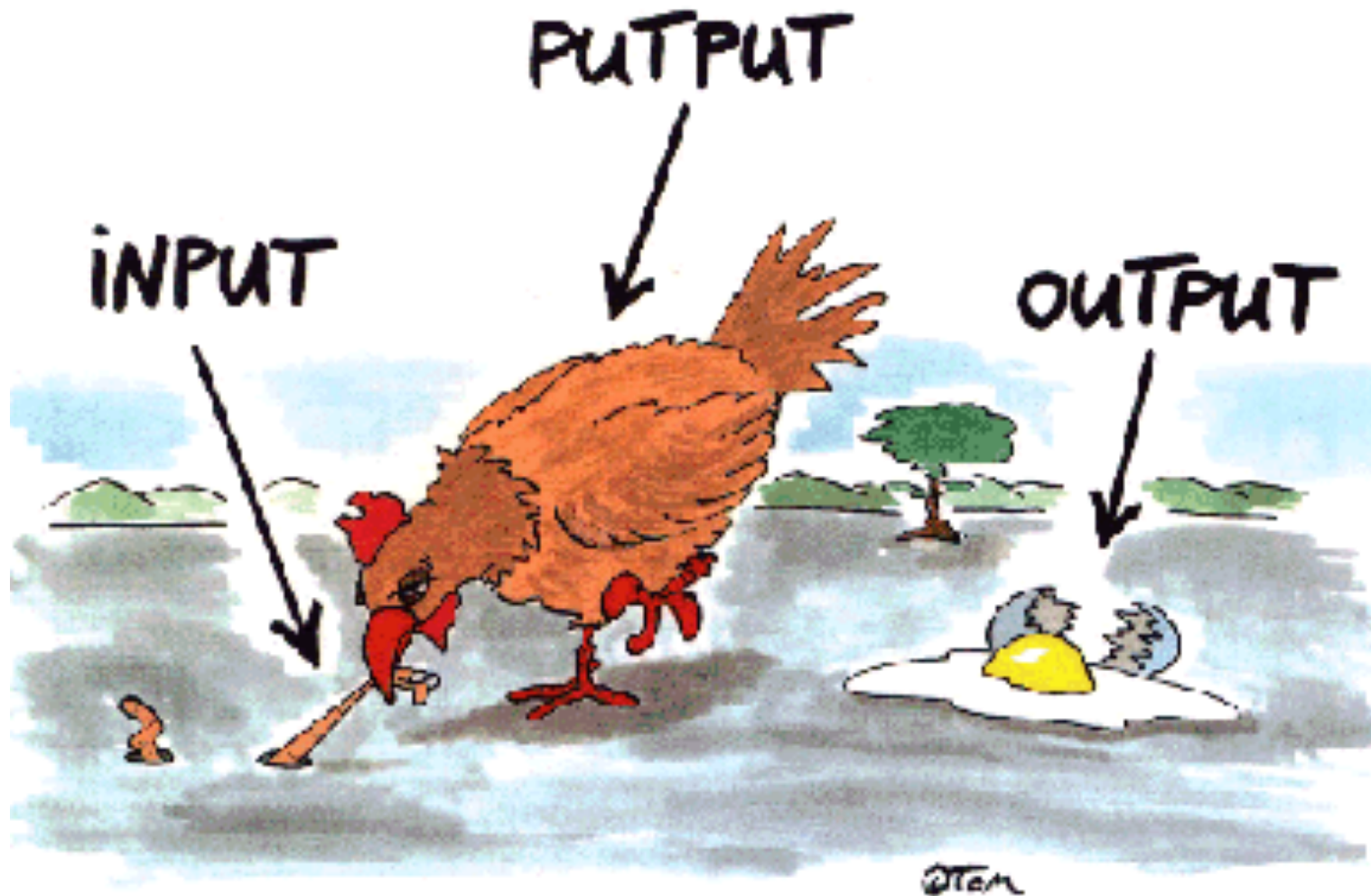
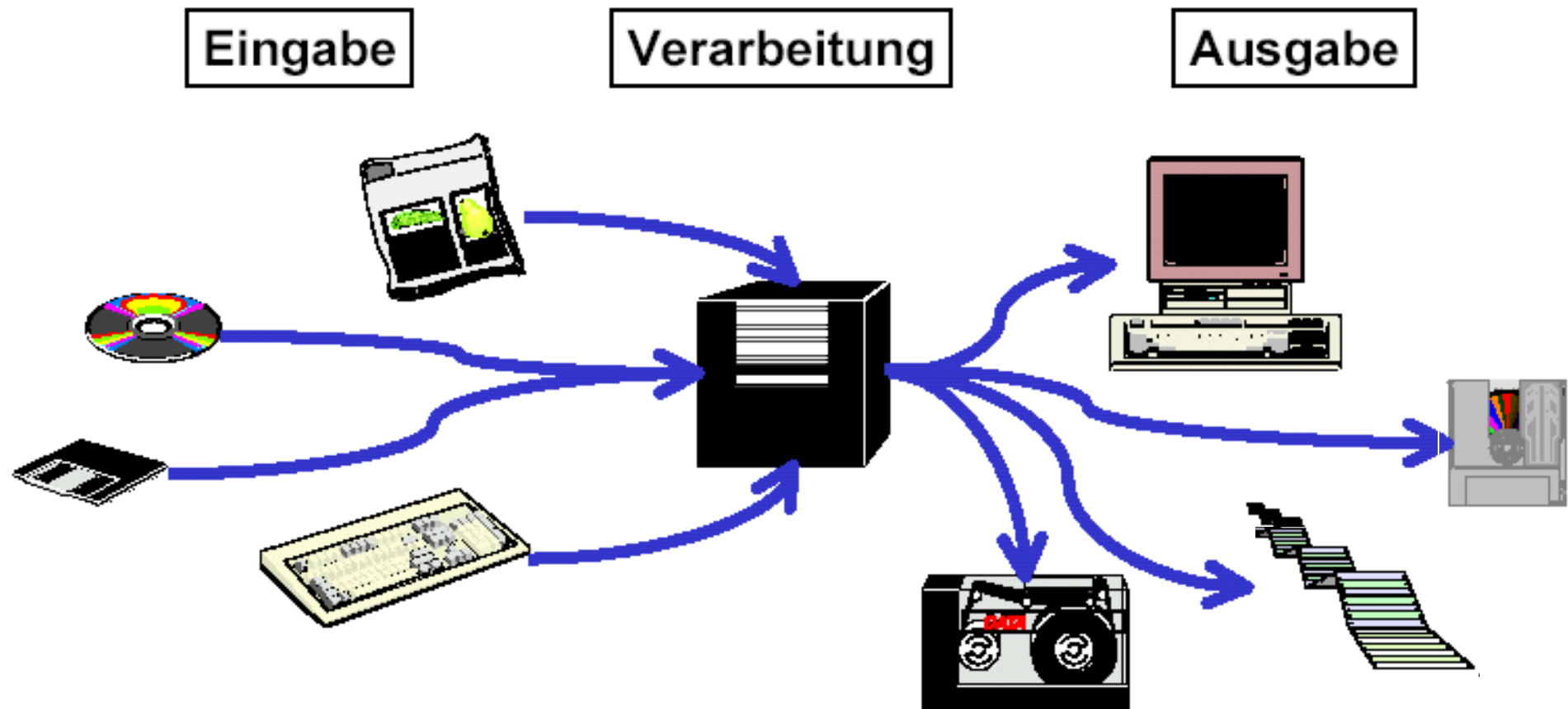


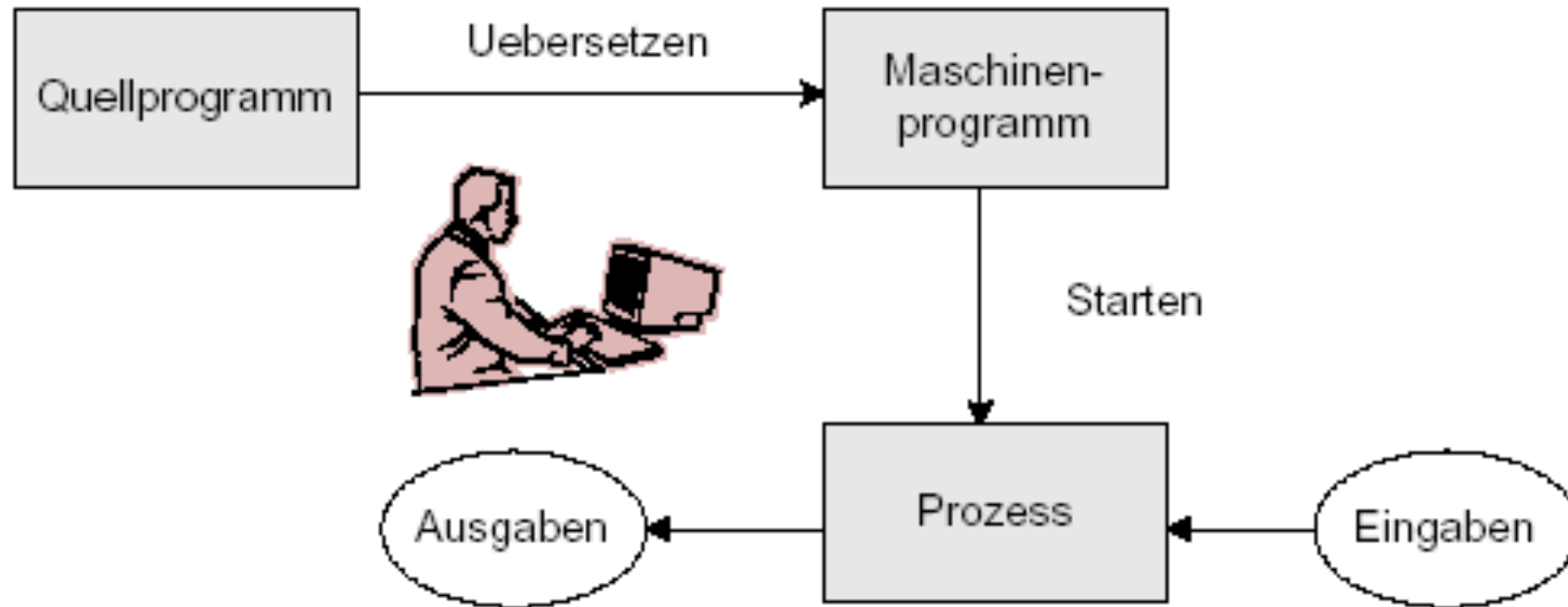
- Programmieren gehört zu grundlegenden Fertigkeiten eines Wirtschaftsinformatikers
- Bestandteil des gesamten Studiums
- Lernziele von Programmierung I
 - Einstieg in die Programmierung
 - Einübung algorithmischen Denkens
 - Strukturierung und Modularisierung von Problemlösungen
 - Sicherer Umgang mit Entwicklungsumgebung
 - Einstieg in die objektorientierte Programmierung

Grundprinzip EVA: Input – Output



Das Grundprinzip: EVA





Programm:

- Eine zur Lösung einer Aufgabe vollständige Anweisung an den Computer
- Der Vorgang zur Erstellung einer derartigen Anweisung heißt **Programmieren**.

