EINI LW

Einführung in die Informatik für Naturwissenschaftler und Ingenieure

Vorlesung 2 SWS WS 11/12

Dr. Lars Hildebrand
Fakultät für Informatik – Technische Universität Dortmund
lars.hildebrand@udo.edu
http://ls1-www.cs.uni-dortmund.de

technische universität dortmund
Dr. Lars Hildebrand

- **Kapitel 3**
  Basiskonstrukte imperativer (und objektorientierter) Programmiersprachen

- **Unterlagen**
  - Gumm/Sommer, Kapitel 2
  - Echtle/Goedicke, Einführung in die Programmierung mit Java, dpunkt Verlag

Gliederung

Dr. Lars Hildebrand

- Variablen
  - Bezeichner, Datentyp, Speicherort, Wert
- Zuweisungen
  - Zuweisungen müssen typverträglich sein
- (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 : eine mathematische Sichtweise

Dr. Lars Hildebrand

- Variable als Stellvertreter für einen unbekannten 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 gdw a - 3 = b gdw a - b = 3

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

Variable: die Sichtweise in der Informatik

Dr. Lars Hildebrand

- 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** Daten**typen** 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

- **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“ initial belegt.

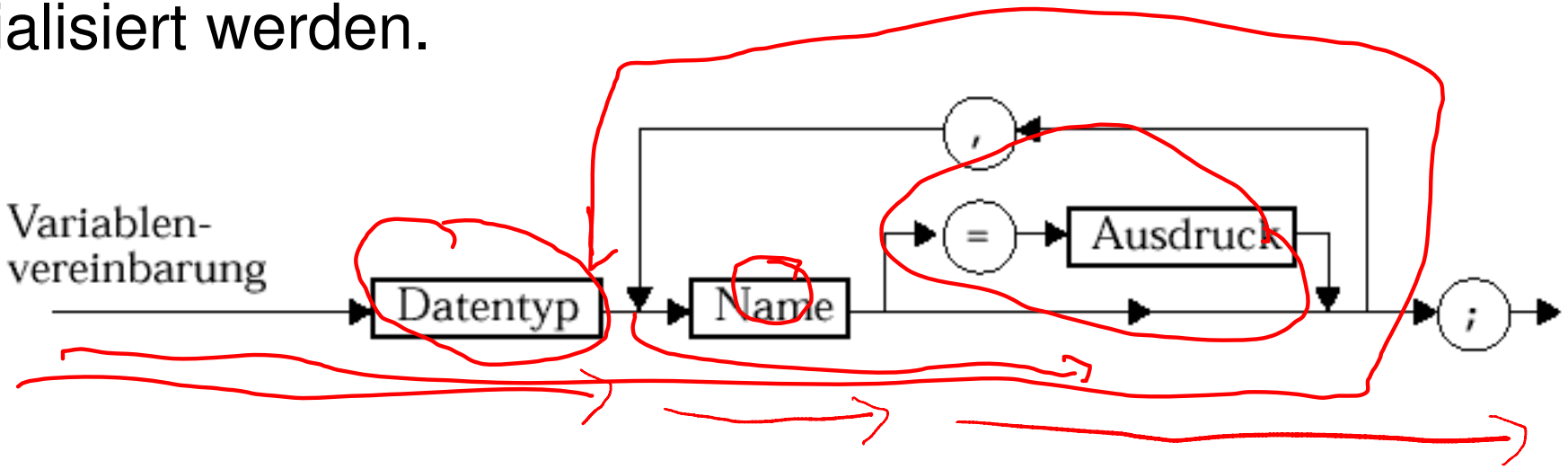
- daher: Deklaration, Initialisierung vor Verwendung einer Variablen

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

Syntaxdiagramm

für die Deklaration von Variablen

Variable können direkt in der Deklaration mit einem Wert initialisiert werden.



