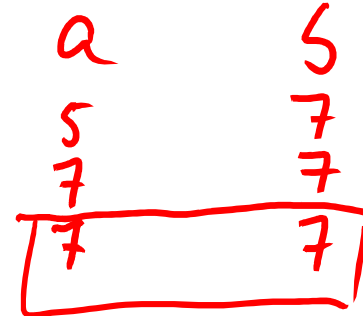
Die Zuweisungen verändern den Speicherinhalt.

- ▶ Besonderheiten, Auswirkungen
- ▶ Ringtausch: Vertausche den Wert der Variablen a und b

- ▶ Variante 1:

a = b ;

b = a ;



liefert **nicht** das erwünschte Ergebnis! **Warum ?**

- ▶ Variante 2: verwendet Hilfsvariable c

c = a ;

a = b ;

b = c ;

funktioniert ! **Warum ?**



Zuweisung

Artikel im EINI-Wiki:

→ **Zuweisung**

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- **Zuweisung**
- Datentypen

Variablen I

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Was können Variablen verwalten ?

▶ Primitive Datentypen

- ▶ die von einer Programmiersprache bereits fest vorgegeben werden.

▶ Konstruierte, komplexe Datentypen

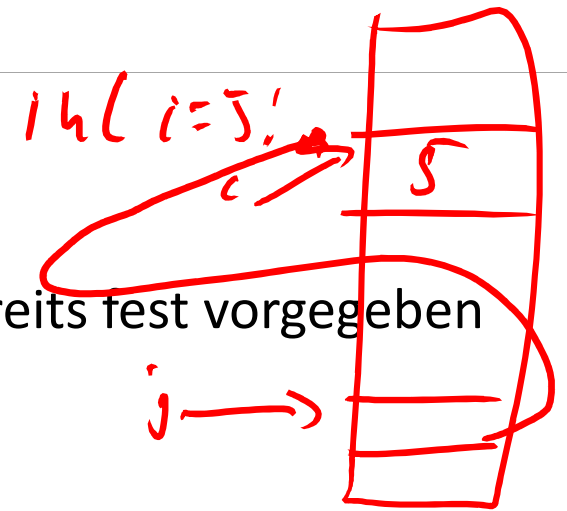
- ▶ die mit Hilfe einiger weiterer Konstrukte deklariert und genutzt werden können.

▶ Zur Erinnerung:

- ▶ „Variable“ beinhaltet: Namen, Speicherort, Typ, Inhalt

▶ Spezielle Variable können Speicheradressen von anderen Variablen als Inhalt haben.

- ▶ Idee: „Ich kenne nicht den Inhalt, aber ich weiß, wo er steht.“
- ▶ also: Eine Variable speichert eine Speicheradresse, daher kann man mit dem Wert der Variablen als Adresse im Speicher auf dessen Inhalt zugreifen.



`int i = 5;`

Variablen II

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

❖ Großer Unterschied zum Variablen-Begriff in der Mathematik!

- ▶ In der Mathematik: Variablen repräsentieren beliebigen aber festen Wert, der ggfs. auch noch unbekannt sein kann.
- ▶ In Programmen:
 - Variablen repräsentieren **Speicherplätze**, die je nach Zuweisung ihren Wert ändern können.
 - Werte müssen den Variablen explizit zugewiesen werden (keine impliziten oder unbekanntenen Werte von Variablen!)

$$a = a + 1;$$

❖ Daher: Unterschied bei der Bedeutung des Gleichheitszeichens (=):

- ▶ In der Mathematik: bezeichnet die (algebraische) Gleichheit von linker und rechter Seite einer Gleichung (algebraische Umformungen).
- ▶ In Programmen:
 - **Zuweisung** (in anderen Programmiersprachen auch mit := bezeichnet)
 - Linke und rechte Seite haben unterschiedliche Bedeutung!