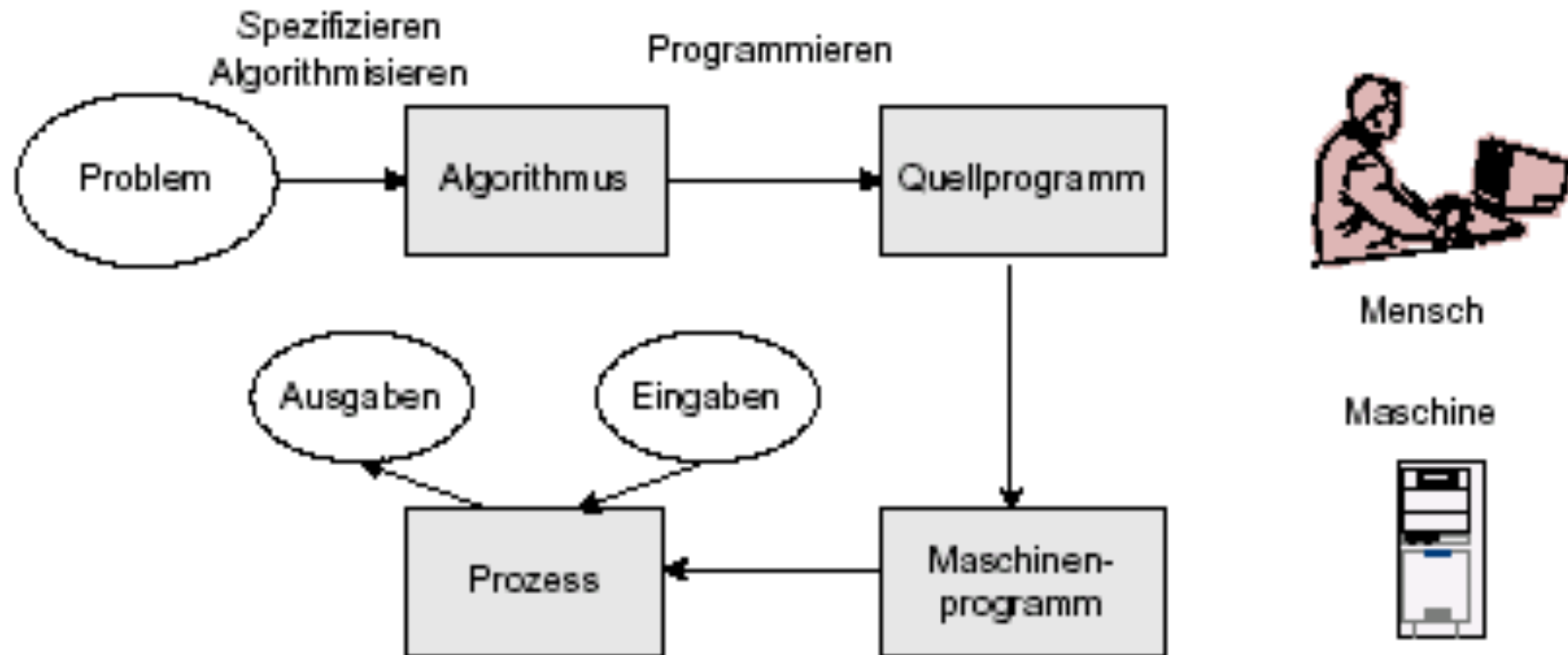
Quelle: Schlichter, Einführung in die Informatik

- Verfahrensvorschrift zur Lösung eines Problems
 - Exakt
 - vollständig formuliert
 - schrittweise ausführbar
 - effektiv ausführbar
 - endlich
- Formulierung kann in natürlicher oder formaler Sprache vorliegen
- Die Ausführung kann durch Menschen oder eine Maschine erfolgen

- Bestimme das Alter der ältesten Person im Raum
 - Gehe zur ersten Person;
 - Frage Person nach dem Alter;
 - Merke das Alter;
 - Wiederhole bis alle Personen gefragt sind
 - gehe zur nächsten Person;
 - frage nach dem Alter;
 - wenn das Alter größer als das gemerkte Alter, dann merke Dir das neue Alter;
 - Das Alter der ältesten Person ist "gemerktes Alter";

- wesentliche Elemente
 - Sequenz von Anweisungen
 - Variablen zum Speichern von Daten
 - Wertzuweisungen
 - Wiederholung (Schleife)
 - Alternativen
- Damit können alle Algorithmen beschrieben werden
- wesentliche Fragen
 - Darstellung
 - Korrektheit
 - Ressourcenverbrauch

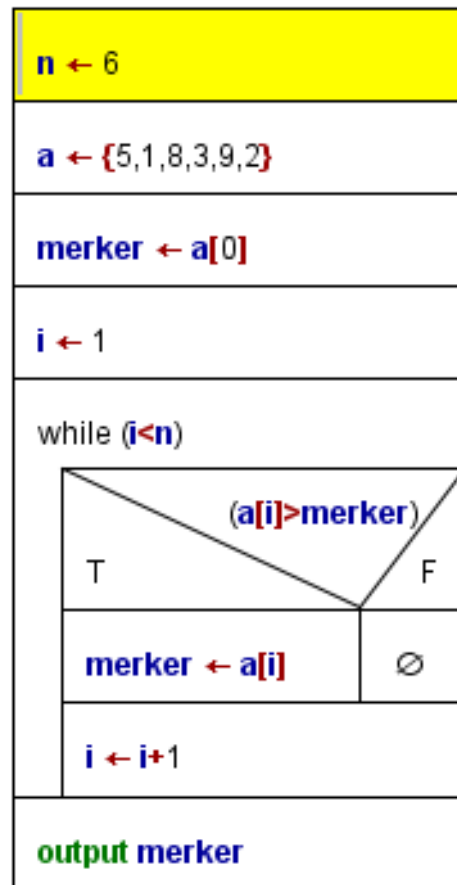
- Eine **Variable** ist ein benannter Behälter für einen Wert
- Änderung des Wertes über Wertzuweisungen
 - $x = 5$
 - $y = x + 1$ // y enthält nun den Wert 6
 - $x = 2 * y$ // x enthält nun den Wert 12
 - Dies ist nicht zu verwechseln mit dem mathematischen „=“, dort wäre so etwas ein Widerspruch!
 - Manchmal schreibt man deswegen auch $x \leftarrow 5$ (hat sich aber nicht durchgesetzt)
- In Java muss eine Variable vor der Verwendung erst definiert werden.



Quelle: Schlichter, Einführung in die Informatik

Darstellung als Struktogramm

Maximum



Problem: Sei n gegeben

Bestimme die größte Zahl der n
Zahlen a_0, a_1, \dots, a_{n-1}

Quelle: <http://structorizer.fisch.lu/>

Ablaufbeispiel (Schreibtischtest)

- Gegebene Folge 5, 1, 8, 3, 9, 2

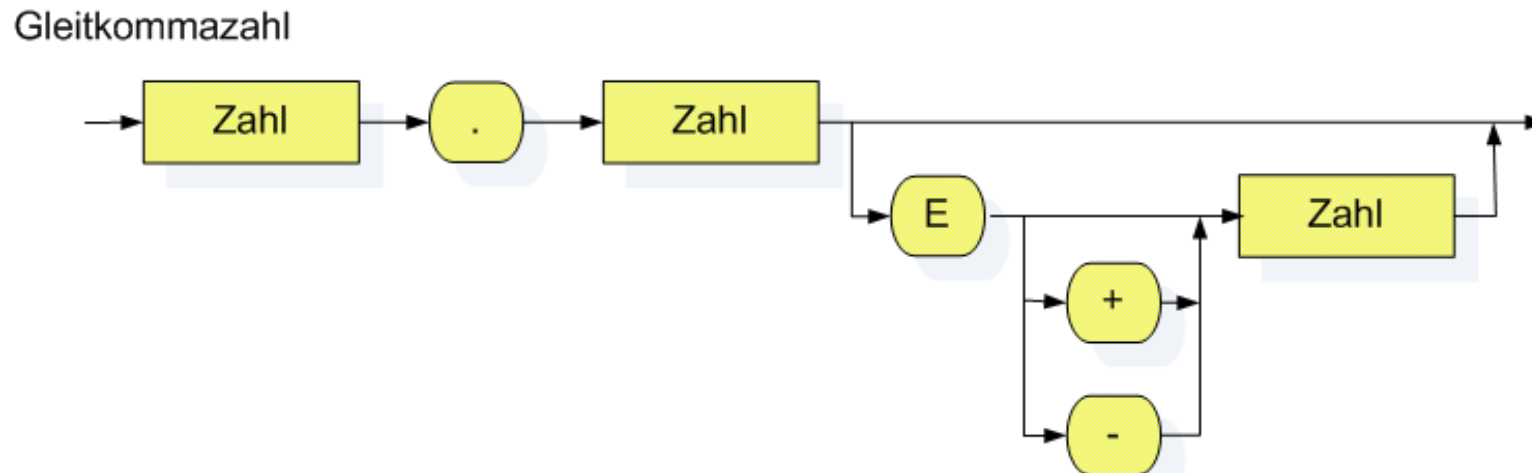
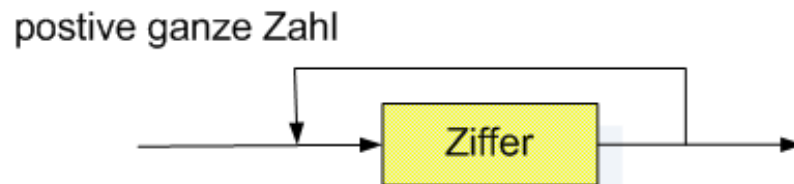
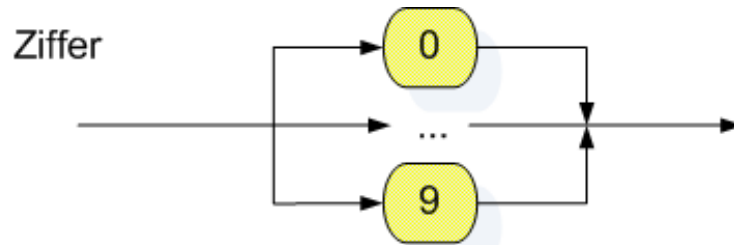
- | i | a_i | merker |
|---|-------|--------|
| | | 5 |
| 1 | 1 | 5 |
| 2 | 8 | 8 |
| 3 | 3 | 8 |
| 4 | 9 | 9 |
| 5 | 2 | 9 |
| 6 | | |

```
einlesen  $a_0, \dots, a_{n-1}$ ;  
setze merker =  $a_0$ ;  
setze  $i = 1$ ;  
solange (  $i < n$  )  
  führe aus  
    falls  $a_i > \text{merker}$   
      dann setze merker =  $a_i$ ;  
    erhöhe  $i$  um 1;  
gebe merker aus;
```

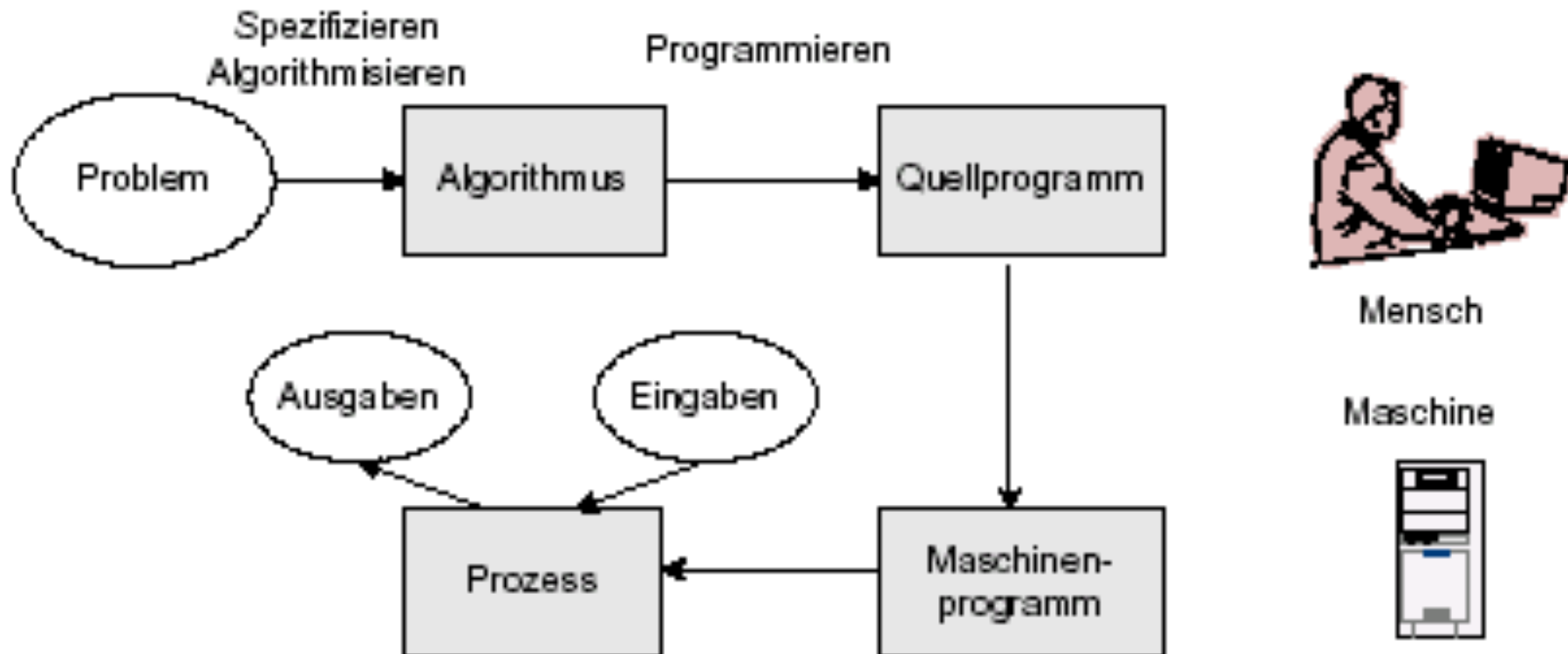
- Die **Syntax** einer Programmiersprache beschreibt den Aufbau ihrer Konstrukte (Anweisungen, Deklarationen) als Folge von Zeichen eines Alphabets.
- Dieser Aufbau wird durch eine Grammatik beschrieben und im Rahmen dieser Vorlesung durch **Syntaxdiagramme** visualisiert.
- Bei Syntaxdiagrammen wird zwischen **Terminalsymbolen** und **Nicht-Terminalsymbolen** unterschieden.
- Für jedes Nicht-Terminalsymbol, gibt es ein eigenes Syntaxdiagramm, welches es als eine Folge von Terminal- und Nicht-Terminalsymbolen darstellt.
- Terminalsymbole werden als abgerundete, Nicht-Terminalsymbole als eckige Kästchen dargestellt.