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

Einfache Datentypen

technische universität dortmund

Dr. Lars Hildebrand

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

- **integer**
  - Wertebereich (typ. 4 Byte): -2^{31} ... 0 ... $2^{31}-1$
  - 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

Dr. Lars Hildebrand

für die Zuweisungen von Werten an eine Variablen

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

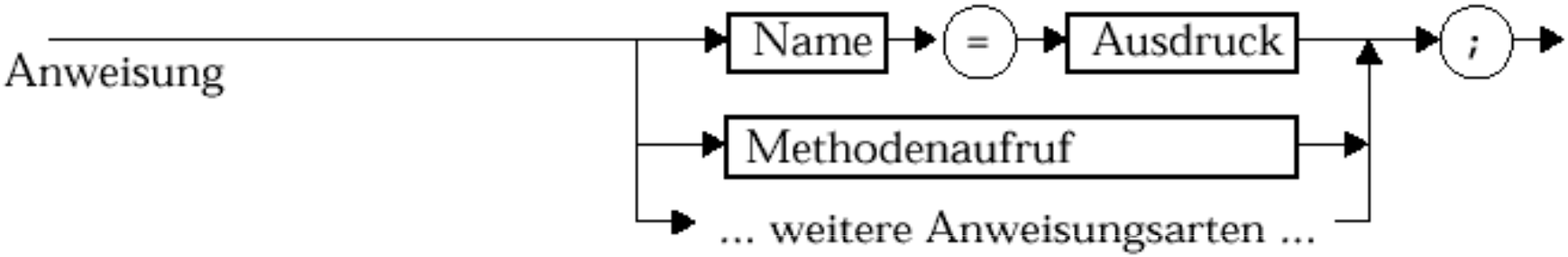


Abb. 2–2 *Syntaxdiagramme*

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

technische universität dortmund

Dr. Lars Hildebrand

Beispiel:

int x;

- Zuweisung einer Konstanten `x = 10;`

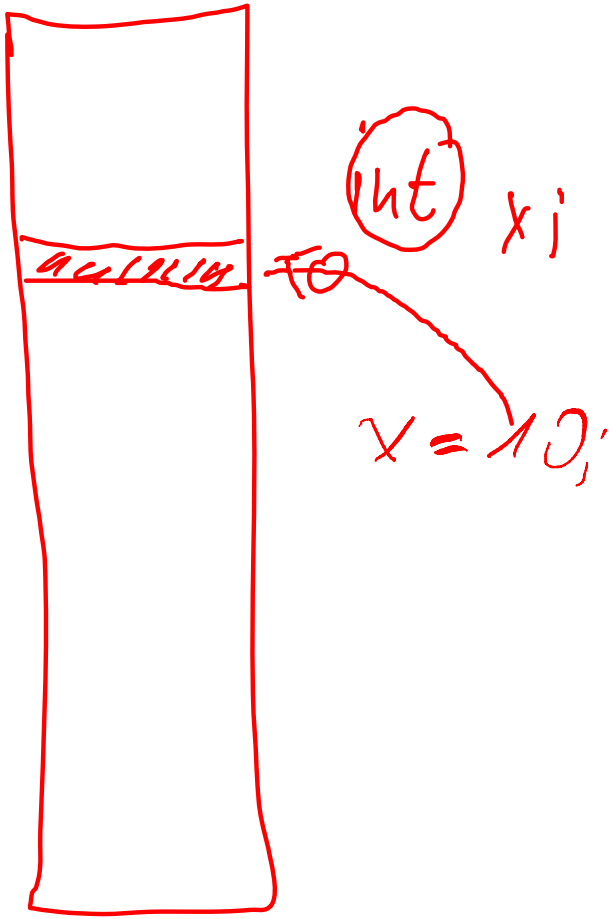
- Zuweisung des Resultates eines arithmetischen Ausdrucks

```
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!

Zuweisung: Variablen werden Werte zugewiesen



Zuweisung

Dr. Lars Hildebrand

- 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 **Variablennamens** in einem Ausdruck

