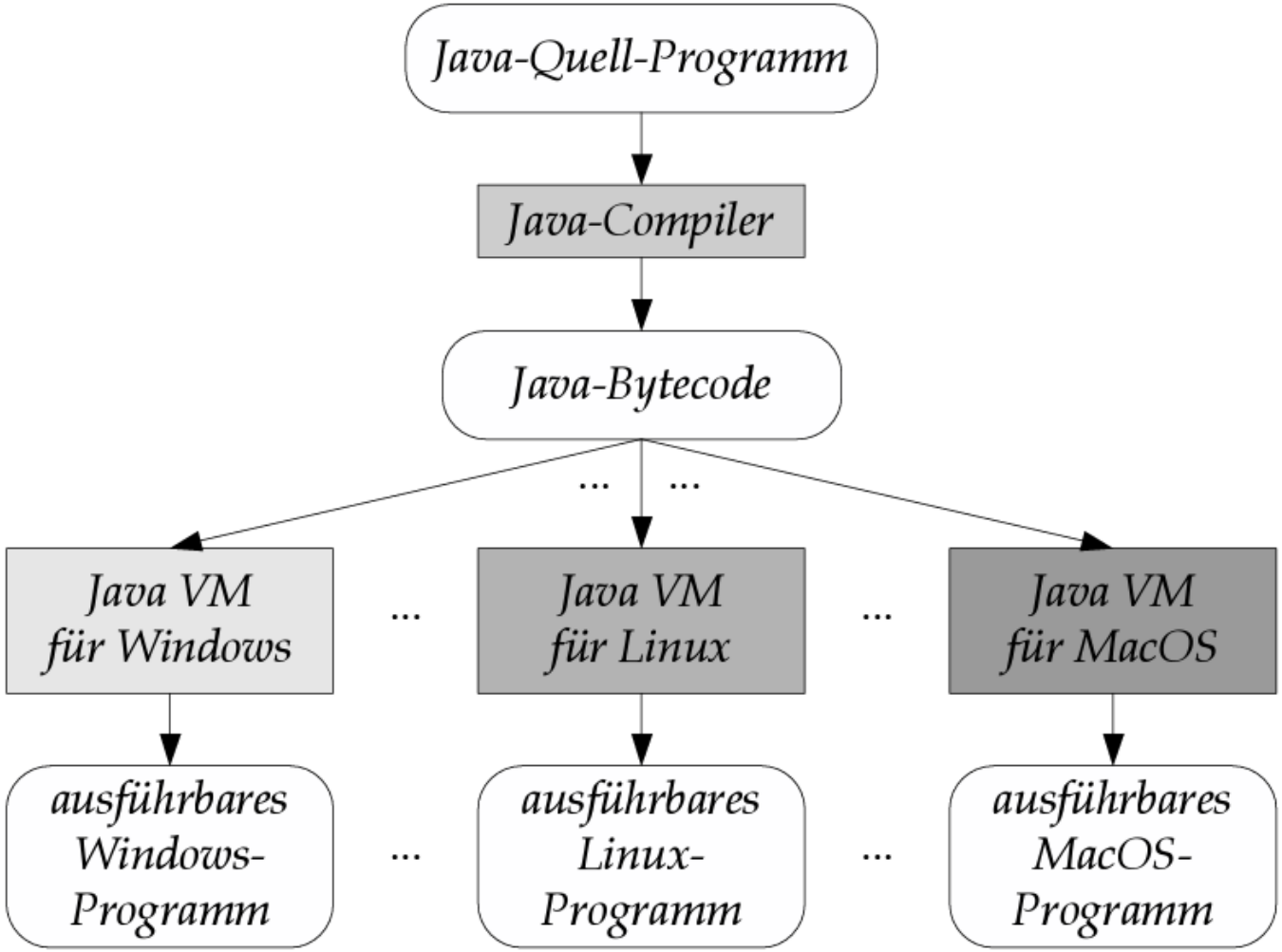
Javaprogrammierung mit NetBeans

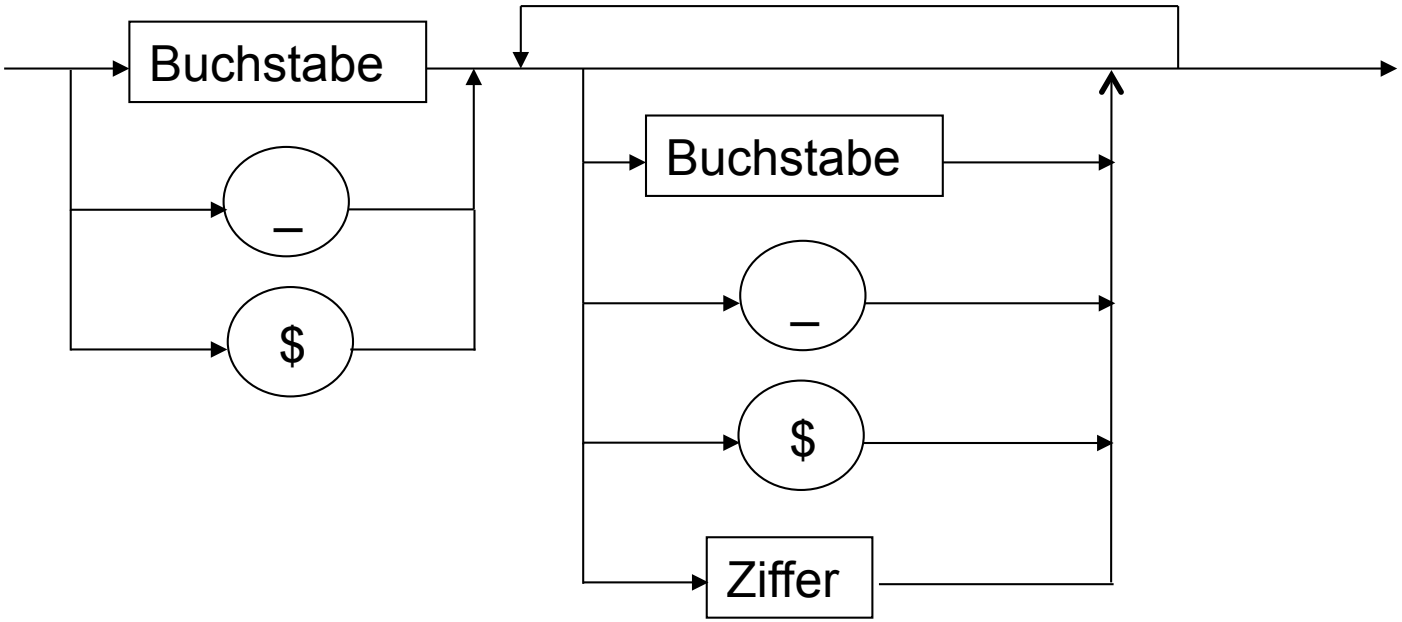
Variablen, Datentypen, Methoden

Programmieren



Java – Bezeichner

Bezeichner:



Groß- und Kleinbuchstaben werden strikt unterschieden.
Schlüsselwörter dürfen nicht als Bezeichner verwendet werden.

Dokumentation - Kommentare

Kommentare innerhalb einer Programmzeile werden mit `//` gekennzeichnet. Der