Variablen und ihre Operationen

Weitere Eigenschaften

- ▶ Besonderheit von Java (und vergleichbaren Programmiersprachen)
 - ▶ Die **Bedeutung der Operationszeichen** in Ausdrücken hängt von den Typen der beteiligten Variablen ab:

```
int x;
```

```
x = 10 + 5;
```

```
System.out.print("Der Wert von x: " + x);
```

- ▶ Addition zweier Int-Werte
- ▶ Verkettung von zwei Zeichenketten mit impliziter Umwandlung des Int-Datentypes
- ▶ (Erläuterungen später)

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Datentypen

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

▶ Datentyp umfasst

- ▶ die Wertemenge und
- ▶ die zulässigen Operationen auf den Werten

z.B. ergibt es wohl wenig Sinn, in dem Ausdruck

13 * x + x/2

für die Variable X den Wert "Hallo " einzusetzen!

▶ Also müssen zusammenpassen:

- ▶ Typ einer Variablen,
- ▶ der **Wert**, der dort gespeichert ist und
- ▶ die Operationen, die auf diesen Wert angewendet werden.

Datentyp boolean

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

▶ Wahrheitswerte: „true“ und „false“

▶ Beispiele für Variable des Datentyps:

▶ boolean angemeldet, bezahlt, storniert;

▶ angemeldet = true;

▶ bezahlt = false;

4 < 5

▶ Operationen auf Werten des Datentyps boolean:

▶ logisch „oder“

||

a || b

▶ logisch „und“

&&

a && b

▶ logisch „nicht“

!

!a

▶ Zudem:

▶ Operationen, die auf den numerischen Datentypen definiert sind und Werte aus dem Datentyp boolean liefern: <, >, <=, >=, ==, !=

Beispiel für Datentyp boolean I

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

```
01 public class booleanBeispiel {
02
03     public static void main(String[] args) {
04
05         double gemessenerDruck = 5.0,
06             maximaldruck = 18.8;
07         int zuflussStufe = 2, abflussStufe = 3;
08         boolean ueberdruck, unkritisch;
09         ueberdruck = gemessenerDruck > maximaldruck;
10         unkritisch = !ueberdruck
11             && gemessenerDruck < 1.2 * maximaldruck
12             && zuflussStufe <= abflussStufe;
13
14         System.out.println( "Überdruck: "+ ueberdruck
15             + " unkritisch: "+ unkritisch );
16
17     }
18 }
```

Beispiel für Datentyp boolean II

Auswertungsreihenfolge aus dem vorigen Beispiel:

false
Ueberdruck = gemessenerDruck > Maximaldruck;

true
unkritisch = !Ueberdruck
&& gemessenerDruck < 1.2 * Maximaldruck
&& ZuflussStufe <= AbflussStufe;

5.0 *ca. 20*

true

true

```
05 double gemessenerDruck = 5.0,  
06     maximaldruck = 18.8;  
07 int   zuflussStufe = 2, abflussStufe = 3;
```

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Prioritäten von Operatoren

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

1. Höchste Priorität genießen die unären Operatoren positives (+) und negatives (-) Vorzeichen sowie das boolesche Komplement (!).