Quelle: Henning/Vogelsang: Taschenbuch Programmiersprachen

Syntaxdiagramm (Beispiele)



Quelle: Mössenböck

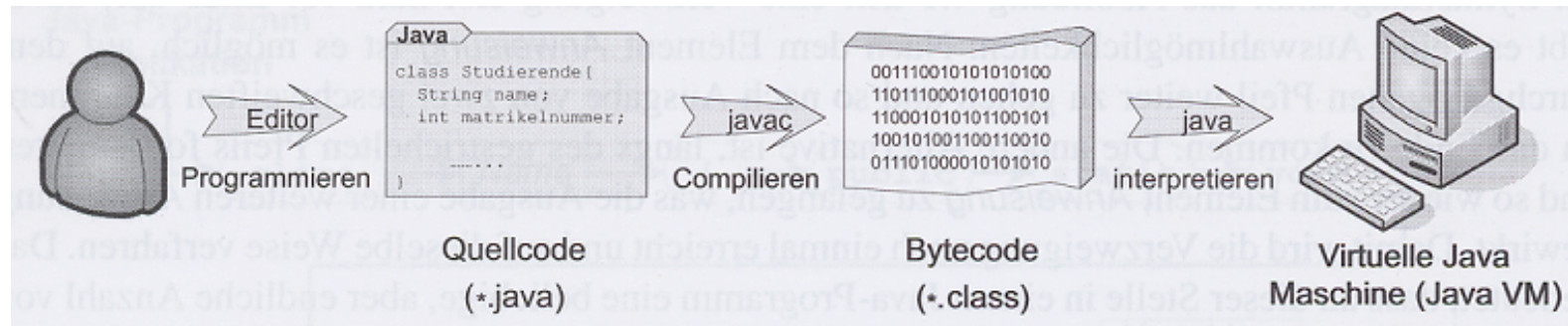


Quelle: Schlichter, Einführung in die Informatik

Programm:

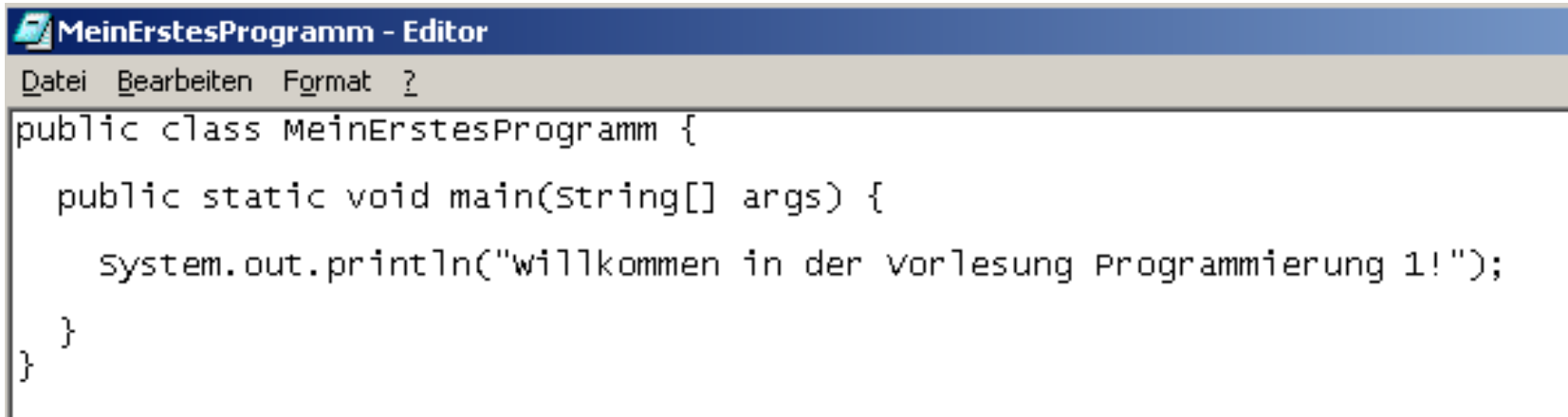
- Eine zur Lösung einer Aufgabe vollständige Anweisung an den Computer
- Der Vorgang zur Erstellung einer derartigen Anweisung heißt **Programmieren**.

Java: Bytecode und virtual Machine



Quelle: Müller, Weichert, Vorkurs Informatik

Erstes Programm in Java



```
MeinErstesProgramm - Editor
Datei Bearbeiten Format ?

public class MeinErstesProgramm {
    public static void main(String[] args) {
        System.out.println("Willkommen in der Vorlesung Programmierung 1!");
    }
}
```

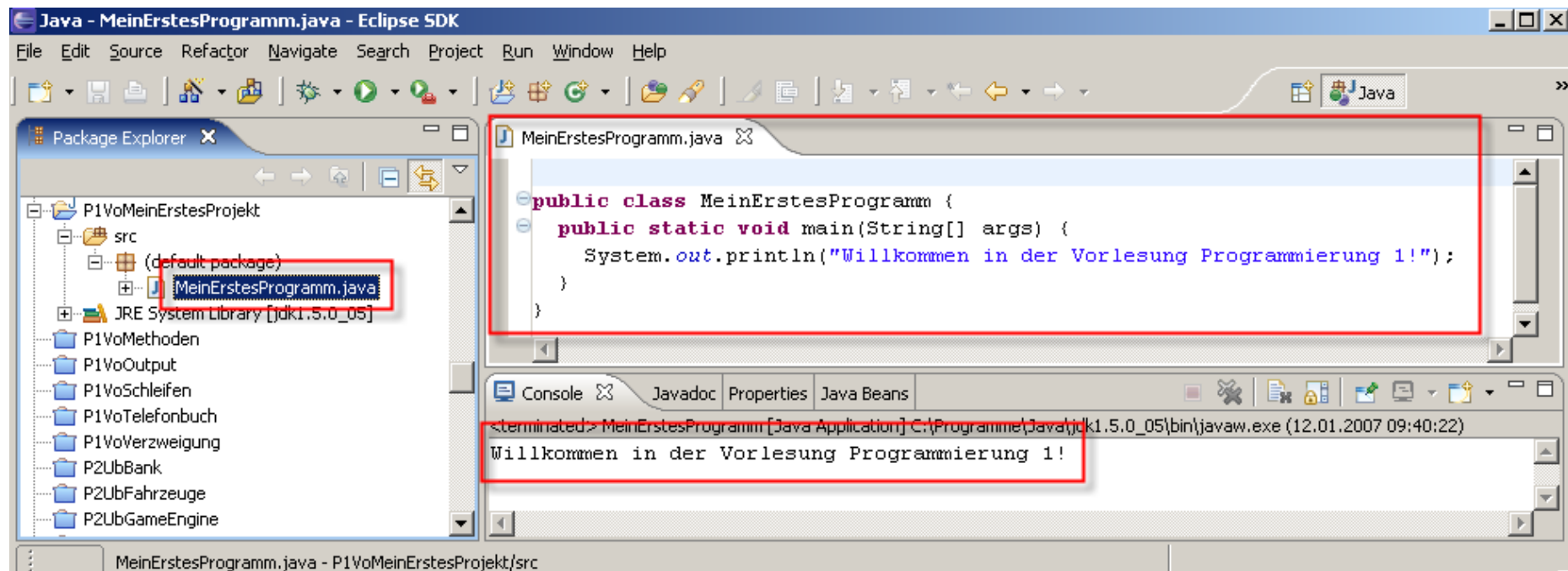


```
Eingabeaufforderung
12.01.2007 08:59 <DIR> .
12.01.2007 08:59 <DIR> ..
12.01.2007 09:14 475 MeinErstesProgramm.class
12.01.2007 08:47 165 MeinErstesProgramm.java
                2 Datei(en) 640 Bytes
                2 Verzeichnis(se), 38.924.636.160 Bytes frei

H:\Java>java MeinErstesProgramm
Willkommen in der Vorlesung Programmierung 1!

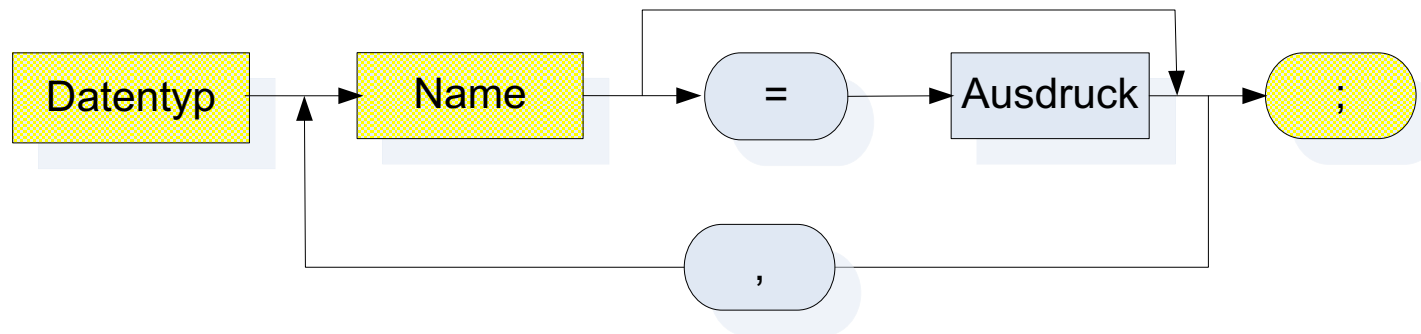
H:\Java>_
```

Erstes Programm in Eclipse



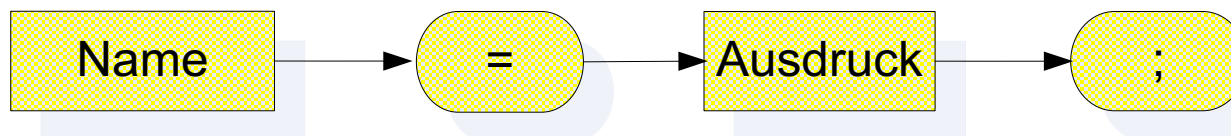
- Eine **Variable** ist ein benannter Behälter für einen Wert
- Änderung des Wertes über Wertzuweisungen
 - $x = 5$
 - $y = x + 1$ // y enthält nun den Wert 6
 - $x = 2 * y$ // x enthält nun den Wert 12
 - Dies ist nicht zu verwechseln mit dem mathematischen „=“, dort wäre so etwas ein Widerspruch!
 - Manchmal schreibt man deswegen auch $x \leftarrow 5$ (hat sich aber nicht durchgesetzt)
- In Java muss eine Variable vor der Verwendung erst definiert werden.

Variablendeklaration



- Beispiel: `int a = 5, b = 6, c;`
 - aber **nicht**: `int x, short y;`

Wertzuweisung



- Beispiel: `c = 5 + a;`

Daten	Typ	Größe	Wertebereich
Ganze Zahlen	byte	8 bit	-2^7 bis $2^7 - 1$ (-128...127)
	short	16 bit	-2^{15} bis $2^{15} - 1$ (-32768...32767)
	int	32 bit	-2^{31} bis $2^{31} - 1$ (-2147483648...2147483647)
	long	64 bit	-2^{63} bis $2^{63} - 1$ (-9223372036854775808... 9223372036854775807)
Fließkommazahlen	float	32 bit	$-3.40282347 \cdot 10^{38}$ bis $3.40282347 \cdot 10^{38}$
	double	64 bit	$-1.79769313486231570 \cdot 10^{308}$ bis $1.79769313486231570 \cdot 10^{308}$
Zeichen	char	16 bit	Unicode Zeichen
boolscher Wert	boolean	8 bit	true/false

- Diese Datentypen sind direkt verfügbar
- Möglichkeit zur Definition eigener Datentypen → folgt später

"Gangnam Style" sprengt YouTube-Zähler

04.12.2014

Der südkoreanische Rapper Psy (36) hat mit dem Erfolg seines Songs "Gangnam Style" die Videoplattform YouTube an ihre Grenzen gebracht.

Empfehlen Die Software, die Abrufe von Videoclips zählt, war technisch auf einen Maximalwert von 2.147.483.647 (32-bit Integer) begrenzt. Man habe nicht gedacht, dass ein Video häufiger angesehen werden würde, räumte die Google-Plattform [in einem Post bei Google+](#) ein. Der Zähler sei nun auf 64-bit Integer aufgerüstet worden.

