

EINI LW

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

Vorlesung 2 SWS WS 15/16

Dr. Lars Hildebrand

Fakultät für Informatik – Technische Universität Dortmund

lars.hildebrand@tu-dortmund.de

<http://ls1-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

- ▶ Gumm/Sommer, Kapitel 2
- ▶ Echte/Goedicke, Einführung in die Programmierung mit Java, dpunkt Verlag

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

Variable: eine mathematische Sichtweise

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

- ▶ Variable als **Stellvertreter** für einen unbekanntes Wert
z.B. Lösung linearer Gleichungssysteme
 $a = b + 3, b = 2 * c, c = 13$
- ▶ Variable in Verbindung mit **Gleichungen** erlauben Ersetzung,
Einsetzen gleichwertiger Beschreibung für eine Variable
z.B. $a = (2 * 13) + 3 = 29$
- ▶ Gleichungen erlauben **Operationen**, die die Lösung unverändert lassen, z .B.:
$$a = b + 3$$
$$\Rightarrow a - 3 = b$$
$$\Rightarrow a - b = 3$$

In diesem Kapitel:

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

**Achtung: in Programmiersprachen werden
Variable und „=“ ganz anders verstanden und behandelt!**



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

Variable : Die Sichtweise in der Informatik

▶ Beispiel

```
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, Initialisierung** vor Verwendung einer Variablen

▶ Konventionen

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

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

Variable : Die Sichtweise in der Informatik

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

▶ Bedeutung des Semikolons

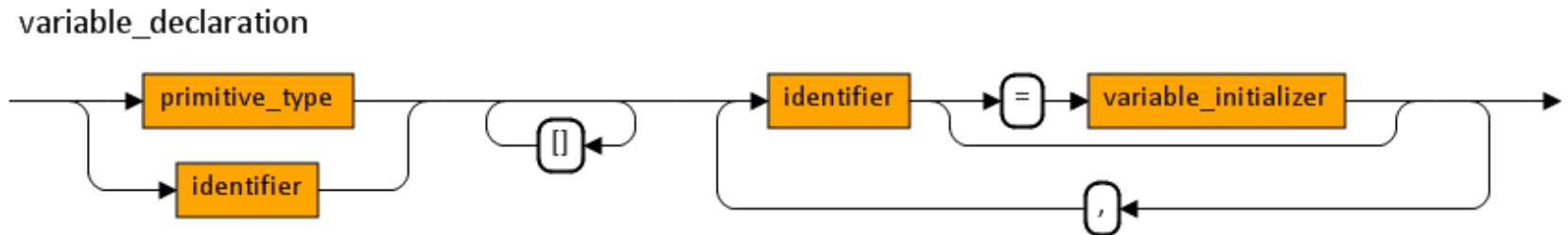
▶ Sequenz

- ▶ „einfachste“ Kontrollstruktur
- ▶ gehört nicht zu einer Anweisung, ist eine eigenständige Kontrollstruktur
- ▶ Trennzeichen zwischen Anweisungen: ;
- ▶ Zuweisungen können aneinandergereiht werden:
- ▶ z.B. $a = 3 ; b = a + 4 ;$

Syntaxdiagramm

für die Deklaration von Variablen

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;
```

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

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): $-2^{31} \dots 0 \dots 2^{31}-1$
 - ▶ Operationen: +, -, *, /, %
 - ▶ Vergleiche: ==, !=, >, >=, <, <=
 - ▶ vordefinierte Methoden, z.B. : Math.min, Math.max, Math.abs
 - ▶ Konstante: z.B. 123, aber auch Integer.MAX_VALUE, MIN_VALUE
- ▶ analog: byte, short, long
 - ▶ Unterschiede im Wertebereich und Speichergröße
- ▶ des Weiteren:
 - ▶ float, double für Gleitpunktzahlen
 - ▶ ...