$+5$ -5 $!a$

2. Multiplikative Operationen („Punktrechnungen“: * / %)

3. Additive Operationen („Strichrechnungen“: + -)

4. Vergleiche (== != < > <= >=)

$a * (b + c)$

5. Und-Verknüpfung (&&)

6. Niedrigste Priorität besitzt die Oder-Verknüpfung (||).

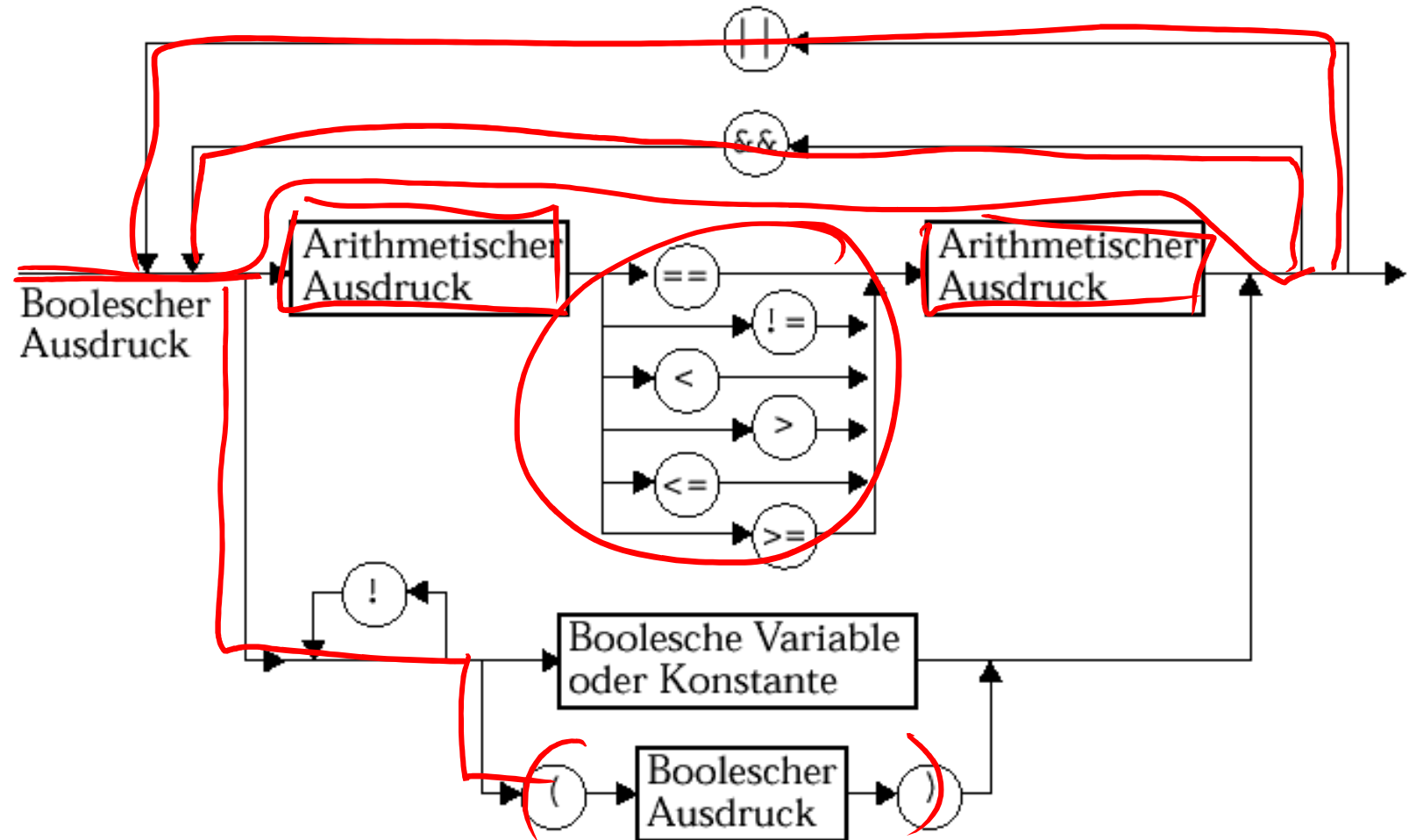
❖ Klammern setzen hilft!

Syntaxdiagramm für Boolesche Ausdrücke

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen



In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Echtle/Goedicke, Heidelberg: *Abb. 2-4*, S. 37 © dpunkt 2000.