Diskutieren

Drucken

PDF

URL



^ Oben

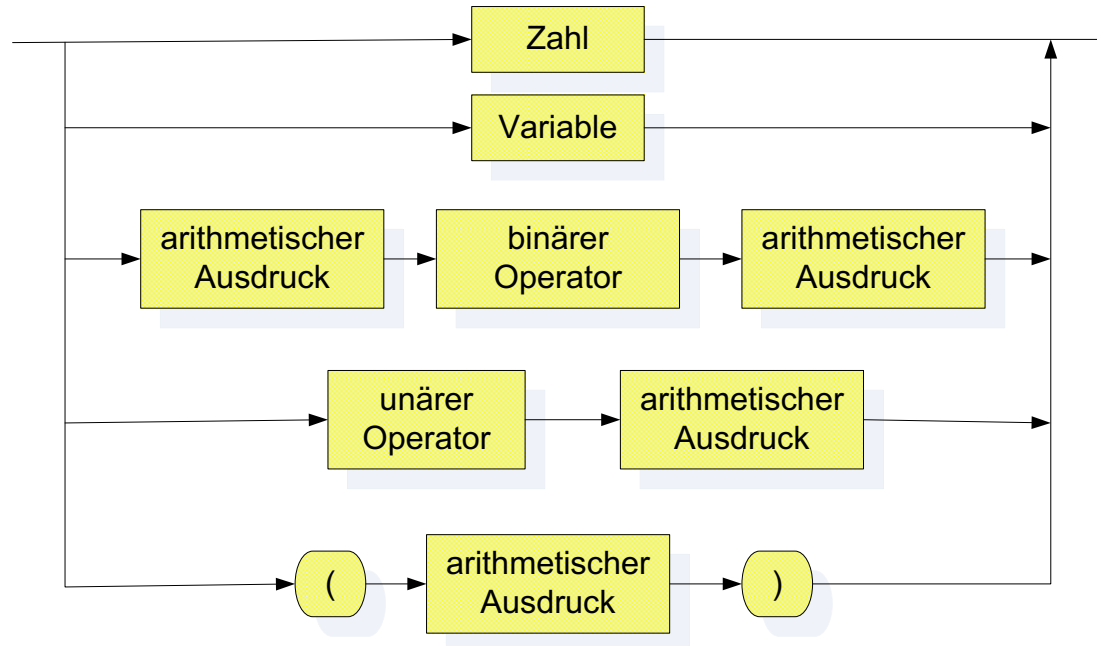


Der Überraschungshit "Gangnam Style" war das erste Video, das bei YouTube die Marke von einer Milliarde Abrufen knackte. Bis Mittwochnachmittag wurde es 2,15 Milliarden Mal angesehen. (dpa/tc)

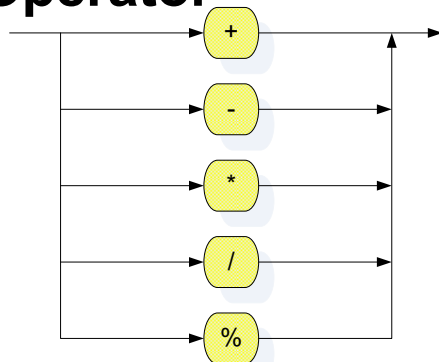
Quelle: <http://www.computerwoche.de/a/gangnam-style-sprengt-youtube-zaehler,3090260>

- Zahlen und Variablen werden durch Operatoren $+$, $-$, $*$, $/$, $\%$ verknüpft
- Regeln für Auswertung
 - Regeln wie in der Mathematik, z.B. „Punkt“ vor „Strich“
 - Klammern zuerst
 - Sonst von links nach rechts
- Beispiele für arithmetischen Ausdruck
 - 5
 - $67 - 16 / 2$
 - $16 - 4 * 2 + 8$
 - x
 - $6 + x$
 - $- 5 * (6 + x)$

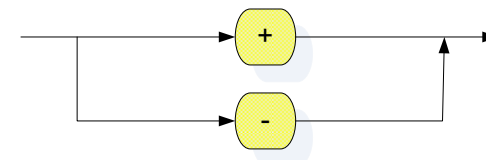
arithmetischer Ausdruck



binärer Operator

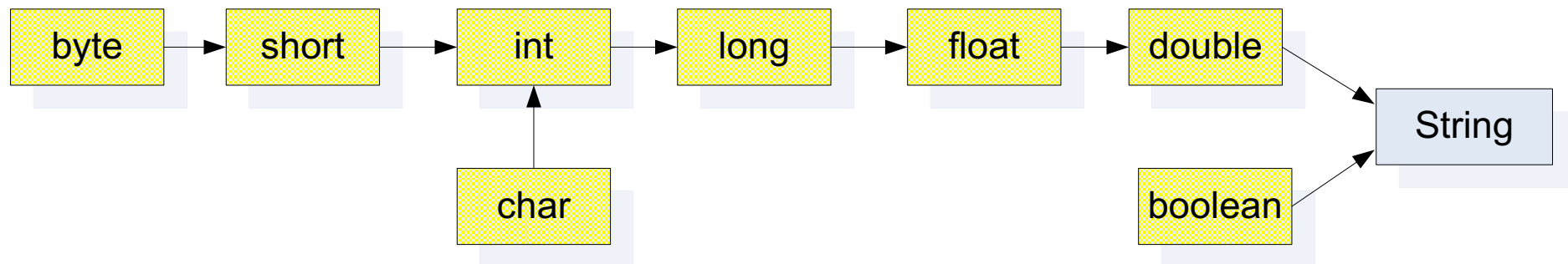


unärer Operator



- Beispiel für Variablenvereinbarung
 - `int a = 5, b = 6, c;`
- Beispiel für Anweisung
 - `c = -(1 + a) * (-b + 2) - b / a + 4 * b % 13 / 2 * 3;`

- **Legende:** Primitive Datentypen



ASCII-Codetabelle, Nummerierung in **Hex**

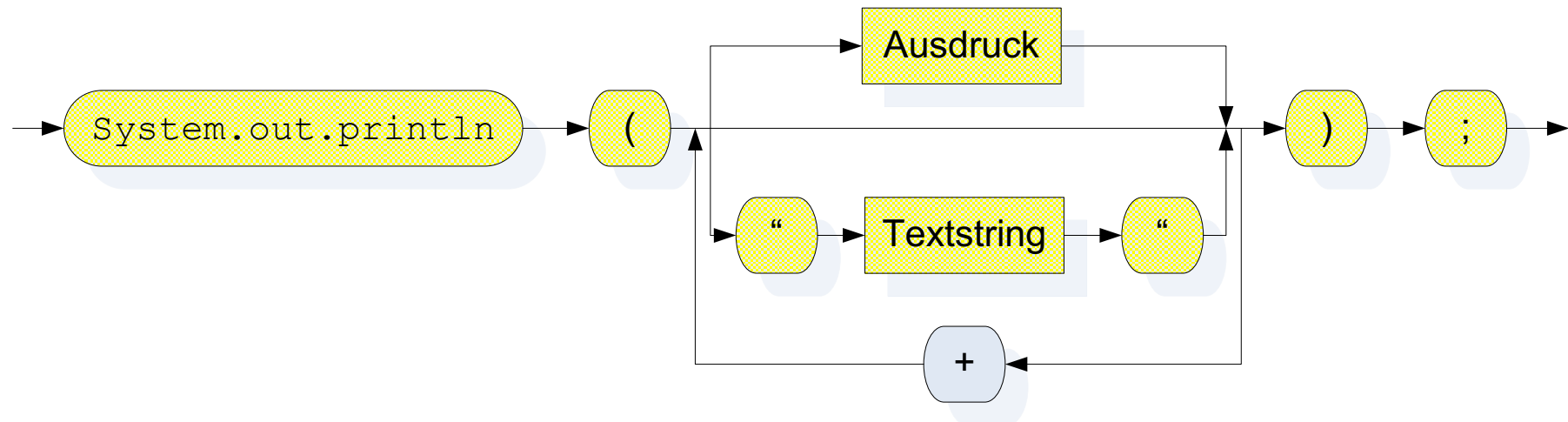
Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Quelle: <http://de.wikipedia.org/wiki/ASCII>

Beispiele:

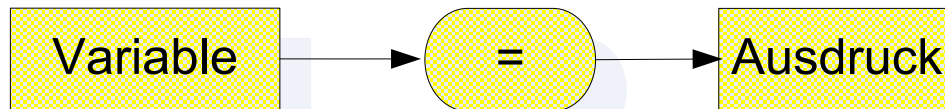
- 'A' wird dargestellt als $41_{\text{Hex}} = 4 \cdot 16 + 1 = 65_{\text{Dez}}$
- 'a' wird dargestellt als $61_{\text{Hex}} = 6 \cdot 16 + 1 = 97_{\text{Dez}}$

Syntax der Ausgabeanweisung

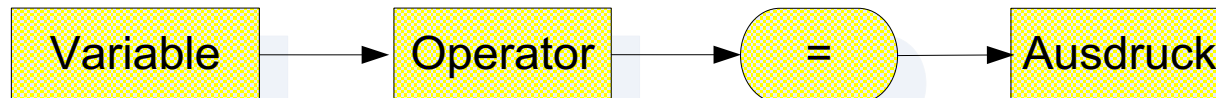


Quelle: Müller, Weichert, Vorkurs

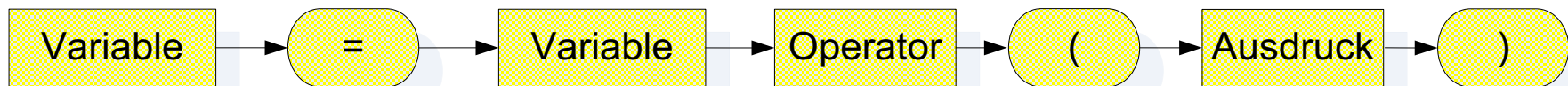
Zuweisung



Spezielle Kurznotation einer Zuweisung



Steht abkürzend für:



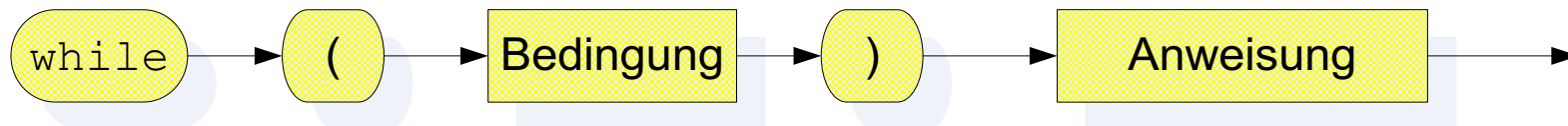
zusätzlich:

- $a++$ und $++a$ steht für $a+=1$ und damit $a = a + 1$;
- $a--$ und $--a$ steht für $a-=1$ und damit $a = a - 1$;

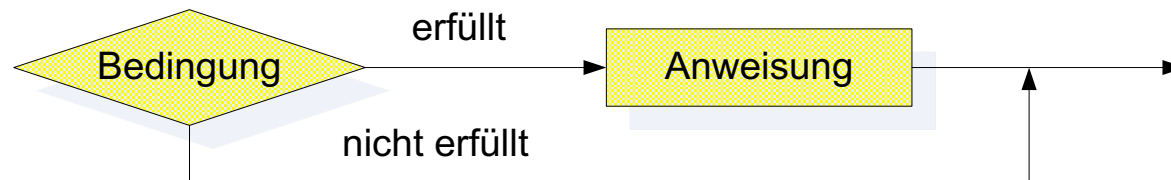
Quelle: Echte, Goedicke, Lehrbuch

- erlaubt eine Anweisung (oder einen Block) mehrmals auszuführen
- erst Bedingung ausgewertet
- falls Bedingung `true` wird Anweisung ausgeführt
- dann an Anfang der Schleife

Syntax



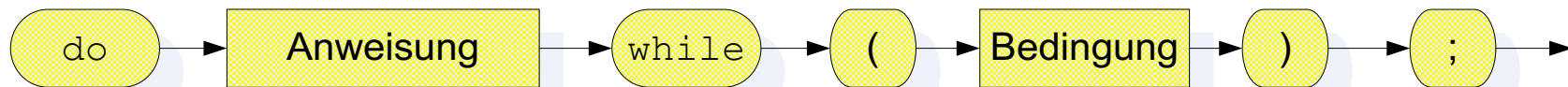
Semantik



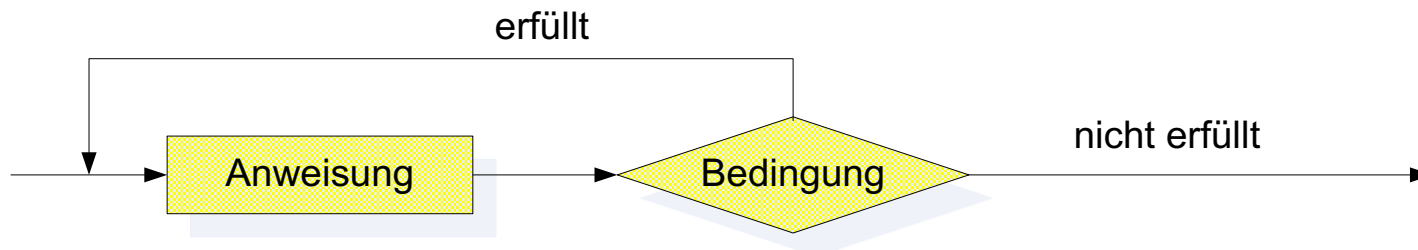
Do-While Schleife

- Anweisung (oder Block) wird auf jeden Fall ausgeführt
- dann Bedingung ausgewertet
- falls Bedingung `true` wird Anweisung erneut ausgeführt