Syntaxdiagramm

für die Zuweisungen von Werten an eine Variablen

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

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

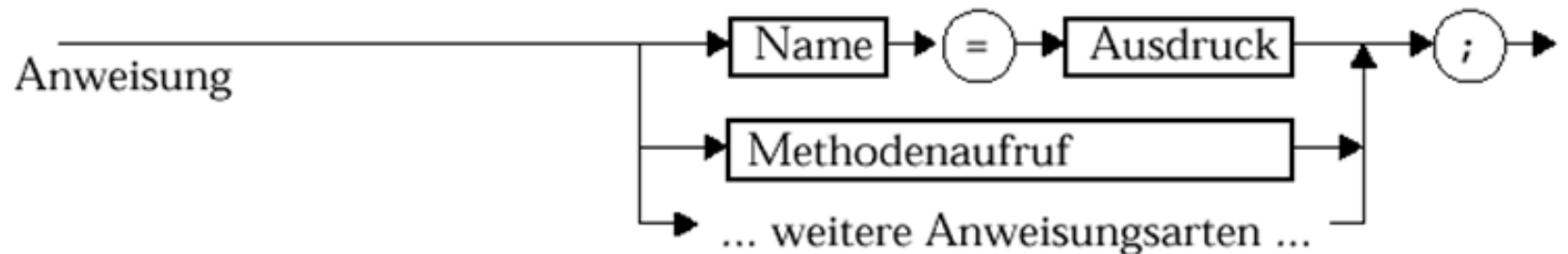


Abb. 2-2 Syntaxdiagramme

Lehrbuch der Programmierung mit Java, Ehtle Goedicke, Heidelberg, © dpunkt 2000

In diesem Kapitel:

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

Variablen werden Werte zugewiesen

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!

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

Variablen werden Werte zugewiesen

Beispiel:

```
int x;
```

```
x = 20;
```

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

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

Zuweisung im Detail

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

- ▶ Grundsätzlich wird durch eine Zuweisung ein errechneter Wert in einer Variablen abgelegt
- ▶ Der grundsätzliche Aufbau:
Variablenname = Ausdruck
b = 5*27;
- ▶ Die **Verwendung des** gespeicherten **Wertes** geschieht **durch** die Angabe des **Variablenamens** in einem Ausdruck
a = b * 8;
- ▶ **Beachte:**
die Verwendung von Variablennamen auf der linken und der rechten Seite eines Ausdrucks hat daher unterschiedliche Bedeutung!

Zuweisung im Detail

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

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 x;  
y = 4;
```

- ▶ Die Auswertung der rechten Seite liefert einen Fehler

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

- ▶ Typ der Variablen und des Ergebnisses sind nicht typkompatibel

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

Zuweisung im Detail

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



linkeSeite2 = (linkeSeite1=rechteSeite1) op rechteSeite2 ;

- ▶ 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);
```

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

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

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

Beispiel: Kurzformen

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

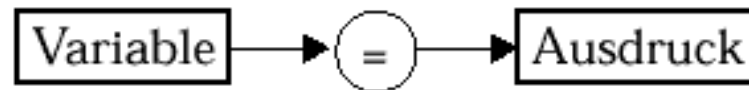
In diesem Kapitel:

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

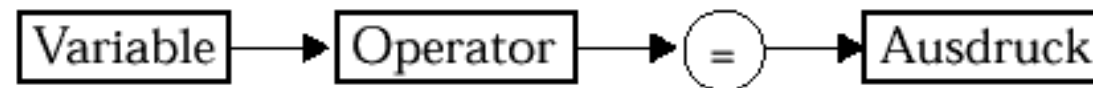
Kurzversionen für häufig vorkommende Zuweisungen

- ▶ 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



Spezielle Kurznotation einer Zuweisung



steht abkürzend für:

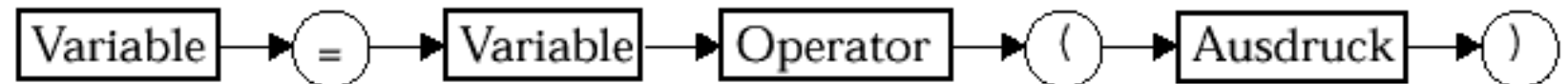


Abb. 2-6 Syntax der Zuweisung

Lehrbuch der Programmierung mit Java, Echte Goedicke, Heidelberg, © dpunkt 2000

Zuweisung im Detail

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 ?**

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

Variable: Was können Variable 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 es steht“
- ▶ also: eine Variable speichert eine Speicheradresse, daher kann man mit dem Wert der Variablen als Adresse im Speicher auf den Inhalt dort zugreifen.

- 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!)