Datentyp char

EINI LogWing /
WiMa

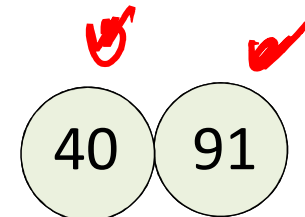
Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

- ▶ Der **Wertebereich** umfasst Groß- und Kleinbuchstaben, Sonderzeichen und Spezialzeichen (Steuerzeichen, z.B. Leerzeichen).
- ▶ **Konstanten** werden in einfache Hochkommata (') gesetzt: 'a'
'Ä' '?'
- ▶ Alle Zeichen werden in einer Tabelle (**Unicode**) mit Nummern versehen. Die Bereiche der Buchstaben und Ziffern sind zusammenhängend.
- ▶ **Steuerzeichen** werden in einer Spezialnotation angegeben (sog. Escape-Sequenzen \):
'\n' Zeilenvorschub, '\t' Tabulator, '\'' für ' , '\"' für " , '\\' für \
- ▶ Die Vergleichsoperationen sind für char auf der Basis der Unicode-Tabelle definiert:
`char rund = '(' , eckig = '[' ;`
`boolean x = rund < eckig ;`



Datentyp String I

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

▶ **Zeichenketten** sind im Datentyp **String** als Werte definiert. **String** ist **nicht** primitiv in Java.

▶ In Java spielen Werte (=Objekte) vom Typ **String** jedoch eine gewisse Sonderrolle.

▶ **Konstanten** können direkt angegeben werden; werden in Anführungszeichen **"..."** eingeschlossen:
"Der Mond scheint blau\n" *oder "!!!"*

▶ Zeichenketten können mittels **+** **verkettet** werden.

▶ Beliebige Datentypen können in Strings umgewandelt werden, wenn sie als Parameter von **println** auftauchen

▶ Als Vergleichsoperationen sind nur **==** und **!=** zugelassen.

Datentyp String II

- ▶ Der Vergleich von Objekten vom Typ **String** ist allerdings mit **Vorsicht** zu genießen!

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

```
public class StringVergleich {  
  
    public static void main(String[] args) {  
        String a = "merkwürdig",  
               b = "merkwürdig",  
               c = new String ("merkwürdig"),  
               d = new String ("merkwürdig");  
  
        System.out.println(a==b);  
        System.out.println(c==d);  
    }  
}
```

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Datentyp String III

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

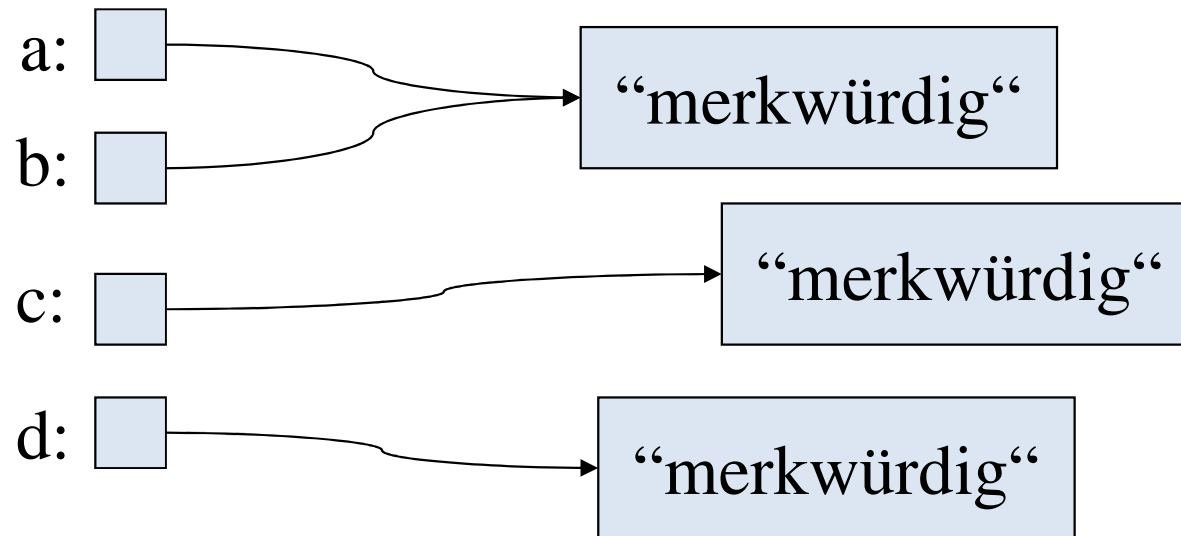
- ▶ In diesem Beispiel werden vier Objekte vom Typ `String` benutzt:

```
public static void main(String[] args) {  
    String a = "merkwürdig",  
          b = "merkwürdig",  
          c = new String ("merkwürdig"),  
          d = new String ("merkwürdig");  
  
    ...  
}
```