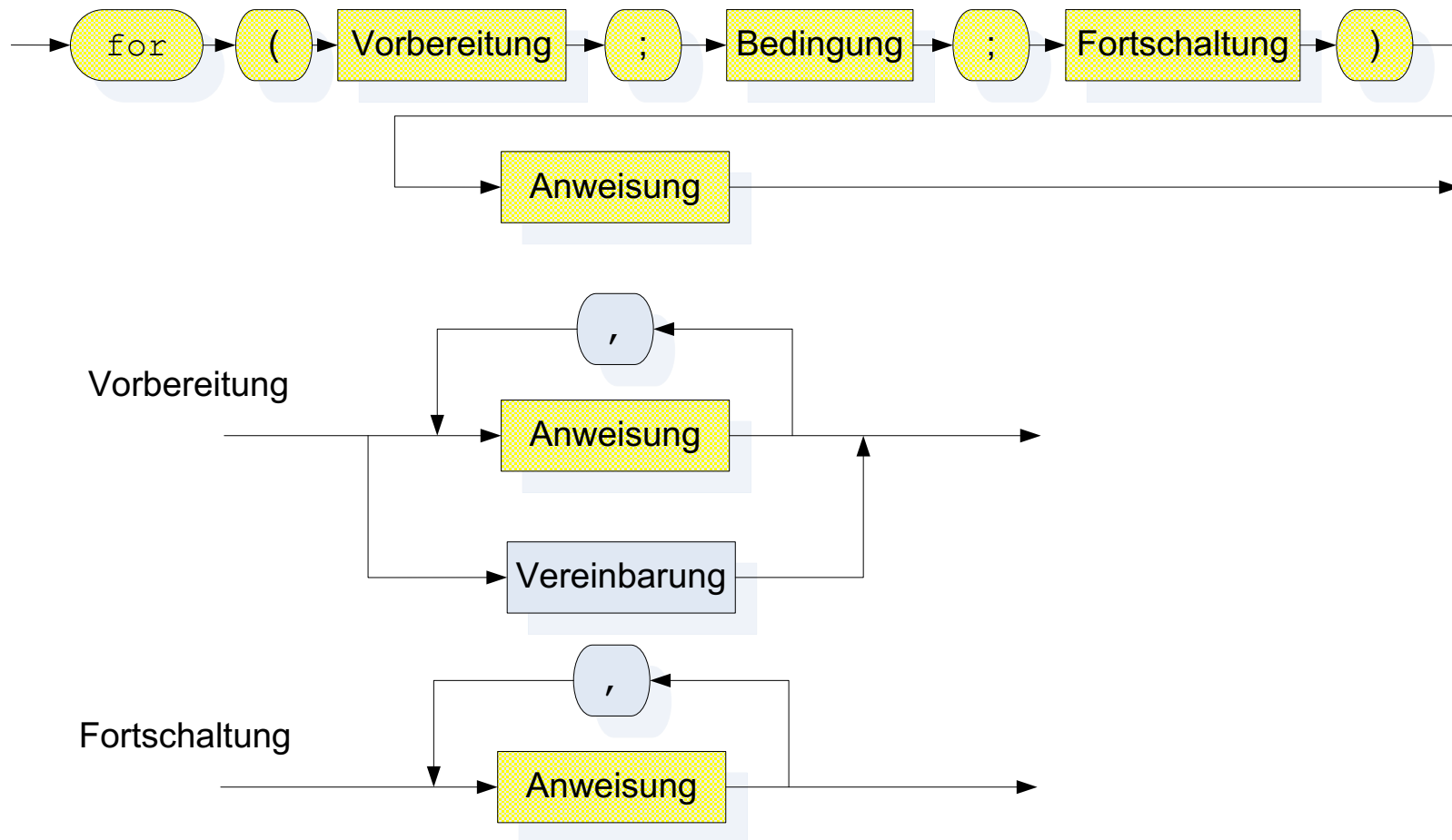
Syntax



Semantik

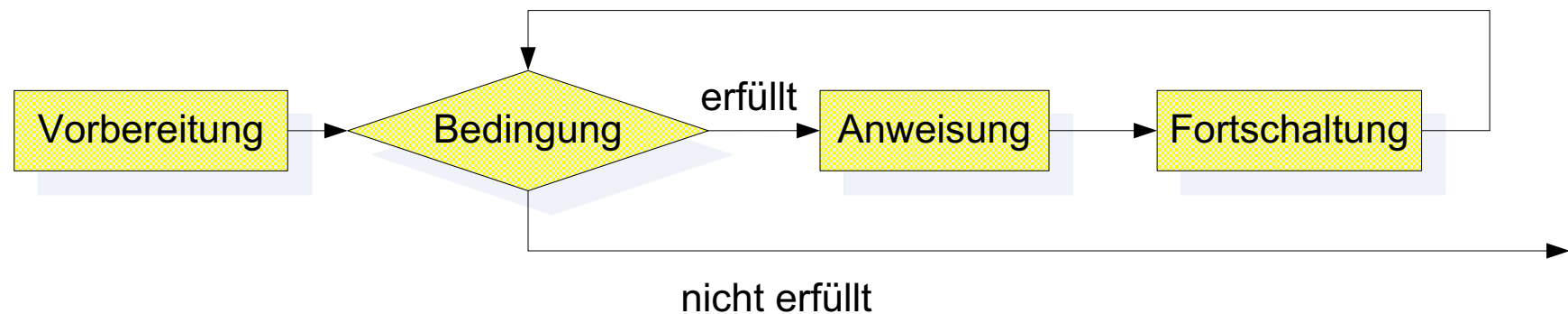


Syntax



- Vorbereitung beim Eintritt in Schleife ausgeführt
- Bedingung vor jeder Ausführung der Anweisung
- Fortschaltung wird **nach** der Anweisung ausgeführt

Semantik



Welche Schleife terminiert?

```
int i,j;
```

```
i = 1;  
j = 1;
```

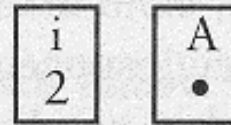
```
do {  
    i = i + j;  
    j++;  
} while (i < 200);
```

```
i = 1;  
j = 20;  
while (i+j > i) {  
    i = i + 2;  
    j--;  
}
```

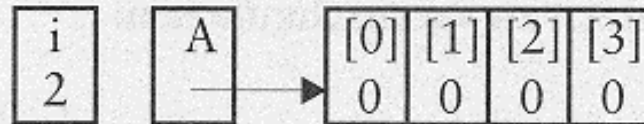
```
i = 100;  
j = 27;  
while (i !=j) {  
    i = i / 2;  
    j = j / 3;  
}
```

Zeiger Prinzip (1)

```
int i = 2;  
int [] A;
```



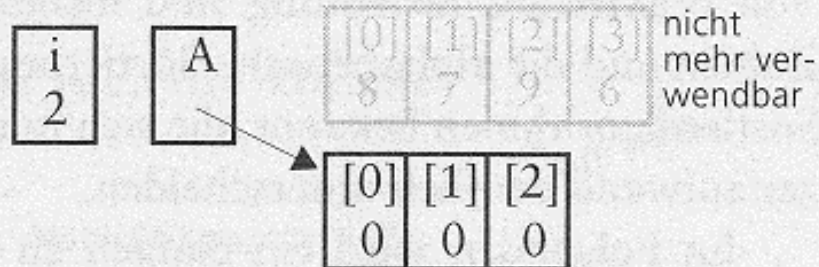
```
A = new int [4];
```



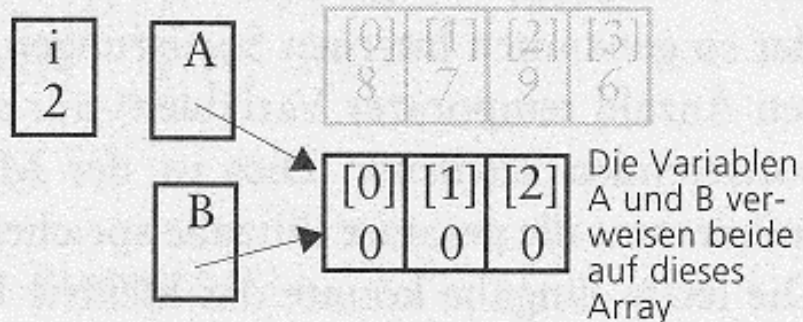
```
A [0] = 8;    A [1] = 7;  
A [i] = 9;    A [3] = 6;
```



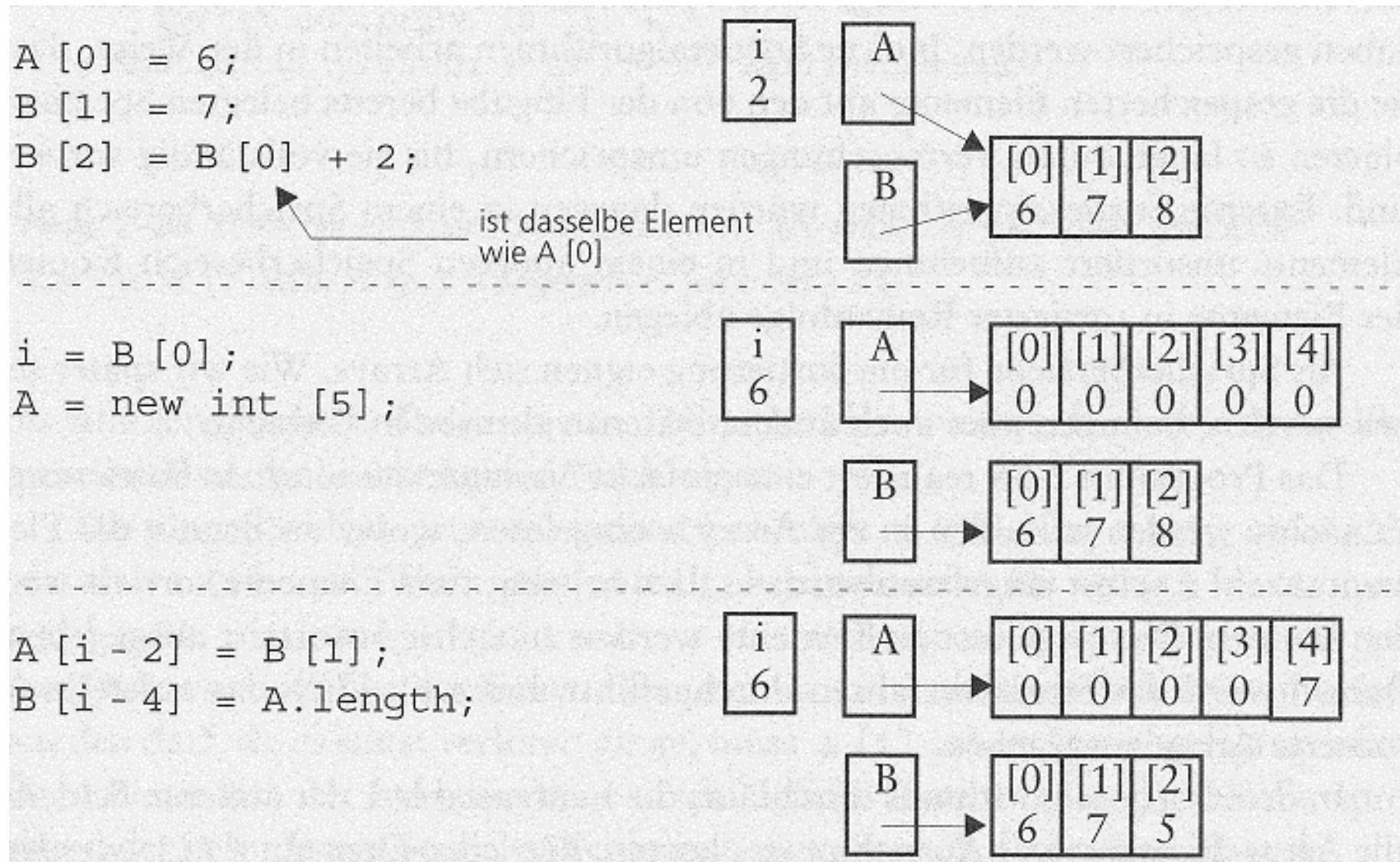
```
A = new int [3];
```



```
int [] B;  
B = A;
```



Zeiger Prinzip (2)



Gegeben sei ein Array `folge` bestehend aus Integer Zahlen (oder anderen Datentypen, für welche Vergleichsoperatoren `<`, `>`, `=` erklärt sind).

Ein Sortieralgorithmus soll die Elemente von `folge` so umordnen, dass

$$\begin{aligned} &\text{folge}[0] \leq \text{folge}[1] \leq \text{folge}[2] \leq \\ &\quad \dots \\ &\quad \leq \text{folge}[\text{folge.length} - 2] \leq \text{folge}[\text{folge.length} - 1] \end{aligned}$$

Um Speicherplatz zu sparen, soll der Algorithmus **keine Kopie** von `folge` erzeugen, sondern die Elemente **direkt umsortieren**.