▶ Daher auch ein großer 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

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

- 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. macht 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;`
- ▶ Operationen auf Werten des Datentyps `boolean`:
 - ▶ logisch „oder“ `||`
 - ▶ logisch „und“ `&&`
 - ▶ logisch „nicht“ `!`
- ▶ Zudem:
 - ▶ Operationen, die auf den numerischen Datentypen definiert sind und Werte aus dem Datentyp `boolean` liefern: `<`, `>`, `<=`, `>=`, `==`, `!=`

Beispiel für Datentyp `boolean`

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
10         ueberdruck = gemessenerDruck > maximaldruck;
11
12         unkritisch = !ueberdruck
13             && gemessenerDruck < 1.2 * maximaldruck
14             && zuflussStufe <= abflussStufe;
15
16         System.out.println( "Überdruck: "+ ueberdruck
17                             + " unkritisch: "+ unkritisch );
18     }
19 }
```


Beispiel für Datentyp `boolean`

Auswertungsreihenfolge aus dem Beispiel:

```
Ueberdruck = gemessenerDruck > Maximaldruck;
```

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

```
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**

Priortitäten 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 boolsche **Komplement (!)**.
 2. **Multiplikative Operationen** („Punktrechnungen“: * / %)
 3. **Additive Operationen** („Strichrechnungen“: + -)
 4. **Vergleiche** (== != < > <= >=)
 5. **Und-Verknüpfung (&&)**
 6. Niedrigste Priorität besitzt **die Oder-Verknüpfung (||)**.
- ▶ Es hilft, Klammern zu setzen!