- ▶ Die letzten beiden Strings (c und d) sind aber unterschiedliche Objekte
 - ▶ und daher liefert
 - ▶ `==` zwischen unterschiedlichen Objekten hier `false!`

Datentyp String IV

Die Situation zwischen den Variablen a, b, c und d lässt sich durch Kästen und Pfeile charakterisieren:



- ▶ **new** erzeugt neue separate Objekte und in diesem Fall mit dem gleichen Inhalt,
- ▶ während die Deklarationen von a und b aus Ersparnisgründen auf ein und dasselbe Objekt verweisen.
- ▶ Dies entscheidet aber der Compiler.

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Streng getypte Sprachen

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

- ▶ Das gesamte System von **Datentypen** in Java wird als **streng** bezeichnet, d. h.:

```
int x = 15;  
x = "Ha (6!)";
```

- ▶ Jede Variable in Java muss mit einem Typen **deklariert** werden, der festlegt
 - ▶ welche Werte die Variable aufnehmen kann und
 - ▶ welche Operationen auf diesen Werten anwendbar sind.
- ▶ Hilft bei der Überprüfung vieler einfacher Fehler!
- ▶ Ist gelegentlich hinderlich, wenn man offensichtliche Gemeinsamkeiten von Datentypen ausnutzen möchte (z.B. Werte aus **short** und **int** verknüpfen).

Operanden von + bestimmen Ergebnistypen!

Einführung des Begriffs der Typkompatibilität:

▶ $3.5 * x$ ist

▶ **typkompatibel**, wenn etwa 3.5 float und x double ist.

▶ **nicht typkompatibel**, wenn x boolean ist.

▶ Der Operator „+“:

▶ bezeichnet Addition zwischen den numerischen Typen.

▶ sorgt für eine String-Verkettung zwischen Strings.

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Streng getypte Programmiersprachen

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

▶ Standardregeln: int + int → int etc.

▶ Ausnahme: int / int → int

$$5 / 2 = 2$$

▶ byte + byte → int

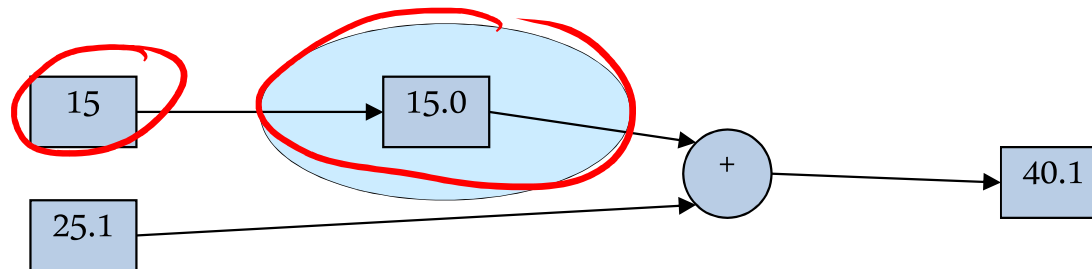
▶ short + short → int

▶ 15 + 25.1

▶ ist in Java erlaubt, aber es muss eine Regel her, die das Ergebnis bestimmt. int + float → float

▶ Es wird kein neuer Operator int + float eingeführt, sondern:

$$15 + 25.1$$



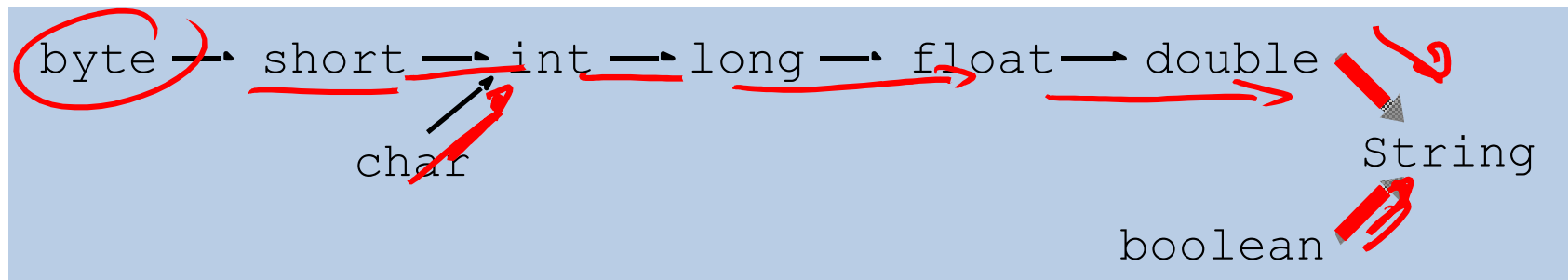
In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Implizite Typanpassung

Diesen Vorgang nennt man implizite Typanpassung.

- ▶ Implizite Typanpassung ist (in Java) nicht zwischen beliebigen Typen möglich.
- ▶ Die Umwandlung geschieht nur in Richtung des allgemeineren Typs hin:
 - ▶ int und boolean sind nicht kompatibel
 - ▶ int → float hingegen schon
- ▶ Insgesamt gilt folgende Regelung für Java:



EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Besonderheit bzgl. des Pfeils (→ String)

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

- ▶ Bereits in den ersten Beispielprogrammen:

```
System.out.println("Wert:" + a);
```

- ▶ Hier wird keine in dem obigen Sinne **implizite** Datentyp-Umwandlung durchgeführt, sondern eine **explizite**:
 - ▶ mit Hilfe der Methode **toString()**, die **implizit** durch **println()** genutzt wird.
- ▶ **toString()** ist im Java-System definiert und kann vom Programmierer verändert werden.
- ▶ Der Java-Compiler sorgt dafür, dass in Ausdrücken der Form **xyz + string** für **xyz** zunächst das zugehörige **toString()** aufgerufen wird.



Reihenfolge der Operatoren

❖ In Ausdrücken Reihenfolge der Operatoren beachten!

▶ Beispiel:

▶ `System.out.print(17 + " und " + 4)`

▶ liefert: 17 und 4

während

▶ `System.out.print(17 + 4 + " und")`

▶ liefert: 21 und

▶ Ansonsten arbeiten die Umwandlungsregeln intuitiv.

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**



Datentypen

Artikel im EINI-Wiki:

- **Datentyp**
 - Primitive Datentypen
- **Operator**
- **Boolean**
- **Boolescher Ausdruck**
- **Character**
- **Zeichenkette**
- **Instanziierung**
- **Compiler**

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

Zwischenstand

EINI LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- Prolog
- Variablen
- Zuweisung
- **Datentypen**

- ✓ Variablen
- ✓ Zuweisungen
- ✓ (Einfache) Datentypen und Operationen
 - ✓ Zahlen
integer, byte, short, long; float, double
 - ✓ Wahrheitswerte (**boolean**)
 - ✓ Zeichen (**char**)
 - ✓ Zeichenketten (**String**)
 - ✓ Typkompatibilität
- Kontrollstrukturen
 - Sequentielle Komposition, Sequenz
 - Alternative, Fallunterscheidung
 - Schleife, Wiederholung, Iteration
- ▶ Verfeinerung
 - ▶ Unterprogramme, Prozeduren, Funktionen
 - ▶ Blockstrukturierung
- ▶ Rekursion



Vielen Dank für Ihre Aufmerksamkeit!

Nächste Termine

- ▶ Nächste Vorlesung – WiMa 09.11.2017, 08:15
- ▶ Nächste Vorlesung – LogWing 10.11.2017, 08:15