nachfolgende Text wird von Compiler nicht ausgewertet und dient nur zur Dokumentation für den Programmierer.

```
Beispiel: x = y + 3 // Wertzuweisung
```

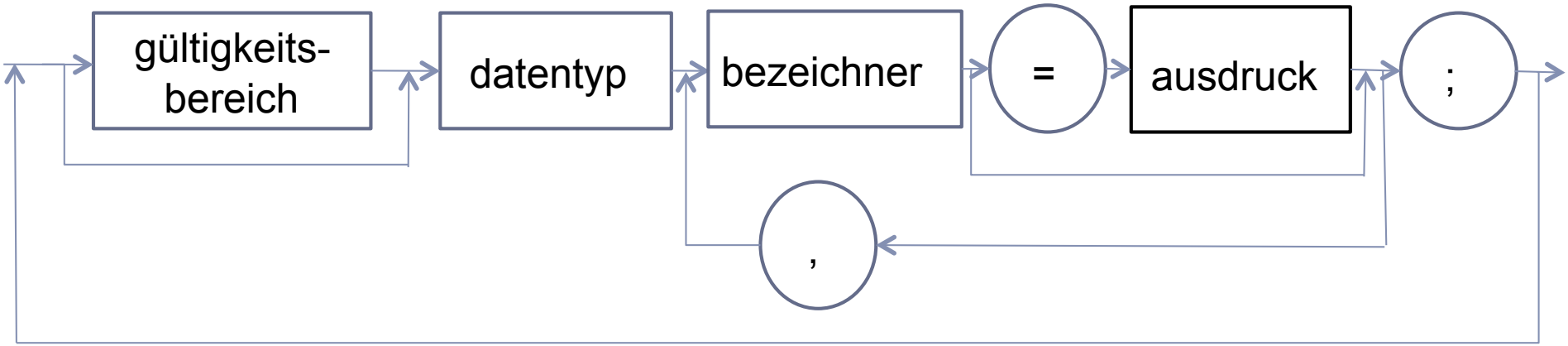
Mehrzeilige Kommentare werden in `/* .. */` eingeschlossen. In den Kommentarbereichen können Informationen abgelegt werden, die anschließend vom Java-Doc-System zur Dokumentationserstellung verwendet werden können.

Beispiel: /Quelle: Java lernen mit BlueJ/

```
/**
 * Ein Kreis, der manipuliert werden kann und sich selbst auf einer Leinwand
 * zeichnet.
 *
 * @author Michael Kölling und David J. Barnes
 * @version 2008.03.30
 */
```

Variablendeklaration

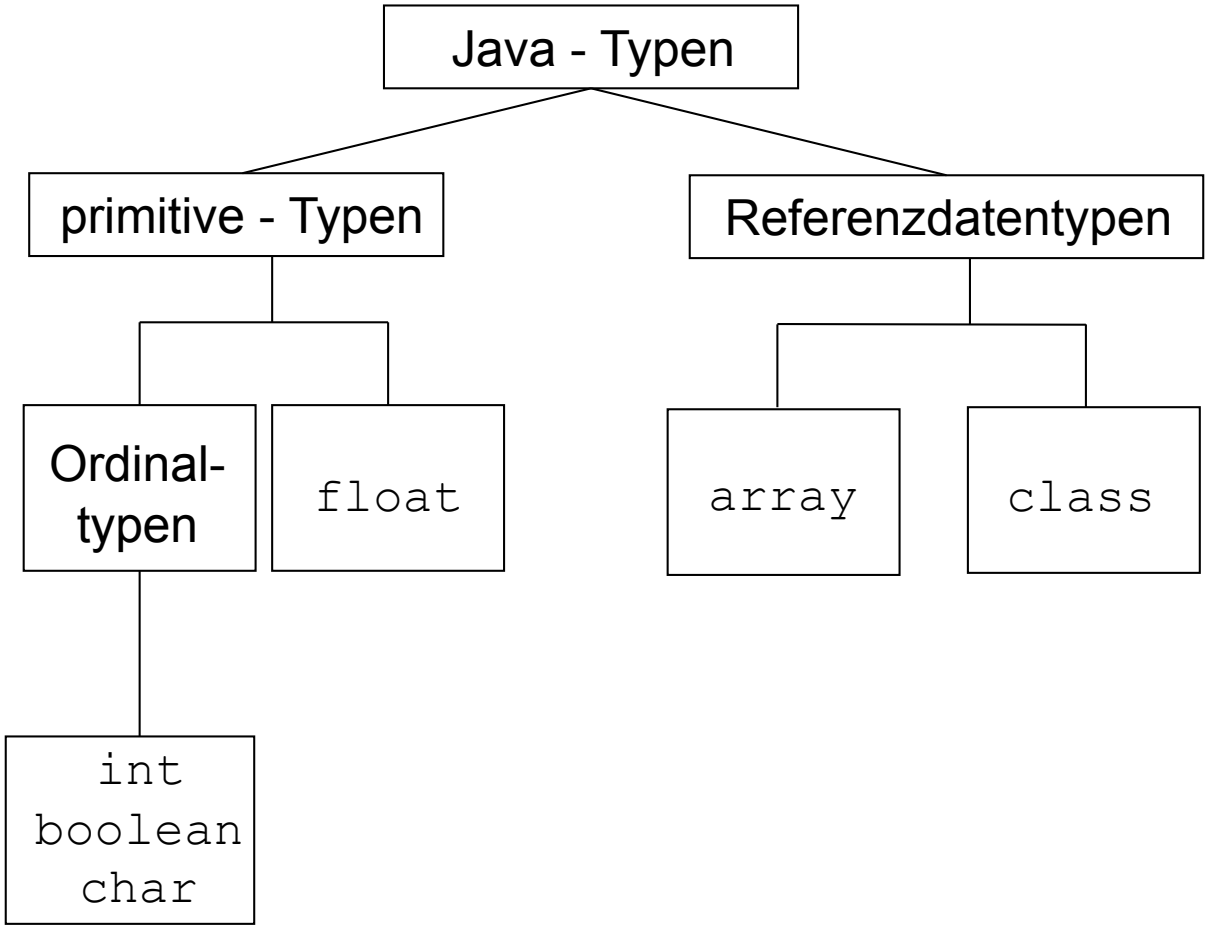
Variablendeklaration und Initialisierung



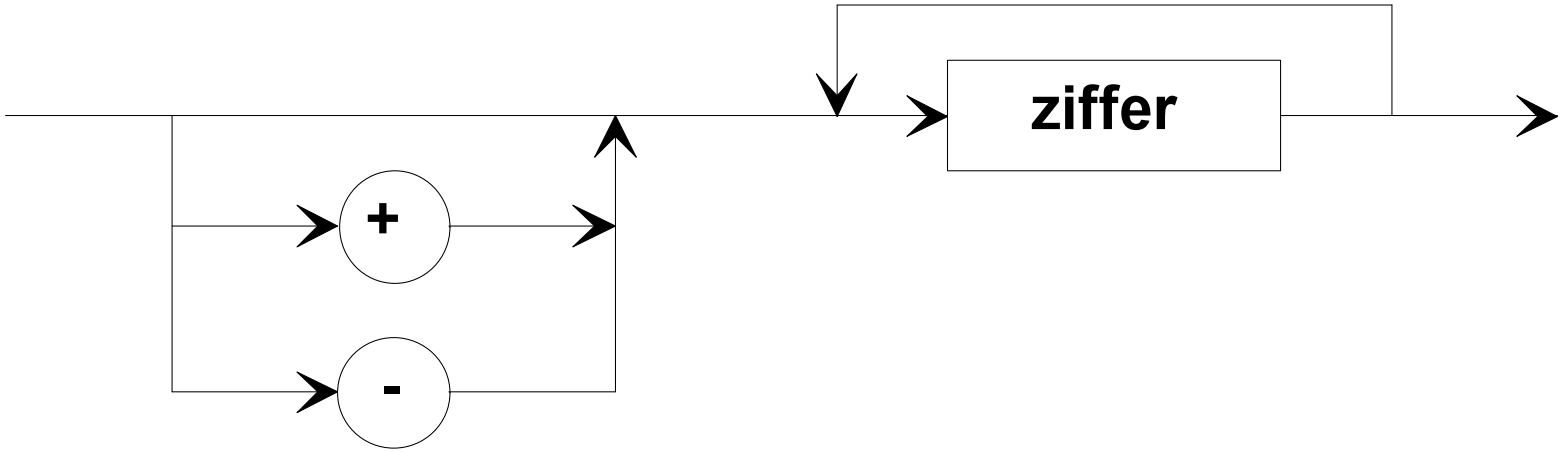
Beispiele:

```
public int i;
private String name;
double zahl = 9.8;
```

Typ-Terminologie in Java



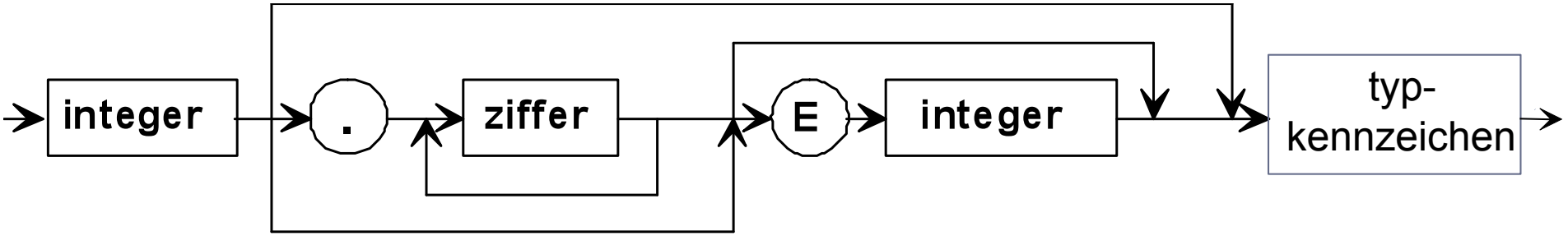
ganzzahlige Datentypen



Java-Implementierung

Typ	Beschreibung	Größe	Kenn-zeichen
byte	Wertebereich: -128 … 127	1 Byte	
short	Wertebereich: -32768 … 32767	2 Byte	
int	Wertebereich: -2147483648 … 2147483647	4 Byte	
long	Wertebereich: -9223372036854775808 … 9223372036854775807	8 Byte	L

Datentypen für Gleitkommazahlen



Java-Implementierung

Typ	Beschreibung	Größe	Kenn-zeichen
float	∗ einfachgenaue Gleitkommazahl ∗ Wertebereich : ± (~1.5*10^-45 ... ~3.4*10^38) ∗ Genauigkeit : 7 - 8 Stellen	4 Byte	f, F
double	∗ doppeltgenaue Gleitkommazahl ∗ Wertebereich : ± (~5.0*10^-324 ... 1.7*10^308) ∗ Genauigkeit : 15 - 16 Stellen	8 Byte	d, D

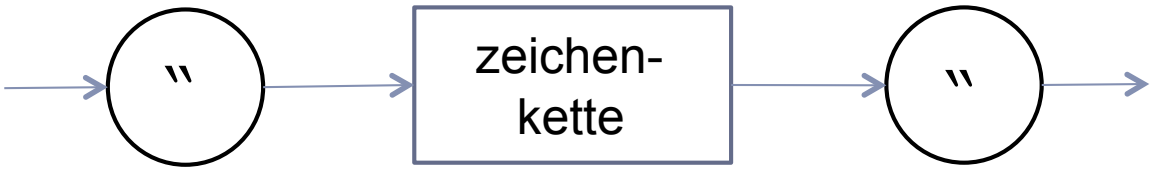
Für Konstanten gilt: es muss mindestens der Dezimalpunkt, das E oder das Typkennzeichen angegeben werden, damit die Zahl als Gleitkommazahl interpretiert wird.

Datentypen für Zeichen - Character

Java-Implementierung

Typ	Beschreibung	Größe
char	Word-Zeichen, angeordnet entsprechend des Unicode-Zeichensatzes. Die ersten 256 Zeichen entsprechen dem ANSI-Zeichensatz	2 Byte

Datentypen für Zeichenketten - String



Typ	Beschreibung
String	**Objekte** einer Klasse String

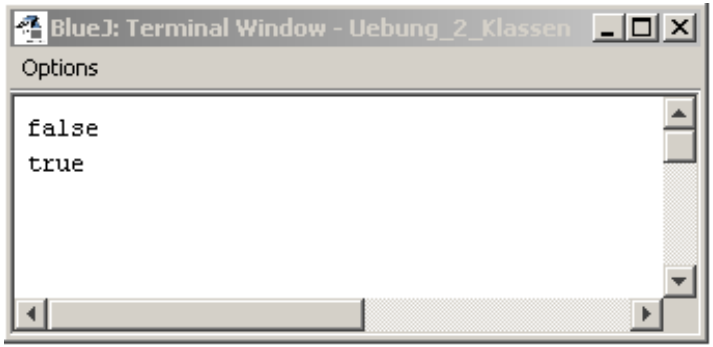
Zeichenketten werden in Java nicht als Werte eines speziellen einfachen Datentyps, sondern als Objekte eines speziellen Klasse String behandelt.

(Schreibweise für String beginnend mit Großbuchstaben beachten!)

Datentypen für Wahrheitswerte - boolean

Zur Verarbeitung von Wahrheitswerten (z.B. dem Ergebnis von Vergleichen) gibt es den Datentyp boolean. Variablen dieses Datentyps können nur die Wahrheitswerte true oder false annehmen.

```
public void vergleich2()
    {
        int a = 3;
        int b = 5;
        boolean w , w1;
        w1 = true;
        w = a > b;
        System.out.println(w);
        System.out.println(w1);
    }
```



Binäre arithmetische Operatoren

Operator	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Divisionsrest bei ganzzahliger Division

Der Datentyp eines Ausdrucks ist von den Datentypen der Operanden abhängig.

Sind bei der *Division* beide Operanden ganzzahlig, so ist das Ergebnis auch ganzzahlig. Ist mindestens einer der Operanden eine Gleitpunktzahl, so ist das Ergebnis der Division auch eine Gleitpunktzahl.

Bestimmung des Ergebnistyps

1. Es wird geprüft, ob einer der Operanden vom typ double ist – ist dies der Fall, so ist der Ergebnistyp double. Die Operation wird im Bereich double ausgeführt.
2. Falls nicht, prüft der Compiler, ob einer der Operanden vom Typ float ist – ist dies der Fall, so ist der Ergebnistyp float. Die Operation wird im Bereich float ausgeführt.
3. Falls nicht, prüft der Compiler, ob einer der Operanden vom Typ long ist – ist dies der Fall, so ist der Ergebnistyp long. Die Operation wird im Bereich long ausgeführt.
4. Trat keiner der Fälle 1 – 3 ein, so ist der Ergebnistyp der Operation auf jeden Fall int. Die Operation wird im Bereich int ausgeführt.

Wird ein andere Ergebnistyp gewünscht, so muss dies durch Typumwandlung (Voranstellen der Typs) erzwungen werden.

Vergleichsoperatoren

Zur Formulierung logischer Ausdrücke werden Vergleichsoperatoren verwendet. Das Ergebnis eines logischen Ausdrucks ist immer der Wahrheitswert wahr oder falsch.

Vergleichsoperatoren:

Operator	Beispiel	liefert genau dann `true`, wenn ...
>	a > b	... a größer als b ist
>=	a >= b	... a größer als oder gleich b ist
<	a < b	... a kleiner als b ist
<=	a <= b	... a kleiner als oder gleich b ist
==	a == b	... a gleich b ist
!=	a != b	... a ungleich b ist

/Quelle: RATZ07, S. 72/

Logische Operatoren

Zur Verknüpfung logischer Ausdrücke werden logische Operatoren verwendet.

Operator	Beispiel	Funktion
&	a & b	Verknüpft a und b durch ein logisches Und
&&	a && b	Verknüpft a und b durch ein logisches Und (nur bedingte Auswertung von b)
|	a | b	Verknüpft a und b durch ein logisches Oder
||	a || b	Verknüpft a und b durch ein logisches Oder (nur bedingte Auswertung von b)
^	a ^ b	Verknüpft a und b durch ein logisches exklusives Oder
!	! a	Negiert a

/Quelle: RATZ07, S. 73/

Beispiel:

```
if ((a>=0) && (a<=20))
  …
```

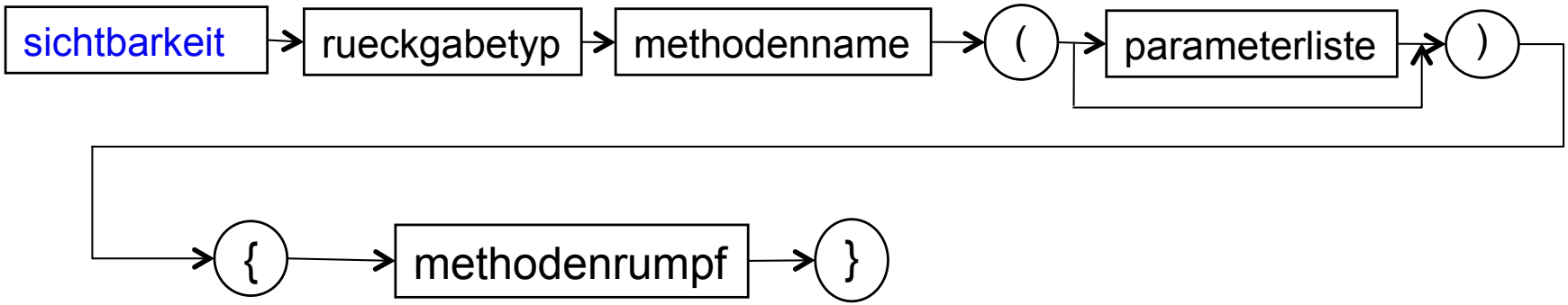
Vorrangregeln

Bezeichnung	Operator	Priorität
Komponentenzugriff bei Klassen	.	15
Komponentenzugriff bei Feldern	[ ]	15
Methodenaufruf	( )	15
Unäre Operatoren	++, --, +, -, ~, !	14
Explizite Typkonvertierung	( )	13
Multiplikative Operatoren	*, /, %	12
Additive Operatoren	+, -	11
Schiebeoperatoren	<<, >>, >>>	10
Vergleichsoperatoren	<, >, <=, >=	9
Vergleichsoperatoren (Gleichheit/Ungleichheit)	==, !=	8
bitweises bzw. logisches Und	&	7
bitweises exklusives Oder	^	6
bitweises bzw. logisches Oder	|	5
logisches Und	&&	4
logisches Oder	||	3
Bedingungsoperator	? :	2
Zuweisungsoperatoren	=, +=, -=, usw.	1

/Quelle: RATZ07, Seite 75/

Deklaration von Methoden

Methodendeklaration:

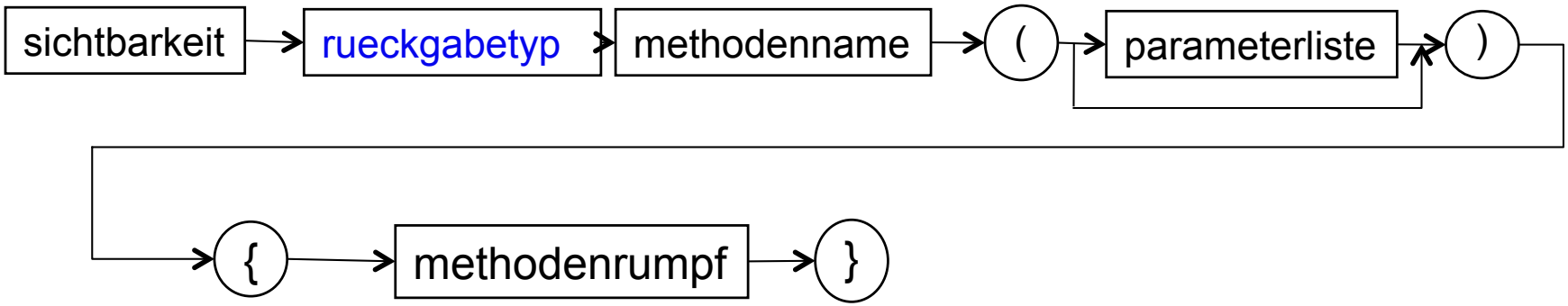


Sichtbarkeit:

Die Sichtbarkeit legt fest, in welchen Bereichen auf die Methode zugegriffen werden kann. Wir verwenden public – Zugriff von außen ist zulässig oder private – der Zugriff ist nur innerhalb des Objektes zulässig.

Deklaration von Methoden

Methodendeklaration:



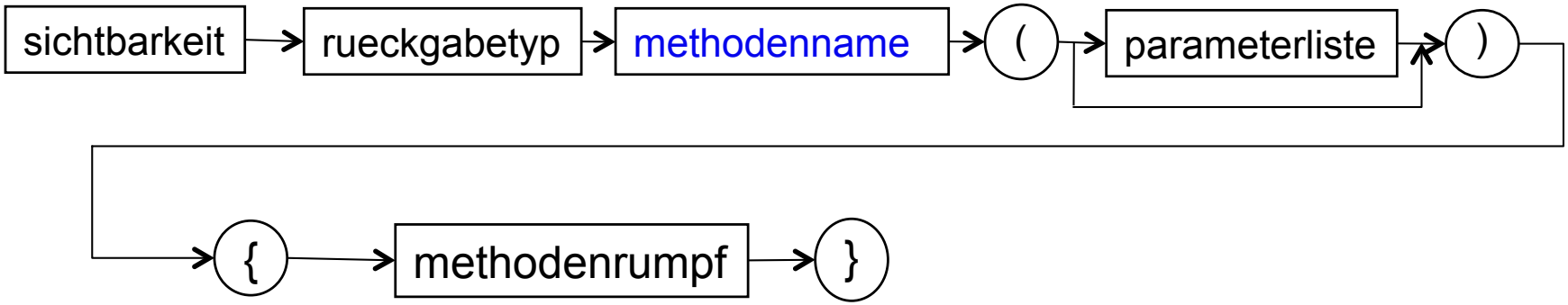
Rückgabetyp:

Der Rückgabetyp legt fest, welchen Datentyp das Ergebnis der Methode hat. Das Ergebnis wird über die Rückgabeanweisung return ausgegeben.

Gibt eine Methode keinen Wert zurück, so hat sie den Rückgabetyp void. In diesem Fall ist keine return Anweisung erforderlich. Wird sie verwendet, so führt ihre Abarbeitung zum Abbruch der Methode.

Deklaration von Methoden

Methodendeklaration:

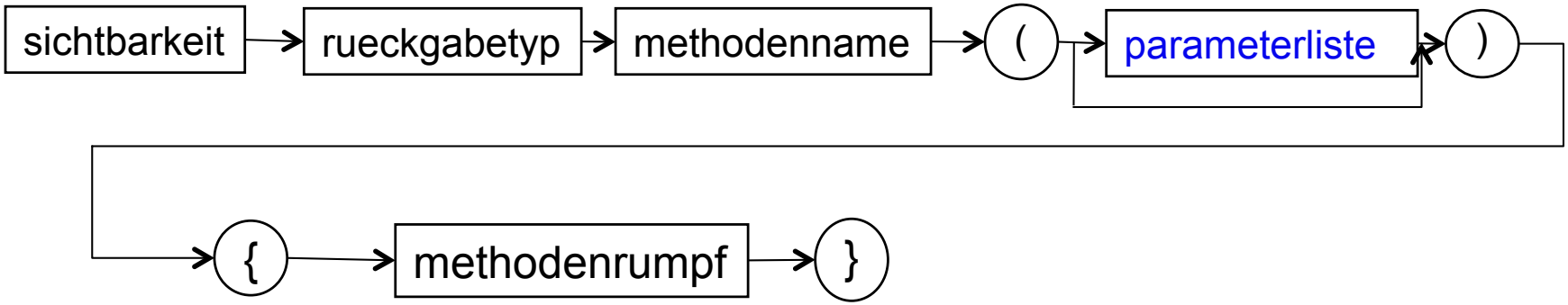


Methodenname:

Der Methodenname muss ein gültiger Bezeichner sein. Über diesen Namen wird die Funktionalität des Objektes aufgerufen.

Deklaration von Methoden

Methodendeklaration:



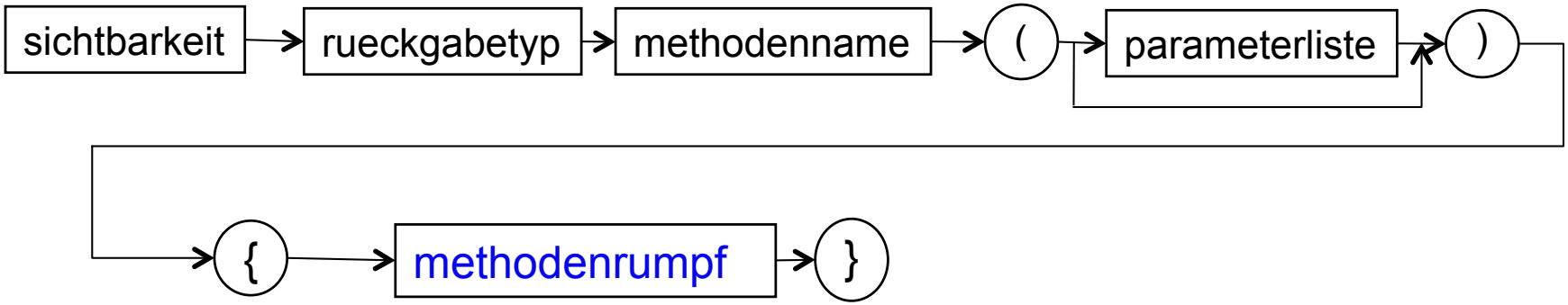
Parameterliste:

Die Parameterliste ist eine Kommaliste von Variablendeklarationen. Die darin deklarierten Variablen werden als formale Parameter bezeichnet. Sie werden beim Aufruf der Methode initialisiert und haben innerhalb der Methode den Status von lokalen Variablen. Werden mehrere Parameter deklariert, so muss für jeden Parameter eine Typenbezeichnung angegeben werden. Die Parameterliste kann leer sein.

Die beim Aufruf verwendeten Parameter müssen zuweisungsverträglich zu denen in der Parameterliste sein.

Deklaration von Methoden

Methodendeklaration:

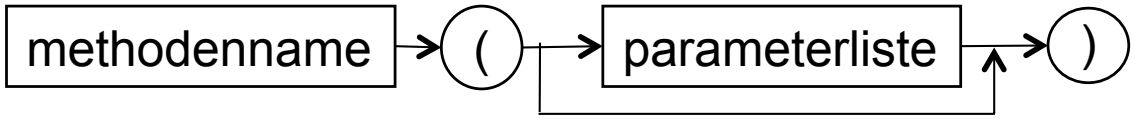


Methodenrumpf:

Der Methodenrumpf beschreibt die Funktionalität einer Methode – den Algorithmus.

Aufruf von Methoden

Methodenaufruf:



Beim Aufruf einer Methode wird der Methodenname, gefolgt von einer Parameterliste angegeben. Diese Parameter werden als Aktualparameter bezeichnet. Sie müssen zuweisungsverträglich zu den formalen Parametern sein.

Arbeiten mit graphischer Oberfläche

-Inhalte von Textfeldern und Labels auf der Oberfläche werden als String verwaltet
-Zugriff auf Inhalt über

```
<komponentenname>.getText();
```
→ liefert String

-Umwandeln von String in andere Datentypen
http://www.dpunkt.de/java/Die_Sprache_Java/Die_Sprachelemente_von_Java/38.html

```
Integer.parseInt(String);
Double.parseDouble(String);
```

-Umwandeln anderer Datentypen in String für die Ausgabe auf der Oberfläche:

```
String.valueOf(wert);
```

-Zugriff auf Komponente zum Schreiben über

```
<komponentenname>.setText(String);
```
→ String wird übergeben