  ```
  a = b *8;
  ```

- Beachte:

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

technische universität dortmund
Dr. Lars Hildebrand

Der Ablauf der Zuweisung

linkeSeite = rechteSeite ;

besteht aus insgesamt drei Schritten:

1. Die **linke Seite** der Zuweisung wird **ausgewertet**
   - ... hier der Variablenname
   - ... kann aber komplexer sein

2. Die **rechte Seite** (Ausdruck) wird **ausgewertet**
   - ... Regeln zu Operatorreihenfolgen etc. werden beachtet

3. Ist der rechten Seite typkompatibel zu der Variablen oder kann automatisch angepasst werden, erfolgt die **Zuweisung**

technische universität dortmund

Dr. Lars Hildebrand

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

- Die genannte Variable könnte nicht vorhanden sein
- Die Auswertung der rechten Seite liefert einen Fehler
- Typ der Variablen und des Ergebnisses sind nicht typkompatibel

Zuweisung

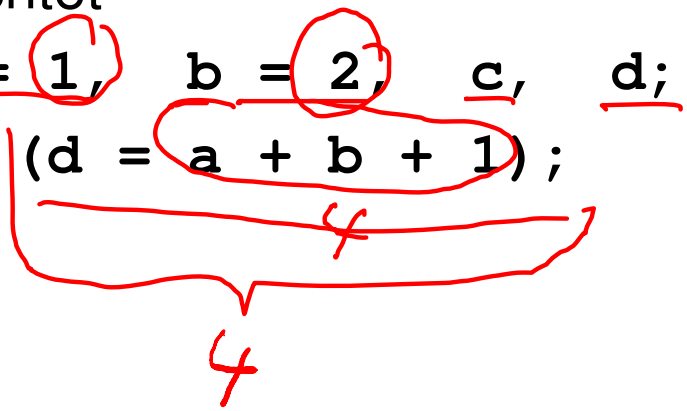
Dr. Lars Hildebrand

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:

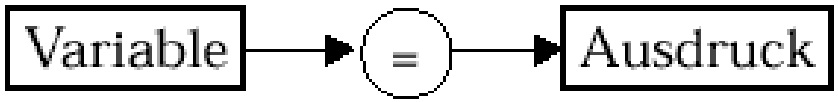
```
a=b=c=d=5;   →   a=(b=(c=(d=5)));
```

Beispiel: Kurzformen

Kurzversionen für häufig vorkommende Zuweisungen

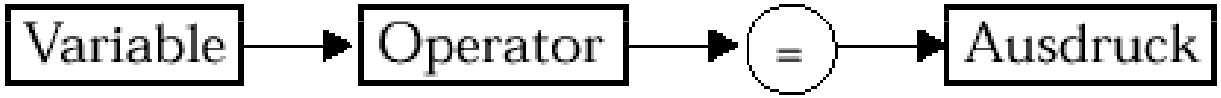
- Erhöhung einer Variablen um einen Wert
  - a = a + 5; kann geschrieben werden a += 5;
- Verminderung einer Variablen um einen Wert
  - a = a - 5; kann geschrieben werden a -= 5;

Zuweisung



a = a + 1; a++;
a = a - 1; a--;

Spezielle Kurznotation einer Zuweisung



steht abkürzend für:

Abb. 2–6 *Syntax der Zuweisung*

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

technische universität dortmund

Dr. Lars Hildebrand

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 ?**

technische universität dortmund
Dr. Lars Hildebrand

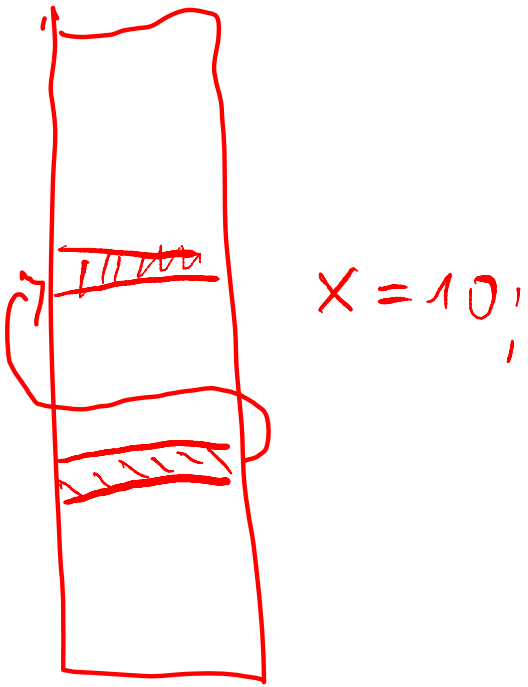
Variable: Was können Variable verwalten ?

- **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.

Variablen

Dr. Lars Hildebrand



Variable

- **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 unbekannten 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!

technische universität dortmund
Dr. Lars Hildebrand

Weitere Eigenschaften

- Besonderheit von Java (und vergleichbaren Programmier-sprachen)
  - 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

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`

technische universität dortmund

Dr. Lars Hildebrand

- 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

Dr. Lars Hildebrand