1. Durchlauf	<table><tr><td>5</td><td>1</td><td>8</td><td>3</td><td>9</td><td>2</td></tr></table>						5	1	8	3	9	2
5	1	8	3	9	2							
2. Durchlauf	1	<table><tr><td>5</td><td>8</td><td>3</td><td>9</td><td>2</td></tr></table>					5	8	3	9	2	
5	8	3	9	2								
3. Durchlauf	1	2	<table><tr><td>8</td><td>3</td><td>9</td><td>5</td></tr></table>				8	3	9	5		
8	3	9	5									
4. Durchlauf	1	2	3	<table><tr><td>8</td><td>9</td><td>5</td></tr></table>			8	9	5			
8	9	5										
5. Durchlauf	1	2	3	5	<table><tr><td>9</td><td>8</td></tr></table>		9	8				
9	8											
Ergebnis	1	2	3	5	8	9						

- **Legende:**

- in dem Eingerahmten wird das kleinste Element gesucht
- die beiden gelben Elemente werden vertauscht

Bubble Sort

1. Durchlauf	5	3	8	9	2	1
	3	5	8	9	2	1
	3	5	8	2	9	1
	3	5	8	2	1	9

2. Durchlauf	3	5	8	2	1	9
	3	5	2	8	1	9
	3	5	2	1	8	9

3. Durchlauf	3	5	2	1	8	9
	3	2	5	1	8	9
	3	2	1	5	8	9

4. Durchlauf	3	2	1	5	8	9
	2	3	1	5	8	9
	2	1	3	5	8	9

5. Durchlauf	2	1	3	5	8	9
	1	2	3	5	8	9

Ergebnis	1	2	3	5	8	9
----------	---	---	---	---	---	---

Erklärung:

In dem Eingerahmten wird
von links nach rechts
paarweise verglichen und
ggf. vertauscht

- Zwei Matrizen $A = (a_{ij})$ und $B = (b_{ij})$ werden addiert bzw. subtrahiert, indem man die an gleicher Stelle stehenden Elemente addiert bzw. subtrahiert.

- Beispiel

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} + \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} = \begin{pmatrix} 1+7 & 2+8 & 3+9 \\ 4+10 & 5+11 & 6+12 \end{pmatrix} = \begin{pmatrix} 8 & 10 & 12 \\ 14 & 16 & 18 \end{pmatrix}$$

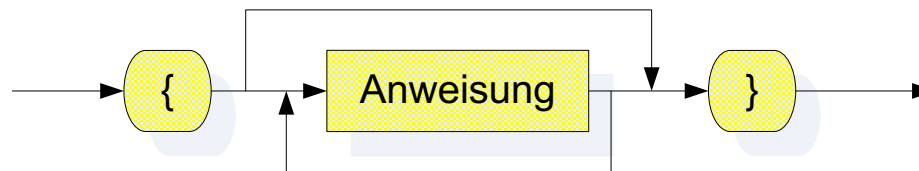
- Das Element c_{ik} des Produktes C erhält man als skalares Produkt der i -ten Zeile von A und der k -ten Spalte von B .

- Beispiel

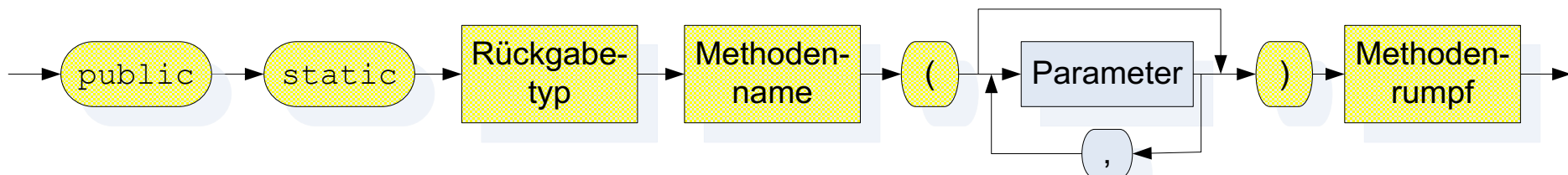
$$\begin{pmatrix} 4 & 1 \\ 2 & 5 \\ 6 & 0 \\ 2 & 8 \end{pmatrix} * \begin{pmatrix} 2 & 1 & 3 \\ 5 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 4*2+1*5 & 4*1+1*1 & 4*3+1*0 \\ 2*2+5*5 & 2*1+5*1 & 2*3+5*0 \\ 6*2+0*5 & 6*1+0*1 & 6*3+0*0 \\ 2*2+8*5 & 2*1+8*1 & 2*3+8*0 \end{pmatrix} = \begin{pmatrix} 13 & 5 & 12 \\ 29 & 7 & 6 \\ 12 & 6 & 18 \\ 44 & 10 & 6 \end{pmatrix}$$

- Alternative Begriffe: Funktion, Unterprogramm
- Enthalten eine Folge von Anweisungen (Methodenrumpf) und lösen ein Teilproblem des Gesamtproblems

Methodenrumpf



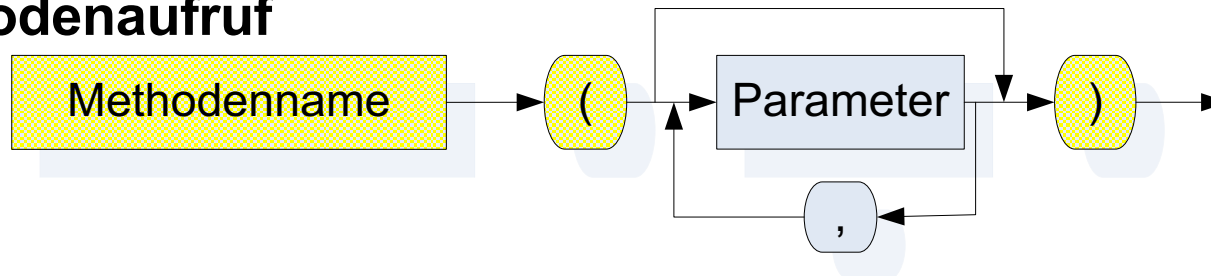
Statische Methode



- Methoden ohne `public static` werden später behandelt

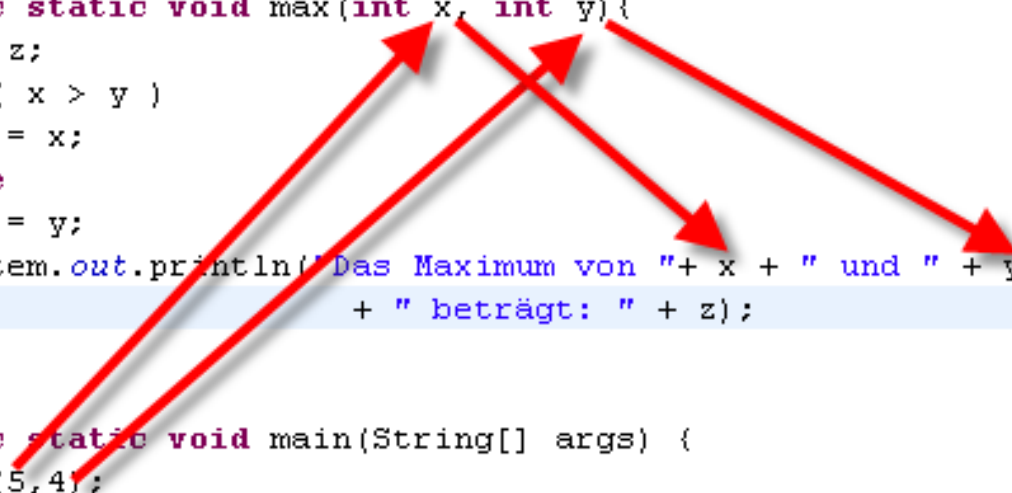
- Methodenaufruf bewirkt, dass die Anweisungen ausgeführt werden

Methodenaufruf



- Parameter im Rumpf werden durch aktuelle Werte ersetzt

```
public class MaxAusdrucken {  
    public static void max(int x, int y) {  
        int z;  
        if ( x > y )  
            z = x;  
        else  
            z = y;  
        System.out.println("Das Maximum von "+ x + " und " + y  
                           + " beträgt: " + z);  
    }  
  
    public static void main(String[] args) {  
        max(5,4);  
        max(7,8);  
    }  
}
```

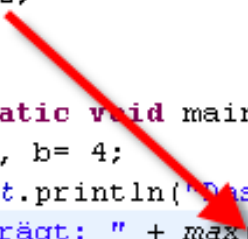


Rückgabe eines Wertes

- Methode kann einen Wert an das aufrufende Programm zurückgeben
 - In diesem Fall muss der Typ des Wertes bei der Definition der Methode angegeben werden
 - Methode muss `return` enthalten

```
public class FunktionMax {  
    public static int max(int x, int y) {  
        int z;  
        if ( x > y )  
            z = x;  
        else  
            z = y;  
        return z;  
    }  
  
    public static void main(String[] args) {  
        int a = 5, b = 4;  
        System.out.println("Das Maximum von " + a + " und " + b +  
            " beträgt: " + max(a,b));  
    }  
}
```

es wird ein Integer zurück gegeben



- Soll kein Wert zurück gegeben werden: Typ `void`

- Wiederverwendung von Code
- Schreiben eigener Operationen (wenn die Sprache nicht ausreicht)
- Strukturierung, d.h. Zerlegung des Programms in übersichtliche Teileinheiten (statt Spaghetti Code)

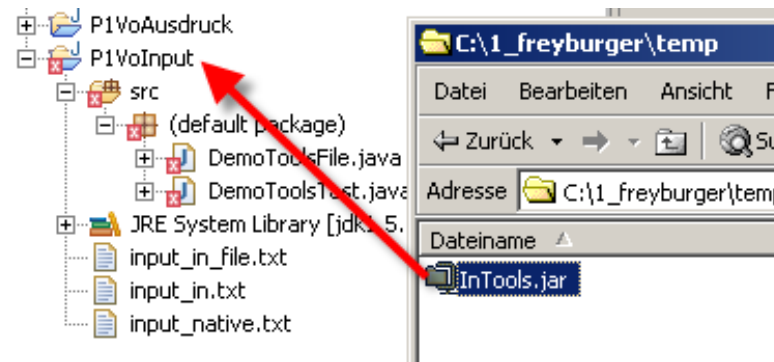
- Der Parameter in der Methodendefinition nennt man **formale Parameter** oder **formale Variablen**, in dem Aufruf **aktuelle Parameter**.
- Die Parameterübergabe besitzt die gleiche Semantik wie eine gewöhnliche Zuweisung
 - gewöhnliche Variablen werden kopiert
 - Arrays wird nur die Pointer kopiert → in der Methode kann das Array verändert werden!

- **Klassenvariablen** sind in der Klasse bekannt
- Variablen in einem Block (z.B. Methode) bezeichnet man als **lokal** bzgl. dieses Blocks
 - Variable ist dort sowie in allen Unterblöcken **gültig** bzw. **sichtbar**.
- Als **formale Variablen** bezeichnet man die Parameter der Methode
- Innerhalb von Methoden **verdecken** lokale Variablen und formale Variablen die Klassenvariablen gleichen Namens, sodass diese während der Ausführung der Methoden vorübergehend nicht sichtbar und damit auch (zumindest allein über ihren Bezeichner) nicht zugreifbar sind.

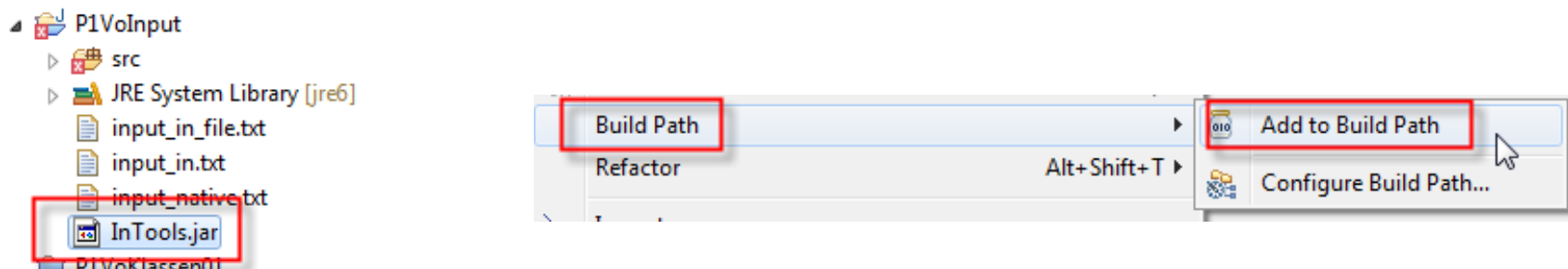
- Verwendet Klassen von Java
- Zwei Optionen (siehe Beispielprogramme)
 - Scanner
 - InputStream
- Tools zur Eingabe
 - Eingabe Tastatur
 - Klasse InTast
 - Quelle: Ratz/Scheffler/Seese, Grundkurs
 - Eingabe File
 - Klasse InFile
 - Quelle: Mössenböck, Sprechen Sie Java
 - InFile.done() ermittelt, ob die letzte Aktion erfolgreich war

Verwendung der Tools zur Eingabe

- So bekommt man die Tools in sein Projekt:
 - Das File **InTools.jar** per **Drag&Drop** in sein Projekt kopieren...