Syntaxdiagramm für Boolesche Ausdrücke

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

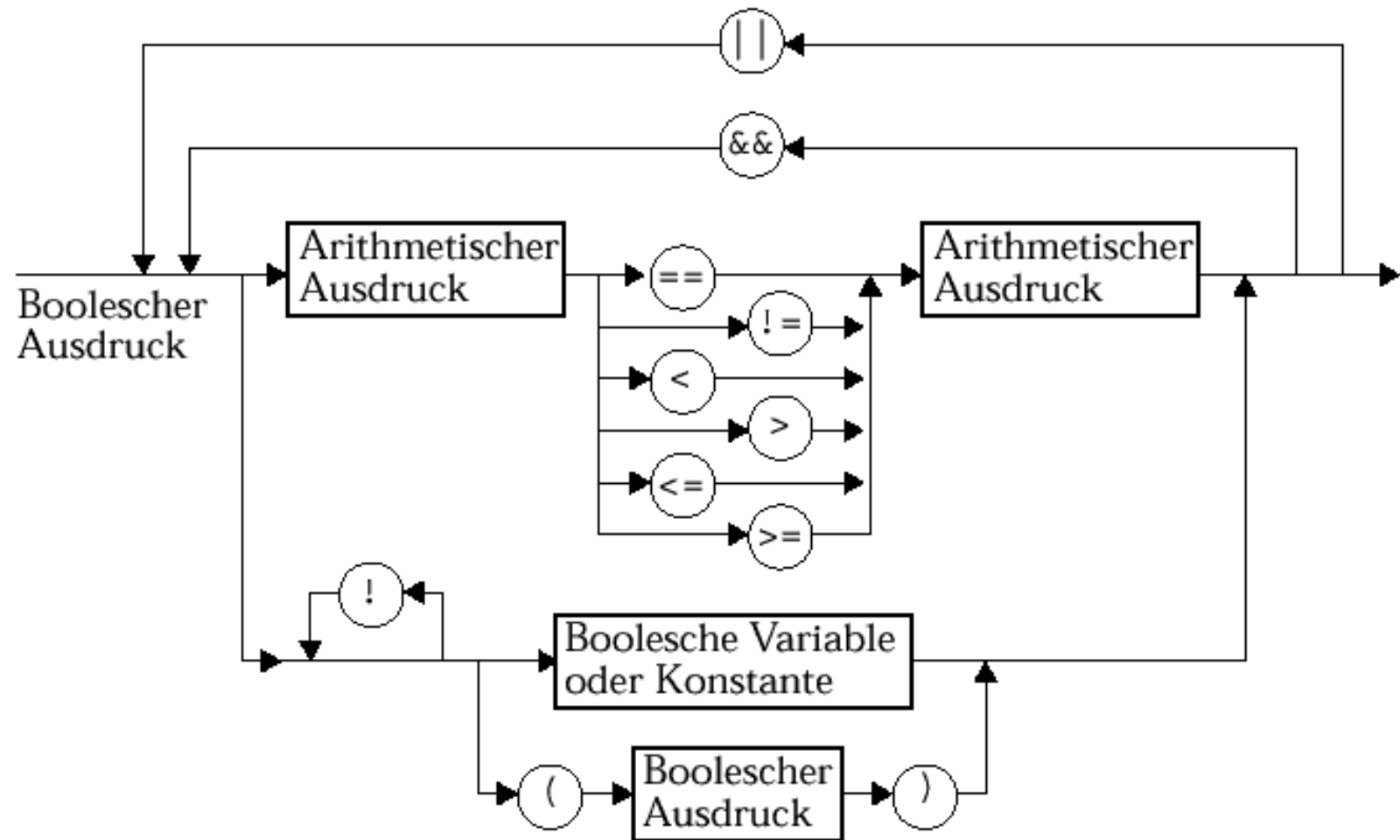


Abb. 2-4 Syntaxdiagramm eines Booleschen Ausdrucks

Lehrbuch der Programmierung mit Java, Echte Goedicke, Heidelberg, © dpunkt 2000

In diesem Kapitel:

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

Datentyp `char`

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

- ▶ Der **Wertebereich** umfasst die Groß- und Kleinbuchstaben, Sonderzeichen und Spezialzeichen (Steuerzeichen, z.B. Leerzeichen)
- ▶ **Konstanten** werden in einfache Hochkommata ' gesetzt: 'a' 'Ä' '?'
- ▶ Die Zeichen werden alle 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 ;
```

40

91

Datentyp `String`

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 Hochkommata `"..."` eingeschlossen:
`"Der Mond scheint blau \n "`
- ▶ 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`

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

```
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);  
    }  
}
```

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

Datentyp `String`

- ▶ 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!

Eini LogWing /
WiMa

Kapitel 3

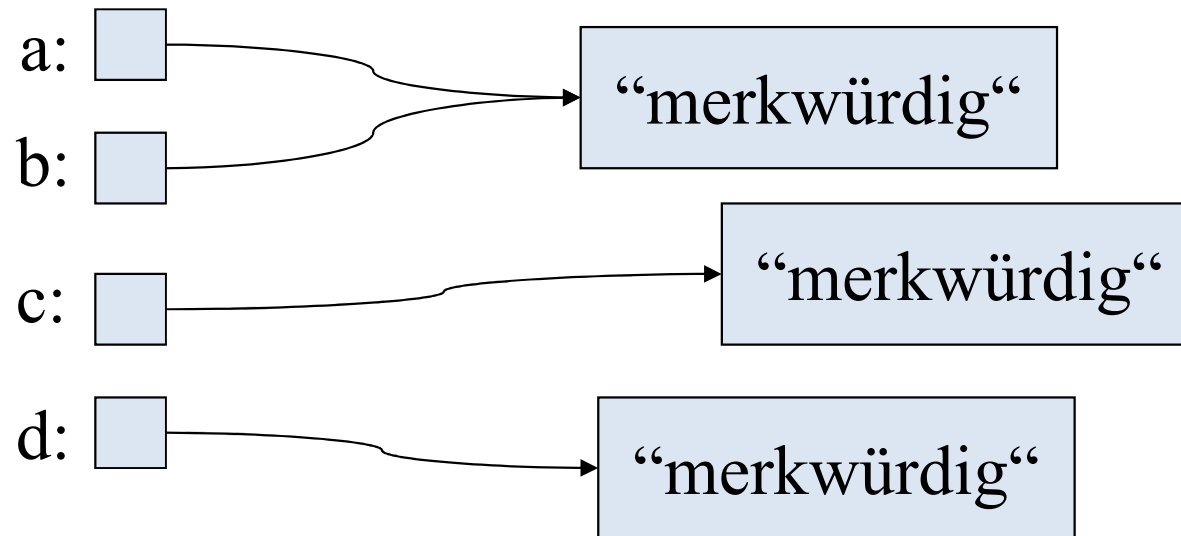
Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

Datentyp `String`

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 Deklaration 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.
- ▶ Jede Variable in Java muss mit einem Typ **deklariert** werden, der festlegt
 - ▶ welche Werte die Variable aufnehmen kann
 - ▶ 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

Daher wird der Begriff der Typkompatibilität eingeführt.

- ▶ Man wird also etwa `3.5*x` als
 - ▶ **typkompatibel** bezeichnen, wenn etwa `3.5` (float) und `x` `double` ist
 - ▶ **nicht typkompatibel** bezeichnen, wenn `x` `boolean` ist
- ▶ Des Weiteren betrachte den Operator „+“:
 - ▶ Bezeichnet Addition zwischen den numerischen Typen
 - ▶ String-Verkettung zwischen Strings

Eini LogWing /
WiMa

Kapitel 3

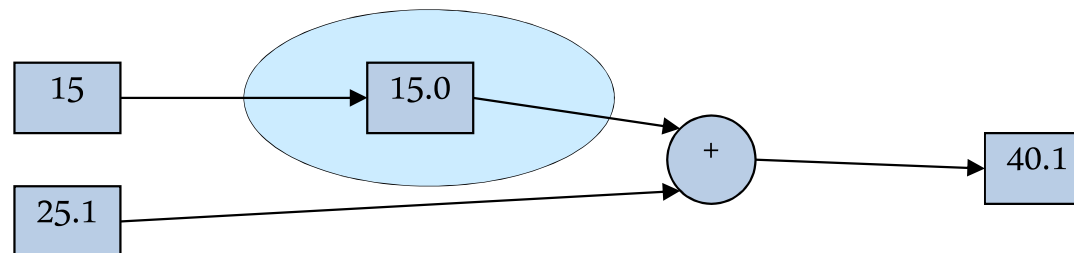
Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

In diesem Kapitel:

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

Streng getypte Fragen

- ▶ Standardregeln: $\text{int} + \text{int} \rightarrow \text{int}$ etc.
- ▶ Ausnahme:
 - ▶ $\text{byte} + \text{byte} \rightarrow \text{int}$
 - ▶ $\text{short} + \text{short} \rightarrow \text{int}$
- ▶ Des Weiteren $15 + 25.1$
 - ▶ ist in Java erlaubt, aber es muss eine Regel her, die das Ergebnis bestimmt.
- ▶ Es wird kein neuer Operator $\text{int} + \text{float}$ eingeführt, sondern:



Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

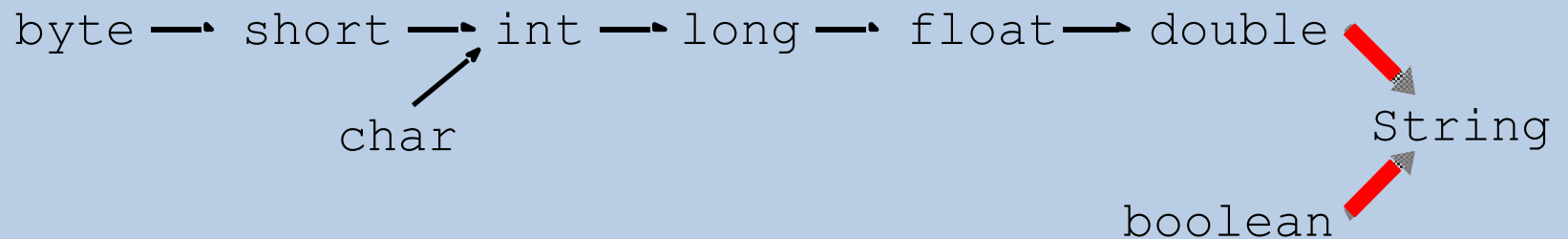
In diesem Kapitel:

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

Implizite Typanpassung

Diesen Vorgang nennt man implizite Typanpassung

- ▶ Ist (in Java) nicht zwischen beliebigen Typen möglich
- ▶ Die Umwandlung geschieht nur in Richtung des allgemeineren Typs hin
 - ▶ int und boolean bleiben 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 veranstaltet, 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!

Eini LogWing /
WiMa

Kapitel 3

Basiskonstrukte
imperativer und
objektorientierter
Programmiersprachen

- ▶ 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

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 19.11.2015, 08:15
- ▶ Nächste Vorlesung – LogWIng 20.11.2015, 08:15