```
public class booleanBeispiel {

  public static void main(String args[]) {

  double  gemessenerDruck = 5.0,
          maximaldruck = 18.8;
  int     zuflussStufe = 2, abflussStufe = 3;
  boolean ueberdruck, unkritisch;

    ueberdruck = gemessenerDruck > maximaldruck;

    unkritisch = !ueberdruck
           && gemessenerDruck < 1.2 * maximaldruck
           && zuflussStufe <= abflussStufe;

    System.out.println( "Überdruck: "+ ueberdruck +
                        " unkritisch: "+ unkritisch );
  }
}
```

Beispiel für Datentyp boolean

Dr. Lars Hildebrand

Auswertungsreihenfolge aus dem Beispiel:

```
Ueberdruck = gemessenerDruck > Maximaldruck;

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

Prioritäten von Operatoren

Dr. Lars Hildebrand

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

Dr. Lars Hildebrand

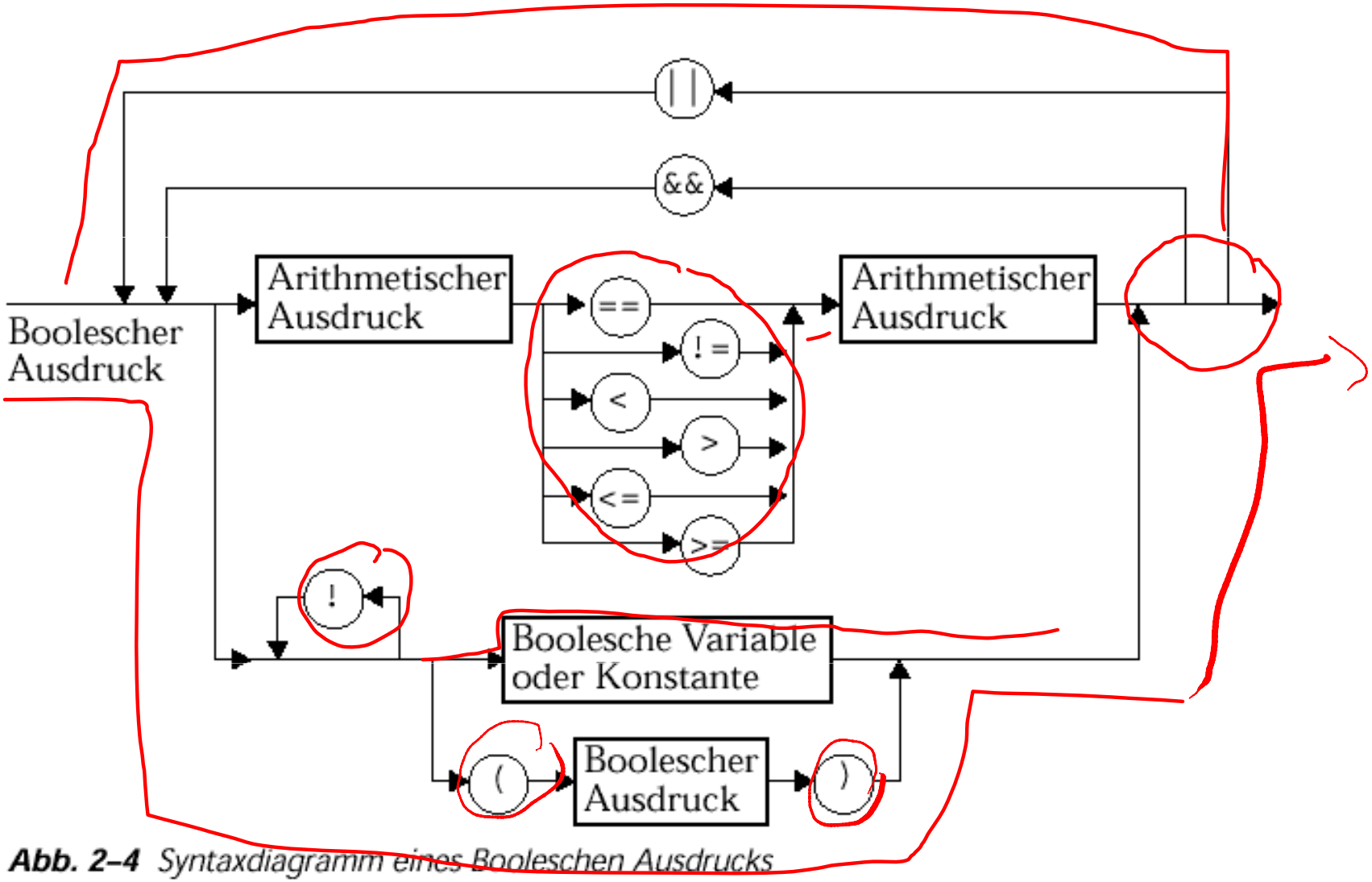


Abb. 2–4 *Syntaxdiagramm eines Booleschen Ausdrucks*

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

Datentyp `char`

- 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`

technische universität dortmund

Dr. Lars Hildebrand

- **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

- 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 + " " + (a==b) + " " +
                        c + " " + (c==d));
    }
}
```

Datentyp `String`

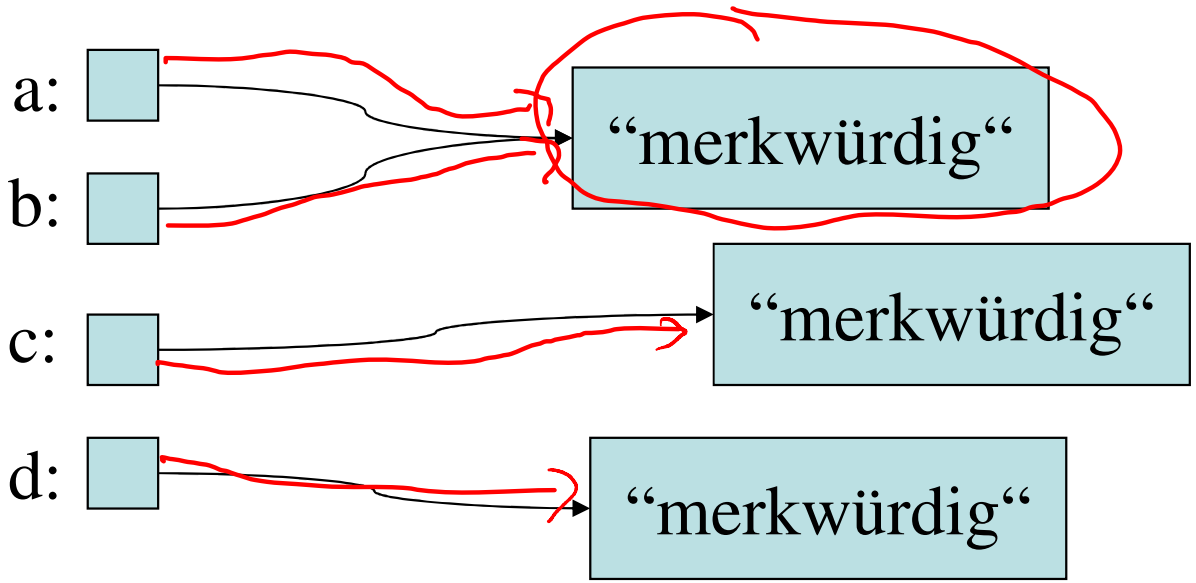
- In diesem Beispiel werden vier Objekte vom Typ `String` benutzt

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

- Die letzten beiden c,d sind aber unterschiedliche Objekte
  - und daher liefert
  - == zwischen unterschiedlichen Objekten hier false!

technische universität dortmund
Dr. Lars Hildebrand

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.

Streng getypte Sprache

Dr. Lars Hildebrand

- 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)

technische universität dortmund
Dr. Lars Hildebrand

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

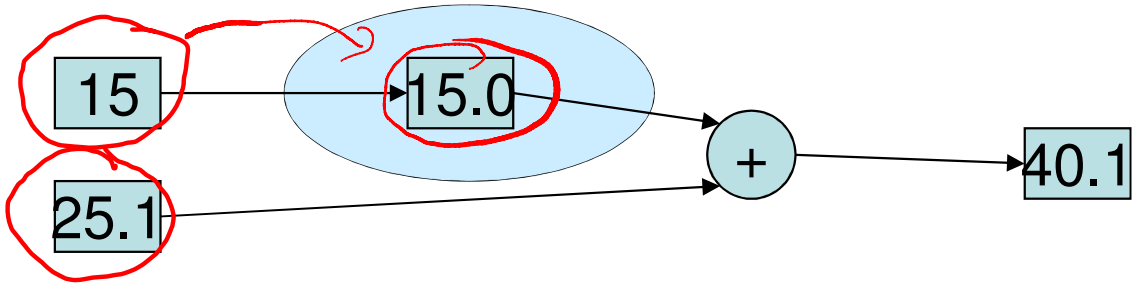
- Des Weiteren betrachte den Operator „+“:
  - Bezeichnet Addition zwischen den numerischen Typen
  - String-Verkettung zwischen Strings

Operanden von + bestimmen Ergebnistypen

- Z.B. int + int -> int etc.
- Ausnahme:
  - byte + byte -> int
  - short + short -> int

- Des Weiteren 15 + 25.1
  - ist in Java erlaubt, aber es muss eine Regel her, die das Ergebnis bestimmt.

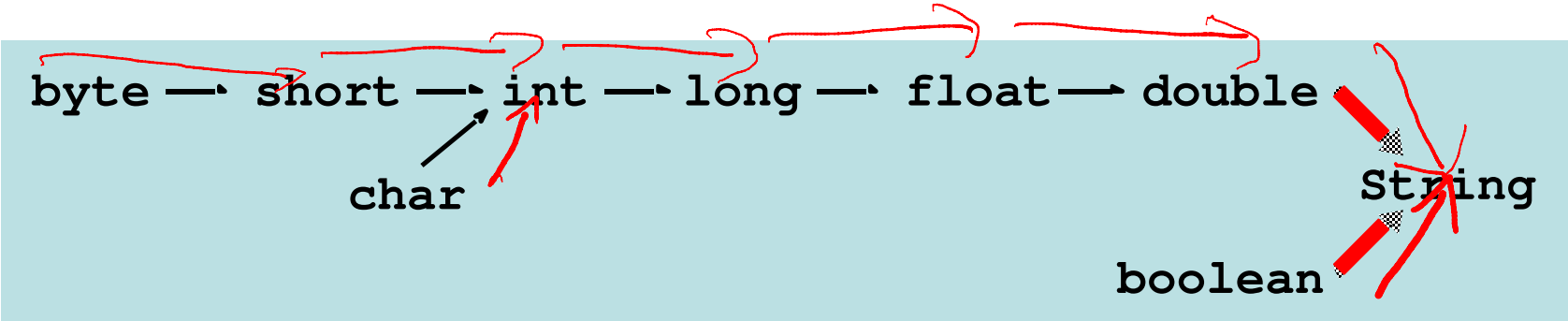
- Es wird kein neuer Operator int + float eingeführt, sondern:



technische universität dortmund
Dr. Lars Hildebrand

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 in Java(!):



Besonderheit bzgl. des Pfeils (-> `String`)

- 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!

- 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

Zwischenstand

Dr. Lars Hildebrand

- Variablen
  - Bezeichner, Datentyp, Speicherort, Wert
- Zuweisungen
  - Zuweisungen müssen typverträglich sein
- (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 11.11.2011, 08:30