- ...und bekannt machen



- Wo man es verwenden möchte: **import InTools.*;**

Verwendung der Klasse InFile

- Wenn das Lesen (bzw. die letzte Aktion) erfolgreich war, wird `InFile.done()` gesetzt

- Eine Datei kann wie folgt verarbeitet werden:
wenn Datei im Projektverzeichnis: Ohne Pfad

```
InFile.open("input.txt");
```

```
if (InFile.done()) {           // Öffnen erfolgreich
```

```
    int x = InFile.readInt();  //den ersten Eintrag lesen
```

```
    while (InFile.done()) {    Nur verarbeiten, wenn lesen erfolgreich!!!!
```

```
        ...process x ...
```

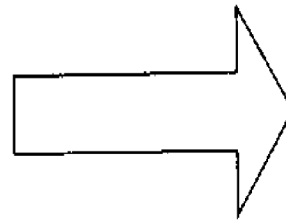
```
        x = InFile.readInt();  // lesen für den nächsten
    }                          // Schleifendurchlauf
```

```
InFile.close();
```

```
}
```

Quelle: Mössenböck, Sprechen Sie Java

Wahre Welt



Modell

Karl Friedrich v. Mustermann

Schopenhauergasse 23

72434 Bad-Sulzingen

Mustermann@musterbach-online.de

Adresse

name: String

strasse: String

hausnummer: int

postleitzahl: int

wohnort: String

mail: String

kommentar: String

Quelle: Ratz/Scheffler/Seese, Grundkurs

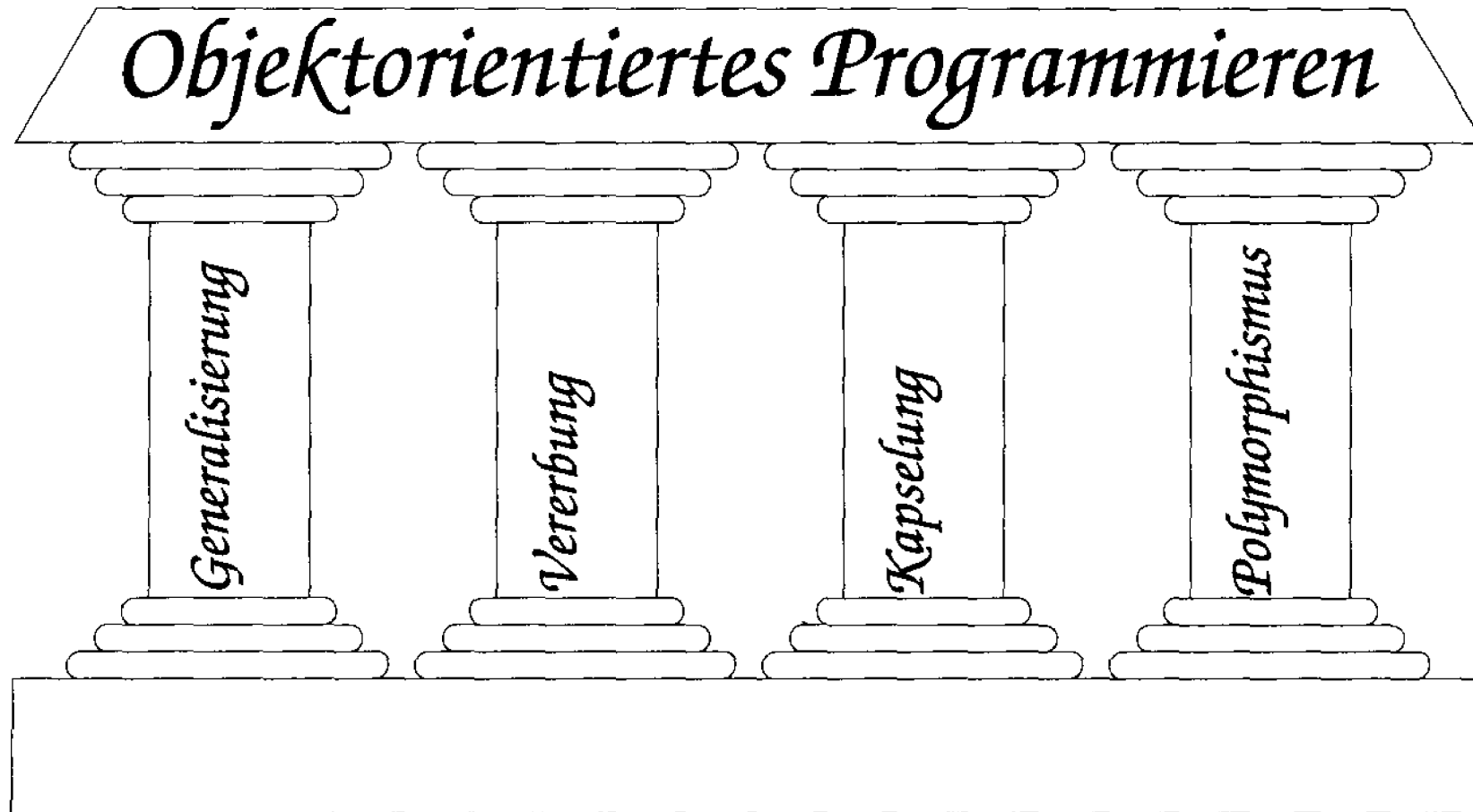
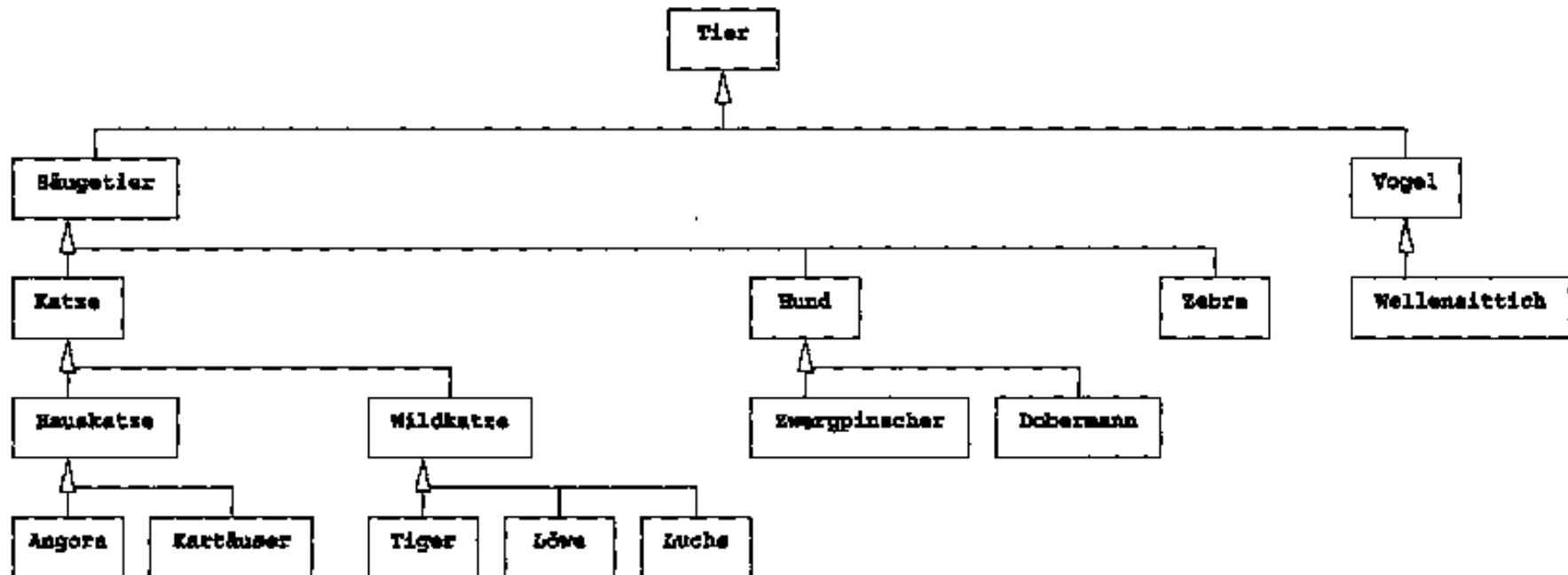


Abbildung 9.2: Grundpfeiler der objektorientierten Programmierung

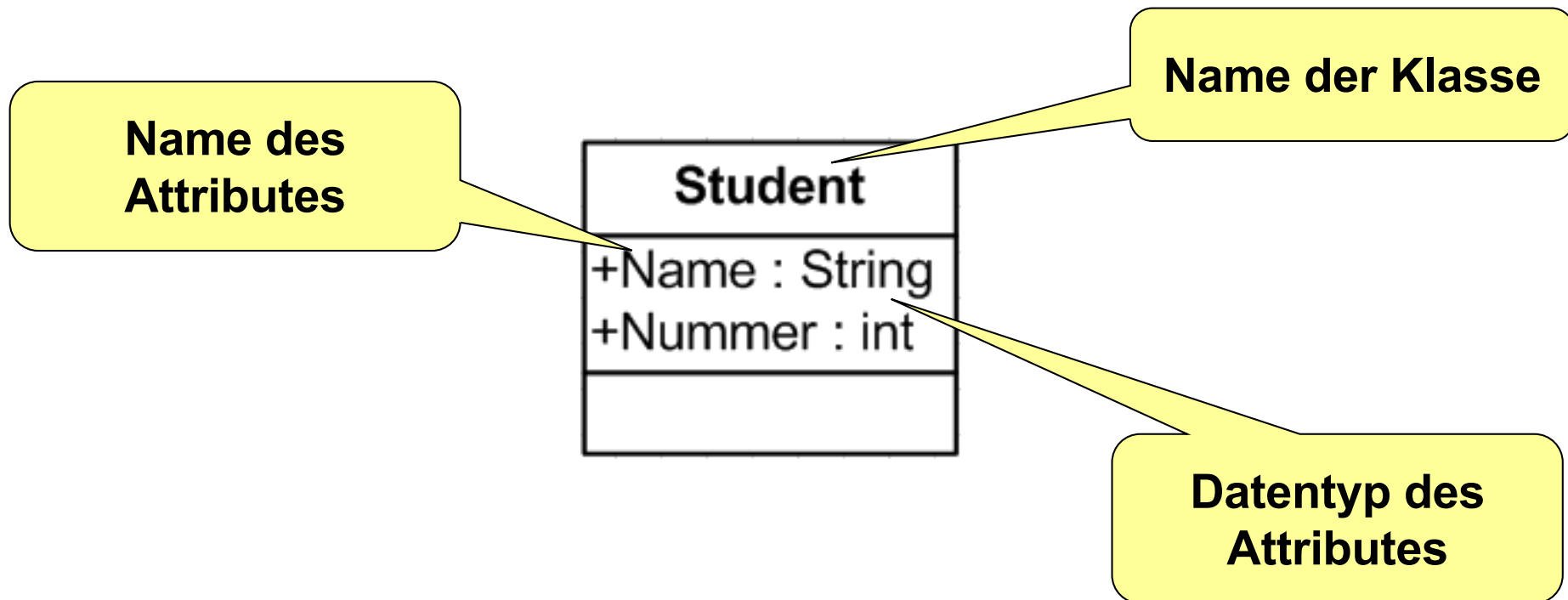
Quelle: Ratz/Scheffler/Seese, Grundkurs



Quelle: Ratz/Scheffler/Seese, Grundkurs

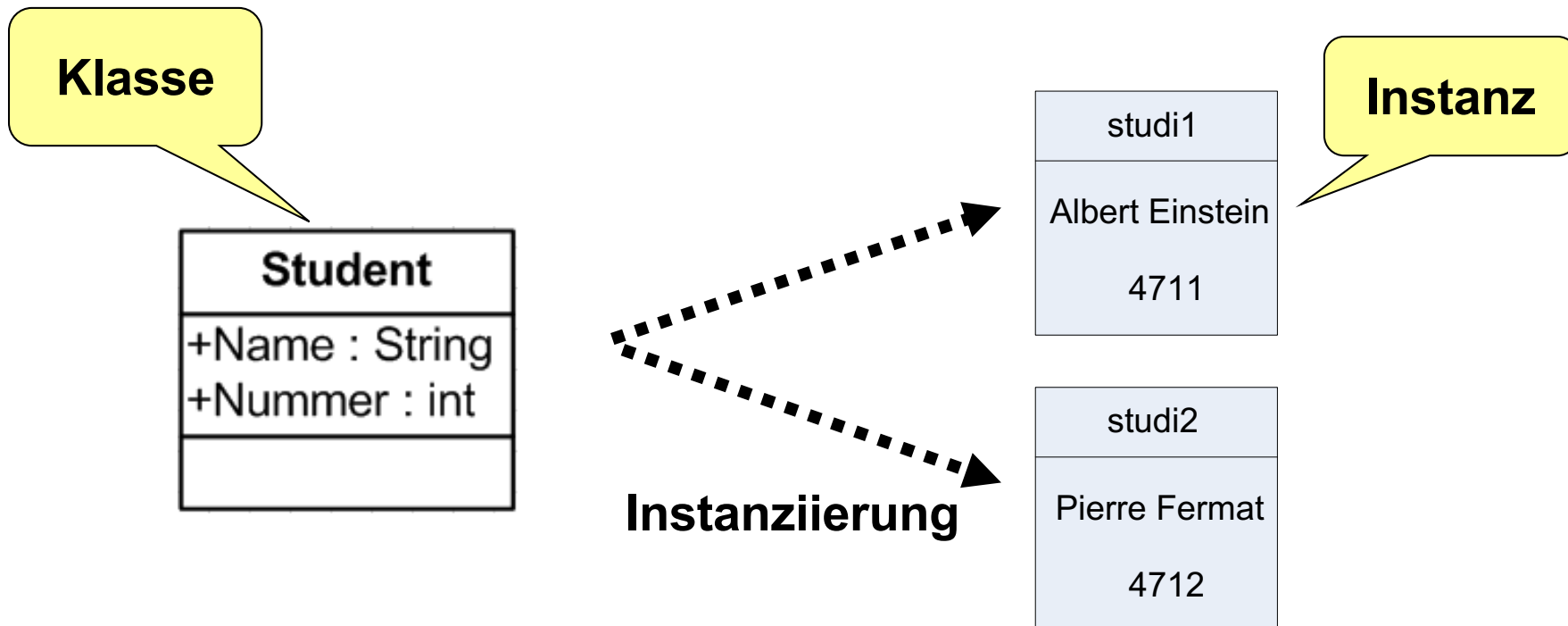
Klasse Student mit Attributen

- Klasse als **Bauplan** für einen Datenspeicher



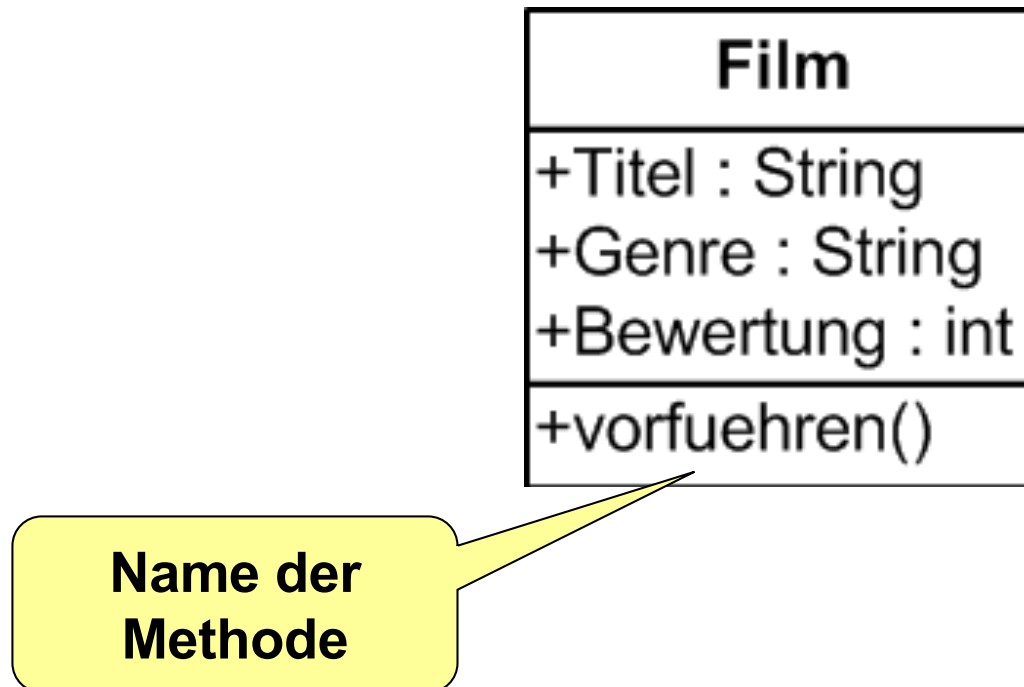
Quelle: Ratz/Scheffler/Seese, Grundkurs

- Aus der Klassendeklaration werden individuelle Vertreter erzeugt: **Objekte** oder **Instanzen**
- → „Instanziierung“



Klasse Film mit Methode

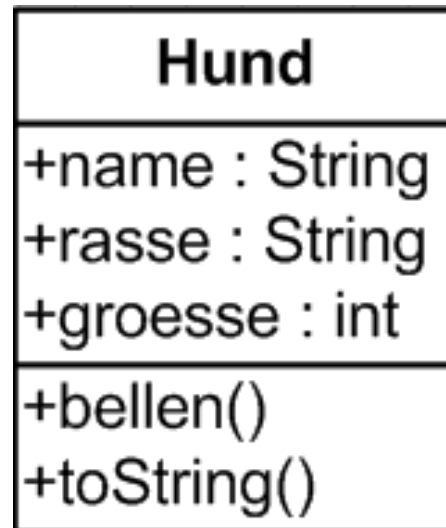
- **Methoden** sind Dinge, die ein Objekt **tun** kann
- Attribute und Methoden bilden eine **Einheit**



Quelle: Sierra / Bates: Java von Kopf bis Fuß

Klasse Hund mit Methode toString

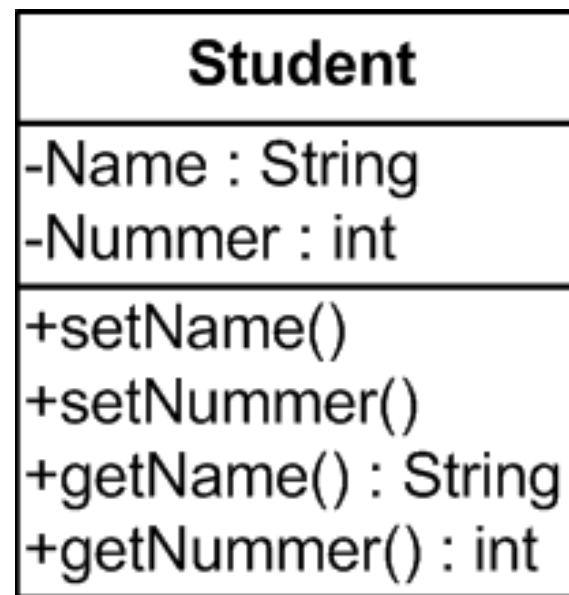
- **toString** spezielle Methode, die beim Drucken des Objektes aufgerufen wird



**Methode zum
Drucken**

Quelle: Sierra / Bates: Java von Kopf bis Fuß

**Attribute sind
privat**



**Methoden sind
öffentlich**

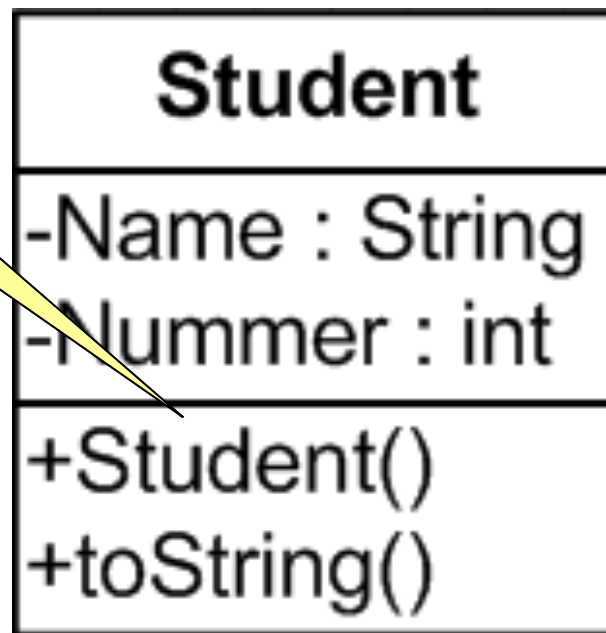
Quelle: Ratz/Scheffler/Seese, Grundkurs

Ein Konstruktor ist eine **Methode**, die den **gleichen Namen** hat wie die Klasse, zu der sie gehört.

Sie wird ohne Funktionstyp und ohne Schlüsselwort void deklariert, kann aber Parameter haben, in denen üblicherweise Initialwerte der Objektfelder übergeben werden

Klasse Student mit Konstruktor

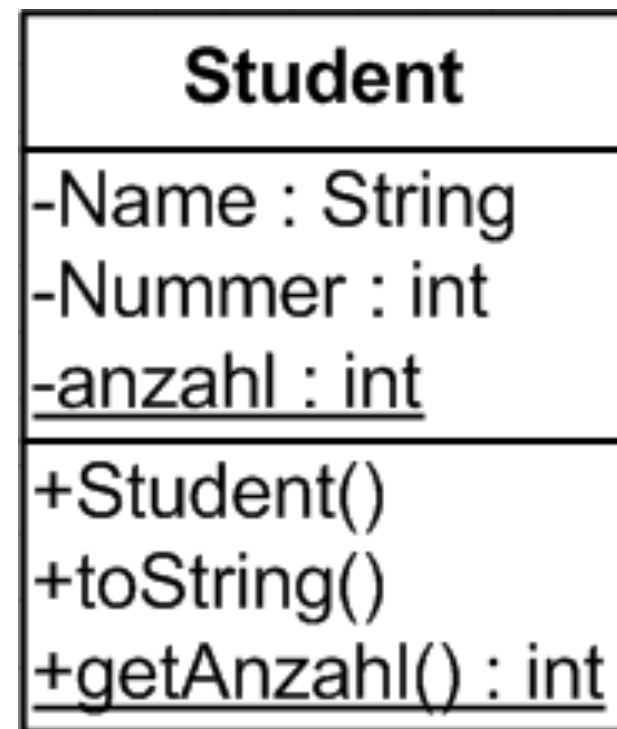
Konstruktor



- Statische Attribute und Methoden gibt es nur ein Mal pro Klasse (unabhängig von Instanzen)

Attribut ist static

Methode ist static



Statisch vs. Objektbezogen

Tab. 11.1 Statische und objektbezogene Komponenten einer Klasse

	Objektbezogene Komponenten	Statische Komponenten
Deklaration	ohne static	mit static
existieren	in jedem Objekt	nur einmal pro Klasse
Felder werden angelegt	wenn das Objekt erzeugt wird	wenn die Klasse geladen wird (am Anfang des Programms)
Felder werden freigegeben	vom Garbage Collector, wenn kein Zeiger mehr auf das Objekt zeigt	wenn die Klasse entladen wird (am Ende des Programms)
Konstruktor wird aufgerufen	wenn das Objekt erzeugt wird	wenn die Klasse geladen wird
Felder werden angesprochen	obj.field this.field	Class.field
Methoden werden aufgerufen	obj.m(); this.m();	Class.m();

Tab. 10.1 Unterschiede zwischen Arrays und Klassen

Arrays	Klassen
Bestehen aus mehreren <i>gleichartigen</i> Elementen, z.B. aus lauter <i>int</i> -Werten.	Können aus mehreren <i>verschiedenartigen</i> Feldern bestehen, z.B. aus einem <i>int</i> -Wert und einem <i>String</i> -Wert.
Die Elemente eines Arrays haben keinen Namen, sondern werden über einen Index angesprochen, z.B. a[3].	Die Felder einer Klasse haben einen Namen und werden über diesen Namen angesprochen, z.B. x.day.
Die Anzahl der Elemente wird bei der Erzeugung des Arrayobjekts festgelegt.	Die Anzahl der Felder wird bei der Deklaration der Klasse festgelegt.

Quelle: Mössenböck

- In vielen modernen Programmiersprachen werden Exceptions verwendet, um Fehler abzufangen.
- Falls ein Befehl nicht ausgeführt werden kann, wird das Programm beendet und eine **Laufzeit-Exception** ausgelöst.
- „Vorbeugen“ ist möglich:
 - Man muss wissen, wo etwas passieren kann (nicht genau, was passieren kann).
 - Diese Stelle wird in einen **try/catch-Block** eingeschlossen.
 - Sobald im try-Block ein Fehler auftritt, wird der try-Block abgebrochen und ein Exception-Objekt erzeugt. Dieses wird als Parameter an den catch-Block übergeben. Der catch-Block wird als ganz normaler Java-Code ausgeführt. Danach geht es hinter dem catch-Block weiter.
- Durch mehrere catch-Blöcke lassen sich unterschiedliche Fehlersituationen behandeln
- Mit **throw** kann man selbst eine Exception werfen
- Noch flexibler ist man durch eine **eigene Exception** (Konzept Vererbung → 2. Semester)

Beispiel zur Kombination von Klassen und Arrays

Struktur der Klassen:

