

# **Datentypen, Kodierung, Variablen, Arrays, Binärzahlen**

### 1. Java Grundlagen: Entwicklungszyklus, Entwicklungsumgebung

### 2. Datentypen, Kodierung, Binärzahlen, Variablen, Arrays

### 3. Ausdrücke, Operatoren, Schleifen und Verzweigungen

### 4. Blöcke, Sichtbarkeit und Methoden (Teil 1)

### 5. Grundkonzepte der Objektorientierung

### 6. Objektorientierung: Sichtbarkeit, Vererbung, Methoden (Teil 2), Konstruktor

### 7. Packages, lokale Klassen, abstrakte Klassen und Methoden, Interfaces, enum

### 8. Arbeiten mit Objekten: Identität, Listen, Komparatoren, Kopien, Wrapper, Iterator

### 9. Fehlerbehandlung: Exceptions und Logging

### 10. Utilities: Math, Date, Calendar, System, Random

### 11. Rekursion, Sortieralgorithmen und Collections

### 12. Nebenläufigkeit: Arbeiten mit Threads

### 13. Benutzeroberflächen mit Swing

### 14. Streams: Auf Dateien und auf das Netzwerk zugreifen

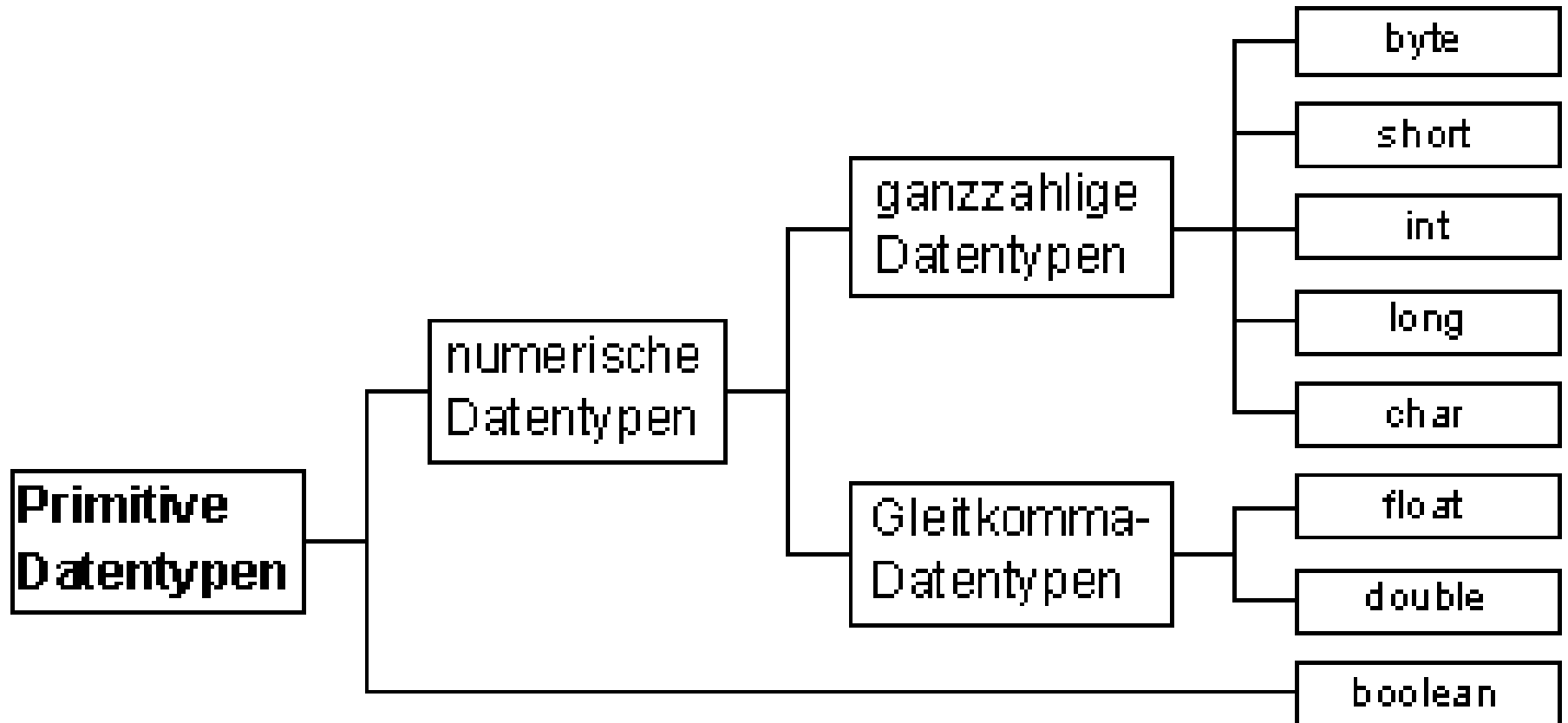
- Java Grundlagen (JDK / JRE, etc.)
- eclipse IDE: Erstes Projekt und erste Klasse mit main-Methode
- eclipse IDE: Debugging und Breakpoints

- Namen von Variablen, Klassen und Methoden
- Beliebig lang
- Beginnen mit Unicode-Buchstaben: A-Z, a-z , \_ und \$
- Üblich: „CamelCase“-Schreibweise

```
public static void main(String[] args) {  
  
    int x = 4;  
    int eineVariable = 6;  
    int __nochEineVariable = 5;  
}
```

# Grunddatentypen

- Acht elementare Datentypen
- Fest definierter Wertebereich
  - >> Plattformübergreifend, was ein wichtiger Vorteil ist
- Besitzen einen Standard-Wert (Default-Wert)



Name	Länge (Bytes)	Wertebereich	Standardwert
<a href="#"><u>boolean</u></a>	1	<a href="#"><u>true</u></a> , <a href="#"><u>false</u></a>	<a href="#"><u>false</u></a>
<a href="#"><u>char</u></a>	2	Alle Unicode-Zeichen	\u0000
<a href="#"><u>byte</u></a>	1	$-2^7 \dots 2^7 - 1$	0
<a href="#"><u>short</u></a>	2	$-2^{15} \dots 2^{15} - 1$	0
<a href="#"><u>int</u></a>	4	$-2^{31} \dots 2^{31} - 1$	0
<a href="#"><u>long</u></a>	8	$-2^{63} \dots 2^{63} - 1$	0
<a href="#"><u>float</u></a>	4	$\pm 3.40282347 * 10^{38}$	0.0
<a href="#"><u>double</u></a>	8	$\pm 1.79769313486231570 * 10^{308}$	0.0



## Der logische Typ boolean

- Abbildung von Werten der Aussagenlogik
- Nur zwei mögliche Werte: „wahr“ oder „falsch“  
(**nicht** 0 bzw. 1 als logische Werte, wie in C)

```
boolean variable = true;  
System.out.println(variable);
```

## Der Zeichentyp char

- Abbildung des UNICODE-Zeichensatzes
- char-Literale: in einfache Hochkommata setzen: 'a'

```
char buchstabe = 'd';
```

```
System.out.println(buchstabe);
```

000	NUL	033	!	066	B	099	c	132	ä	165	Ñ	198	ã	231	þ
001	Start Of Header	034	"	067	C	100	d	133	å	166	▪	199	Ä	232	Þ
002	Start Of Text	035	#	068	D	101	e	134	ä	167	°	200	ℒ	233	Ú
003	End Of Text	036	\$	069	E	102	f	135	ç	168	¿	201	ℝ	234	Û
004	End Of Transmission	037	%	070	F	103	g	136	ê	169	®	202	℥	235	Ü
005	Enquiry	038	&	071	G	104	h	137	ë	170	¬	203	℟	236	Ý
006	Acknowledge	039		072	H	105	i	138	è	171	½	204	℣	237	Ý
007	Bell	040	(	073	I	106	j	139	í	172	¼	205	=	238	~
008	Backspace	041	)	074	J	107	k	140	î	173	í	206	℥	239	˘
009	Horizontal Tab	042	*	075	K	108	l	141	ì	174	«	207	⌘	240	-
010	Line Feed	043	+	076	L	109	m	142	Ä	175	»	208	ð	241	±
011	Vertical Tab	044	,	077	M	110	n	143	Å	176	∴	209	Ð	242	–
012	Form Feed	045	-	078	N	111	o	144	É	177	∵	210	È	243	¼
013	Carriage Return	046	.	079	O	112	p	145	æ	178	⌘	211	Ê	244	¶
014	Shift Out	047	/	080	P	113	q	146	Æ	179		212	Ë	245	§
015	Shift In	048	0	081	Q	114	r	147	ø	180	¡	213	Ì	246	÷
016	Delete	049	1	082	R	115	s	148	ö	181	À	214	Í	247	˘
017	-- frei --	050	2	083	S	116	t	149	ò	182	Á	215	Î	248	▪
018	-- frei --	051	3	084	T	117	u	150	ú	183	Â	216	Ï	249	-
019	-- frei --	052	4	085	U	118	v	151	û	184	Ë	217		250	.
020	-- frei --	053	5	086	V	119	w	152	ÿ	185	℥	218		251	!
021	Negative Acknowledge	054	6	087	W	120	x	153	Ö	186		219	■	252	³
022	Synchronous Idle	055	7	088	X	121	y	154	Û	187	¶	220	■	253	²
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	¶	221	¡	254	■
024	Cancel	057	9	090	Z	123	{	156	£	189	¢	222	¡	255	
025	End Of Medium	058	:	091	[	124		157	Ø	190	¥	223	■		
026	Substitute	059	;	092	\	125	}	158	×	191	₹	224	Ó		
027	Escape	060	<	093	]	126	~	159	f	192	₹	225	ß		
028	File Separator	061	=	094	^	127		160	á	193	₹	226	Ö		
029	Group Separator	062	>	095	_	128	Ç	161	í	194	₹	227	Ö		
030	Record Separator	063	?	096	`	129	ü	162	ó	195	₹	228	ð		
031	Unit Separator	064	@	097	a	130	é	163	ú	196	–	229	Ö		
032		065	A	098	b	131	â	164	ñ	197	₹	230	µ		

- (Praktisch) alle Schriftzeichen aller Sprachen
- 17 Planes mit je 65.536 mgl. Zeichen
- Unicode 6.0: 109.449 Zeichen sind derzeit besetzt

Siehe dazu: <http://de.wikipedia.org/wiki/Unicode>



## Die erste der 17 Planes (Ebenen): Basic Multilingual Plane

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

■	Lateinische Schriften und Symbole
■	Lautschriften
■	Andere europäische Schriften
■	Nahost- und Südwestasiatische Schriften
■	Afrikanische Schriften
■	Südasiatische Schriften
■	Südostasiatische Schriften
■	Ostasiatische Schriften
■	CJK-Ideogramme
■	Kanadische Silben
■	Symbole
■	Diakritika
■	UTF-16-Surrogates und privater Nutzungsbereich
■	Verschiedene Zeichen
■	Nicht belegte Codebereiche

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	\$ 0024	% 0025	& 0026	' 0027	( 0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[ 005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	 007C	}	~ 007E	DEL 007F
80	€ 20AC		/ 201A	f 0132	// 201E	... 2026	† 2020	‡ 2021	^ 02C6	‰ 2030	Š 0160	< 2033	Œ 0152		Ž 017D	
90		\ 2018	/ 2019	“ 201C	” 201D	• 2022	— 2013	— 2014	™ 02DC	Š 2122	Š 0161	> 203A	œ 0153		ž 017E	ÿ 0178
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	­ 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

## Ganzzahlige Datentypen (Integer / Integrale Typen)

- Abbildung von ganzen Zahlen (...,-2,-1,0,1,2,3,...)
- Sowohl positive als auch negative Zahlen möglich
- Wertebereich wird durch die Anzahl der Bits festgelegt
- Übliche Bitgrößen:
  - 8Bit (=1 Byte)      byte
  - 16Bit (=2 Byte)    short
  - 32Bit (=4 Byte)    int
  - 64Bit (=8 Byte)    long
- Literale: Dezimal-, Oktal- (0), Hexadezimalschreibweise (0x)

## Ganzzahlige Datentypen (Integer / Integrale Typen)

**byte**            zahl1        = 1 ;

**short**           zahl2        = 2 ;

**int**             zahl3        = 3 ;

**long**            zahl4        = 4 ;



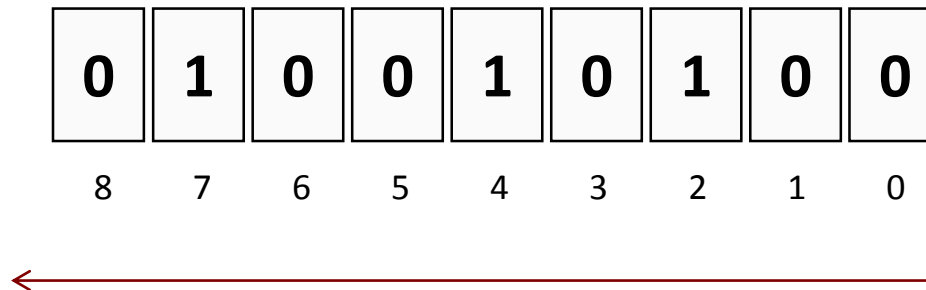
# Binärkodierung

- Speicherung und Verarbeitung von Informationen auf der Basis von Bits
- Ein Bit hat den Zustand 0 (aus / falsch) oder 1 (ein / wahr)
- Zwei Zeichen



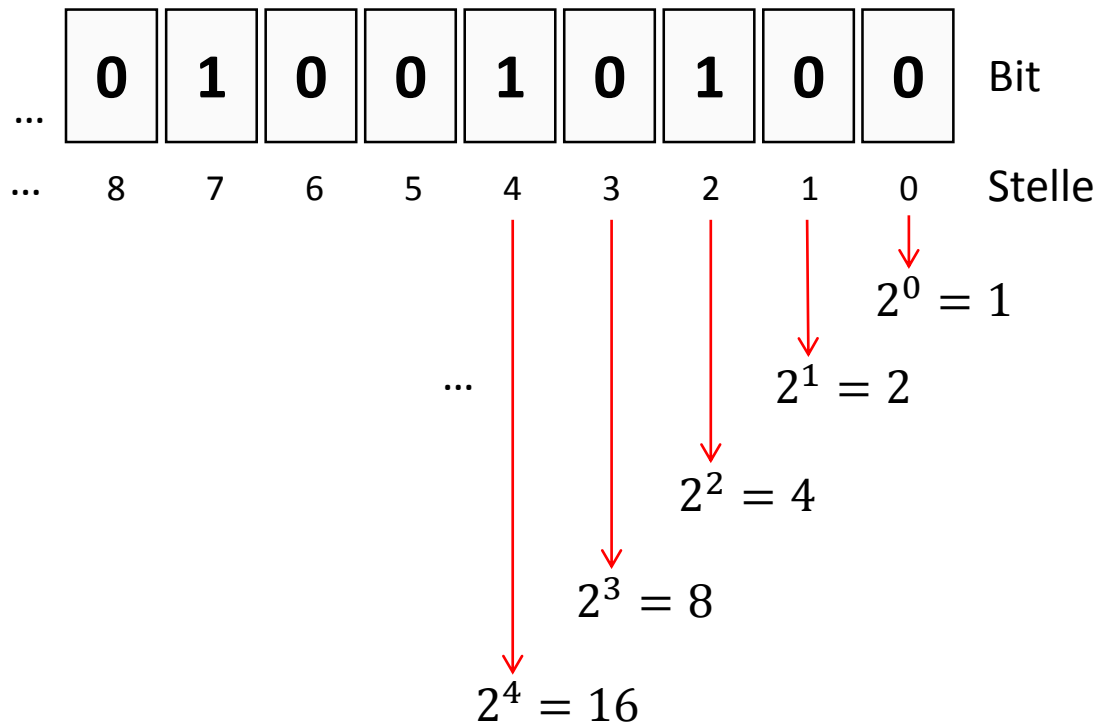
Binär-Uhr

- **Darstellung von Zahlen im Dualsystem**

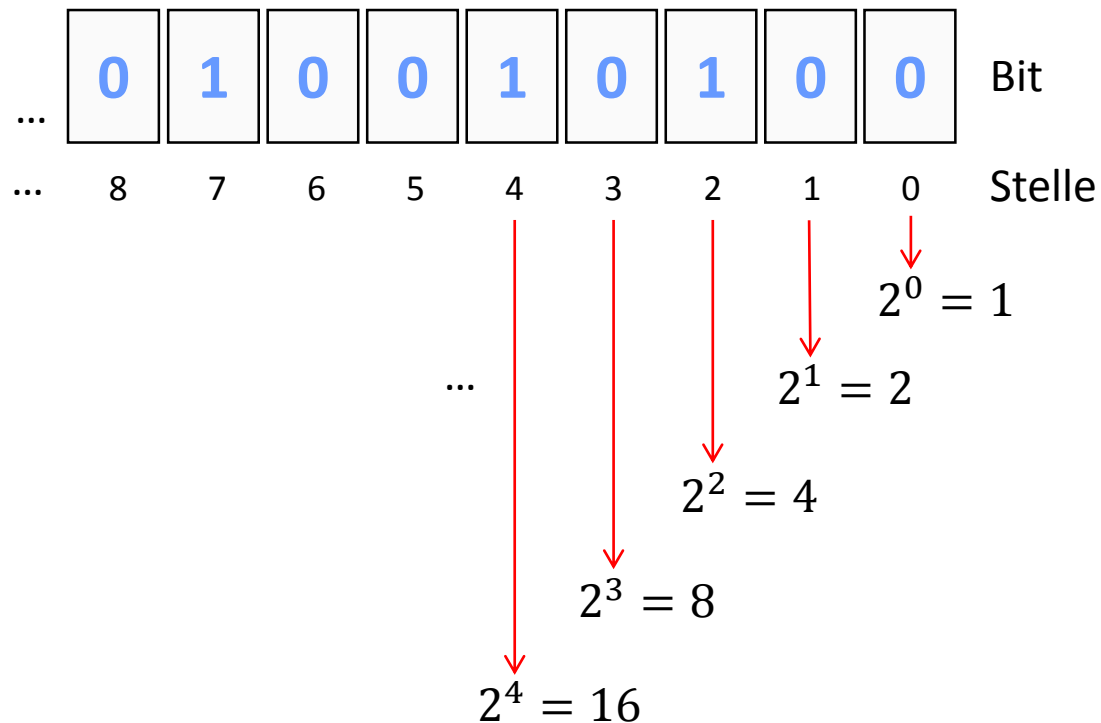


- Die Stellen (Positionen) werden von rechts nach links gezählt

- Wert einer Stelle im Dualsystem:

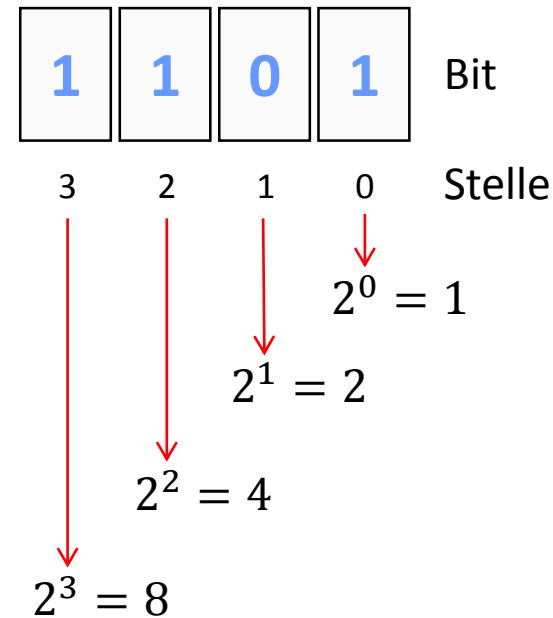


- Umrechnung in das Dezimalsystem



$$x = 0 * 2^0 + 0 * 2^1 + 1 * 2^2 + 0 * 2^3 + 1 * 2^4 + \dots$$

Beispiel: Ziffernfolge 1101



Dual:  $[1101]_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = [13]_{10}$

Dezimal:  $1 \cdot 10^3 + 1 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0 = [1101]_{10}$

- Wertebereich: Abhängig von der Anzahl der Bits

Anzahl der Bits	Mögliche Kombinationen
1	0, 1
2	00,01,10,11
3	000,001,010,011,100,101,110,111
...	...

- Anzahl der Kombinationen:  $2^{\text{Anzahl der Bits}}$

# Binärzahlen: 4 Bit ohne Vorzeichen

Bit 3 ( $2^3=8$ )	Bit 2 ( $2^2=4$ )	Bit 1 ( $2^1=2$ )	Bit 0 ( $2^0=1$ )	Dezimalzahl
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15



## Dezimalzahlen

- Umrechnen in Binärzahlen


1. Beginne mit Berechnung für 0. Bit
2. Dezimalzahl durch 2 teilen. Der Divisionsrest ergibt den Wert für das aktuelle Bit
3. Ist Divisionsergebnis = 0 oder das letzte Bit erreicht?

**Nein, weiter mit  
dem ganzzahligen  
Divisionsergebnis  
ab Schritt 2**

**Ja, fertig**

## Beispiel

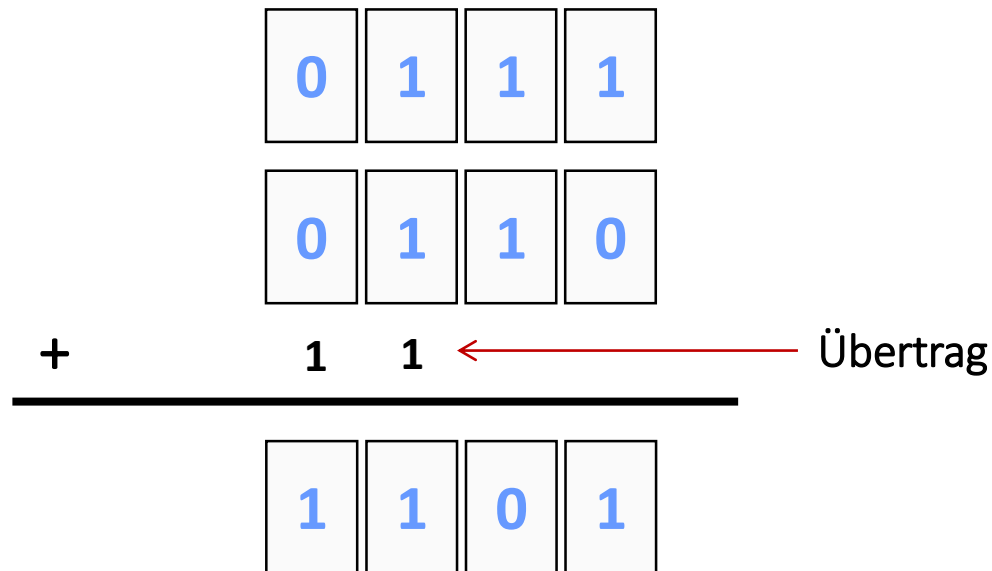
Bitnummer	Dezimalzahl	/2	Rest
0	5	2	1
1	2	1	0
2	1	0	1
3	0	0	0



*Dezimal : 5*

*Binär : 0101*

## Addition



### Rechenregeln

0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10

7

6

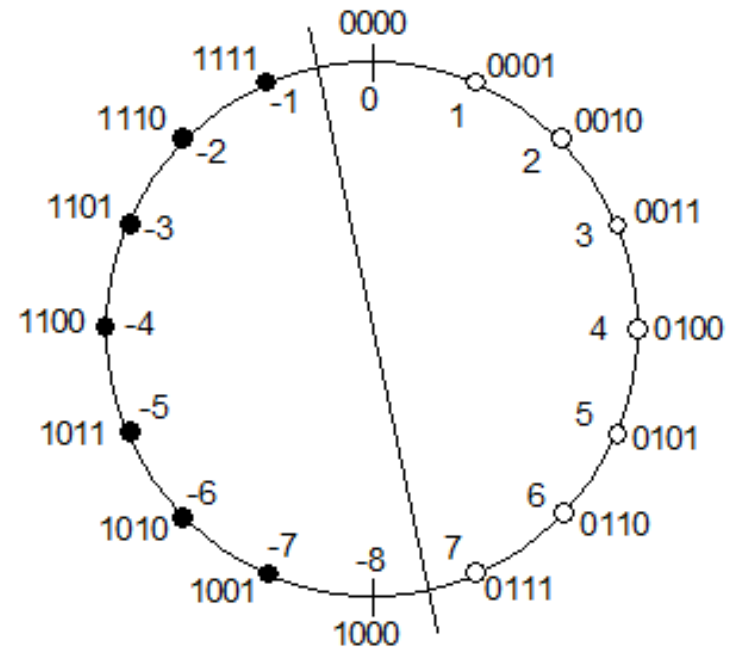
13

# Dualzahlen: 4 Bit mit Vorzeichen

Bit 3 ( <b>Vorzeichen</b> )	Bit 2 ( $2^2=4$ )	Bit 1 ( $2^1=2$ )	Bit 0 ( $2^0=1$ )	Dezimalzahl
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	-8
1	0	0	1	-7
1	0	1	0	-6
1	0	1	1	-5
1	1	0	0	-4
1	1	0	1	-3
1	1	1	0	-2
1	1	1	1	-1

## Zweierkomplement

- Höchstwertiges Bit wird für Vorzeichen verwendet
- 0 = positive Zahl, 1 = negative Zahl

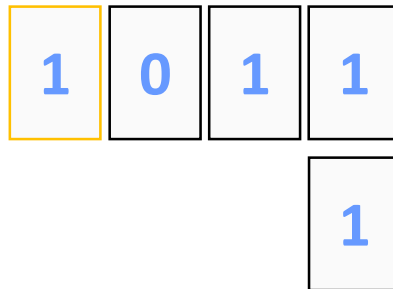


Zahlenkreis vierstelliger Dualzahlen

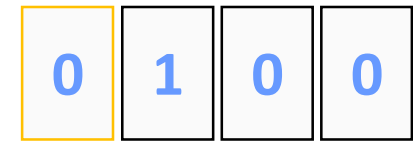
Quelle: [http://www.info-wsf.de/index.php/Rechnen\\_im\\_Bin%C3%A4rsystem](http://www.info-wsf.de/index.php/Rechnen_im_Bin%C3%A4rsystem)

Subtraktion:  $7 - 4 = 7 + (-4) = 3$

Einerkomplement  
zu 4



Zahl 4



+

1 1

Zweierkomplement  
zu 4

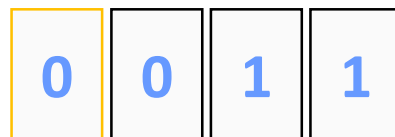


Zahl 7

+



1



Ergebnis: Zahl 3

- Multiplikation
- Division
- Siehe: <http://de.wikipedia.org/wiki/Dualsystem>

## Ganzzahlige Datentypen (Integer / Integrale Typen)

- Abbildung von ganzen Zahlen (...,-2,-1,0,1,2,3,...)
- Sowohl positive als auch negative Zahlen möglich
- Wertebereich wird durch die Anzahl der Bits festgelegt
- Übliche Bitgrößen:
  - 8Bit (=1 Byte)      byte
  - 16Bit (=2 Byte)    short
  - 32Bit (=4 Byte)    int
  - 64Bit (=8 Byte)    long
- Literale: Dezimal-, Oktal- (0), Hexadezimalschreibweise (0x)



Hexadezimalziffern,  
binär und dezimal:

Hex.	Dualsystem	Dez.
0	0 0 0 0	00
1	0 0 0 1	01
2	0 0 1 0	02
3	0 0 1 1	03
4	0 1 0 0	04
5	0 1 0 1	05
6	0 1 1 0	06
7	0 1 1 1	07
8	1 0 0 0	08
9	1 0 0 1	09
A	1 0 1 0	10
B	1 0 1 1	11
C	1 1 0 0	12
D	1 1 0 1	13
E	1 1 1 0	14
F	1 1 1 1	15

Gezählt wird wie folgt:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
FF0	FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FFA	FFB	FFC	FFD	FFE	FFF
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
FFF0	FFF1	FFF2	FFF3	FFF4	FFF5	FFF6	FFF7	FFF8	FFF9	FFFA	FFFB	FFFC	FFFD	FFFE	FFFF
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Siehe: <http://de.wikipedia.org/wiki/Hexadezimalsystem>

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	\$ 0024	% 0025	& 0026	' 0027	( 0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	<b>N 004E</b>	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[ 005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	 007C	}	~ 007E	DEL 007F
80	€ 20AC		/ 201A	f 0132	// 201E	... 2026	† 2020	‡ 2021	^ 02C6	‰ 2030	Š 0160	< 2033	Œ 0152		Ž 017D	
90		\ 2018	/ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	~ 02DC	™ 2122	Š 0161	> 203A	œ 0153		ž 017E	ÿ 0178
A0	NBSP 00A0	ı 00A1	ć 00A2	£ 00A3	¤ 00A4	¥ 00A5	ı 00A6	Š 00A7	™ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	– 00AD	® 00AE	— 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

**Buchstabe N**  
**(Hex) 004E**  
**Zeichen Nr. 78**

# Grunddatentypen (Fortsetzung)

## Fließkommazahlen

- **Approximative** Abbildung von reellen Zahlen
- Sind Teilmenge der rationalen Zahlen
- Genauigkeit wird durch die Anzahl der Bits festgelegt.
- Häufigste Form der Kodierung [IEEE 754](#)
- Übliche Bitgrößen
  - 32Bit (single precision)
  - 64Bit (double precision)

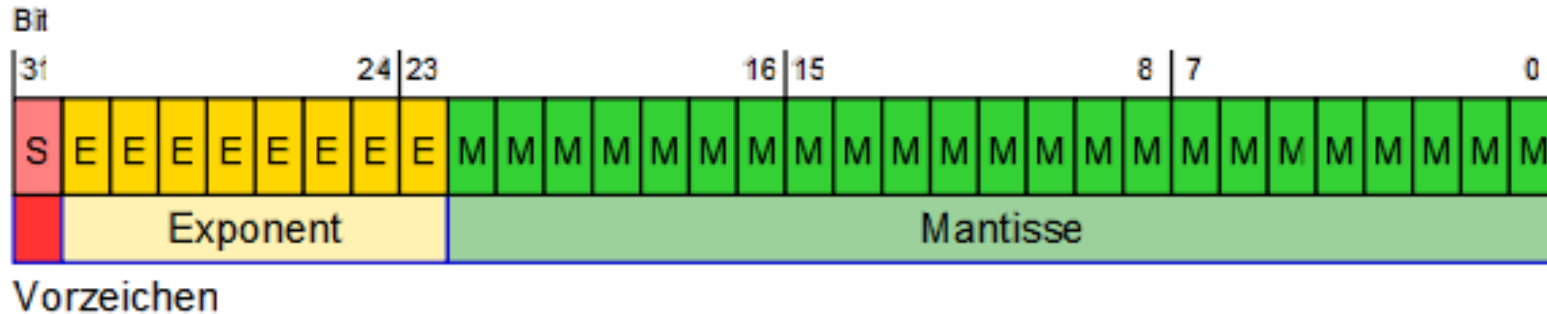
## Darstellung von Zahlen in Exponenten-Schreibweise:

- Beispiel:  $32000,00 = 32 * 10^3$
- Mantisse: 32
- Basis: 10
- Exponent: 3

Normalisierung: Der Exponent wird so gewählt, dass die Zahl eine bestimmte Form hat, z. B.

- $3.2 * 10^4$

- Aufteilung der Bits nach IEEE 754



- Fließkommazahlen in Java

Java-Datentyp	Anzahl der Bits	Wertebereich
float	32	$-3.403 \cdot 10^{38} \dots + 3.403 \cdot 10^{38}$
double	64	$-1.798 \cdot 10^{308} \dots + 1.798 \cdot 10^{308}$

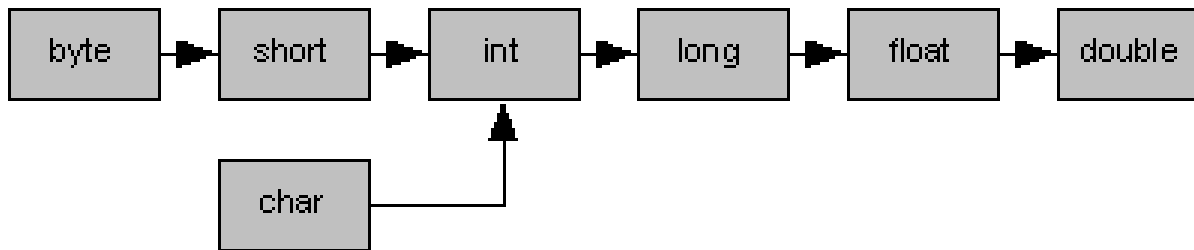
## Der Zeichenkettentyp (String)

- Abbildung von Wörtern und Sätzen
- Verkettung von mehreren Zeichen (char)

Java-Datentyp	Wertebereich
String	Beliebige aneinandergefügte Zeichen: <i>„The quick brown fox jumped over the lazy dog“</i>

## Typkonvertierungen (in Java)

- Einschränkende und erweiternde Konvertierungen

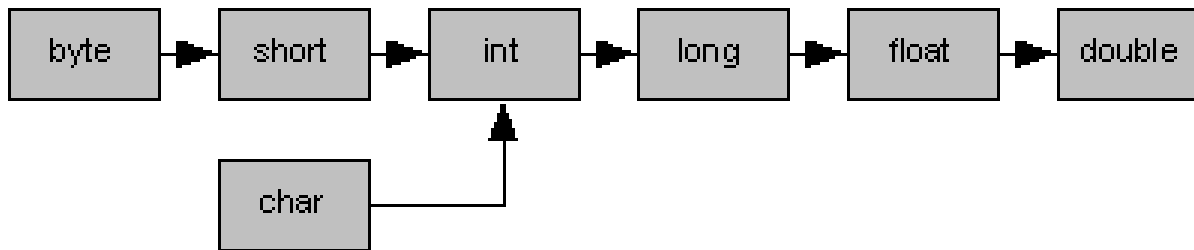


- Erweiternde Konvertierungen in Pfeilrichtung
- Einschränkende Konvertierungen entgegen der Pfeilrichtung
- Besonderheit: Es gibt keine Konvertierung von/zu boolean



## Typkonvertierungen (in Java)

- Einschränkende und erweiternde Konvertierungen



- Erweiternde Konvertierungen in Pfeilrichtung
- Einschränkende Konvertierungen entgegen der Pfeilrichtung
- Besonderheit: Es gibt keine Konvertierung von/zu boolean

# Erweiternde Konvertierungen

```
1 public class PrimitiveTypeConversions {  
2  
3     public static void main(String[] args) {  
4  
5         // erweiternde Konvertierungen  
6         byte byteX      = 7;  
7         short shortX    = byteX;  
8         int intX        = shortX;  
9         long longX      = intX;  
10        float floatX    = longX;  
11        double doubleX  = floatX;  
12  
13        char charY      = 'y';  
14        int intY        = charY;  
15  
16        // Überspringen einzelner Elemente  
17        long longZ      = byteX;  
18    }  
19 }
```

# Einschränkende Konvertierungen

```
1 public class PrimitiveTypeConversions {  
2  
3     public static void main(String[] args) {  
4  
5         // einschränkende Konvertierungen  
6         double doubleX = 7.0;  
7         float floatX   = (float) doubleX;  
8         long longX     = (long) floatX;  
9         int intX       = (int) longX;  
10        short shortX   = (short) intX;  
11        byte byteX     = (byte) shortX;  
12  
13        int intY        = 65;  
14        char charY     = (char) intY;  
15  
16        // Überspringen einzelner Elemente  
17        byte byteZ     = (byte) doubleX;  
18    }  
19 }
```

# Variablen

- **Zweck:** Speichern von Daten im Hauptspeicher zum späteren Lesen und Verändern
- Java kennt drei Arten von Variablen
  - 1- Instanzvariablen
  - 2- Klassenvariablen
  - 3- Lokale Variablen
- Java Variablen sind typisiert, d.h. der Typ einer Variable wird explizit vom Entwickler festgelegt und bei der Kompilierung des Programms überprüft.

- Deklaration von Variablen

```
Typname Variablenname;
```

- Es wird eine Variable vom Typ Typname mit dem Namen Variablenname erstellt.
- Variablen können in JAVA bei der Deklaration bereits initialisiert werden

```
Typname Variablenname = Wert;
```

## Beispiele

```
1  /* Listing0402.java */
2
3  public class Listing0402
4  {
5      public static void main(String[] args)
6      {
7          int a;
8          a = 1;
9          char b = 'x';
10         System.out.println(a);
11         double c = 3.1415;
12         System.out.println(b);
13         System.out.println(c);
14         boolean d = false;
15         System.out.println(d);
16     }
17 }
```

## Lebensdauer (in Java)

- Lokale Variablen
  - Beginn: Deklaration der Variable
  - Ende: Verlassen des Blocks, in dem die Variable deklariert worden ist
- [Instanzvariablen]
  - Beginn: Erzeugen einer neuen Objektinstanz einer Klasse
  - Ende: Zerstören der Objektinstanz
- [Klassenvariablen]
  - Beginn: Laden einer Klasse
  - Ende: Beenden des Programms



## Sichtbarkeit (in Java)

- Lokale Variablen
  - Sichtbar bis zum Ende des Blocks, in dem sie angelegt worden sind.
  - Sichtbar auch in Unterblöcken
- [Instanzvariablen]
  - Sichtbar innerhalb einer Klasse
  - Können von lokalen Variablen verdeckt werden (eine lokale Variable mit dem Namen „test“ hat Vorrang vor der Instanzvariable mit dem Namen „test“)
- [Klassenvariablen]
  - Entspricht der Sichtbarkeit von Instanzvariablen

# Arrays

## Übersetzt bedeutet Array

- Anordnung
- Reihe
- Bereich
- Feld (IT)

## Erklärung laut [Wikipedia](#):

„Mit Hilfe eines Feldes können **Daten** eines üblicherweise **einheitlichen Datentyps** so im Speicher eines Computers abgelegt werden, dass ein Zugriff auf die Daten über einen **Index** möglich wird.“

## Deklaration eines Arrays in Java

- Ähnlich der Deklaration einer Variablen mit dem Unterschied, dass eckige Klammern an den Typnamen angefügt werden

Typname [] Arrayname;

```
1 int[] a;  
2 double[] b;  
3 boolean[] c;
```

## Initialisierung eines Arrays in Java

- Legt die Größe fest
- Initialisierung über den sog. `new` Operator

`Arrayname = new Typname [Größe];`

```
1 a = new int[5];  
2 b = new double[10];  
3 c = new boolean[15];
```

```
1 int[] a = new int[5];  
2 double[] b = new double[10];  
3 boolean[] c = new boolean[15];
```

## Initialisierung eines Arrays in Java

- Literale Initialisierung

```
Typname[] Arrayname = {x,y,z};
```

```
1 int[] x = {1,2,3,4,5};  
2 double[] y = {1.0,2.0,3.0};  
3 boolean[] z = { true, true };  
4
```

## Zugriff auf die Elemente eines Arrays

- Bei der Initialisierung eines Arrays werden die Elemente von 0 bis  $n-1$  durchnummeriert ( $n$ =Anzahl der Elemente)
- Zugriff erfolgt über den so erstellten numerischen Index
- Indexnummer wird in eckigen Klammern angegeben

Zuweisen:      `Arrayname[index] = Wert;`

Lesen:          `Variablenname = Arrayname[index];`

# Array Zugriff

```
1 public class Listing0408 {  
2     public static void main(String[] args) {  
3         int[] prim = new int[5];  
4         boolean[] b = {true, false};  
5         prim[0] = 2;  
6         prim[1] = 3;  
7         prim[2] = 5;  
8         prim[3] = 7;  
9         prim[4] = 11;  
10  
11         System.out.println("prim hat "+prim.length+" Elemente");  
12         System.out.println("b hat "+b.length+" Elemente");  
13         System.out.println(prim[0]);  
14         System.out.println(prim[1]);  
15         System.out.println(prim[2]);  
16         System.out.println(prim[3]);  
17         System.out.println(prim[4]);  
18         System.out.println(b[0]);  
19         System.out.println(b[1]);  
20     }  
21 }
```



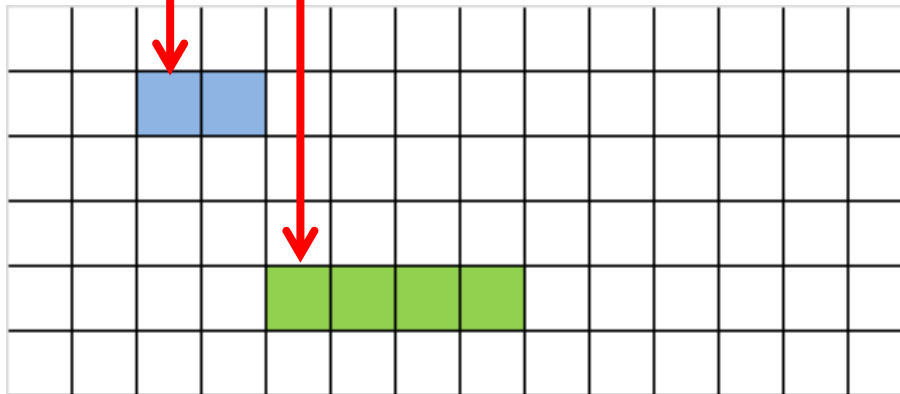
## Zweck: Speichern von Daten im Hauptspeicher

```
short aNumber = 2;
```

```
int anotherNumber = 99;
```

Deklaration

Zuweisung

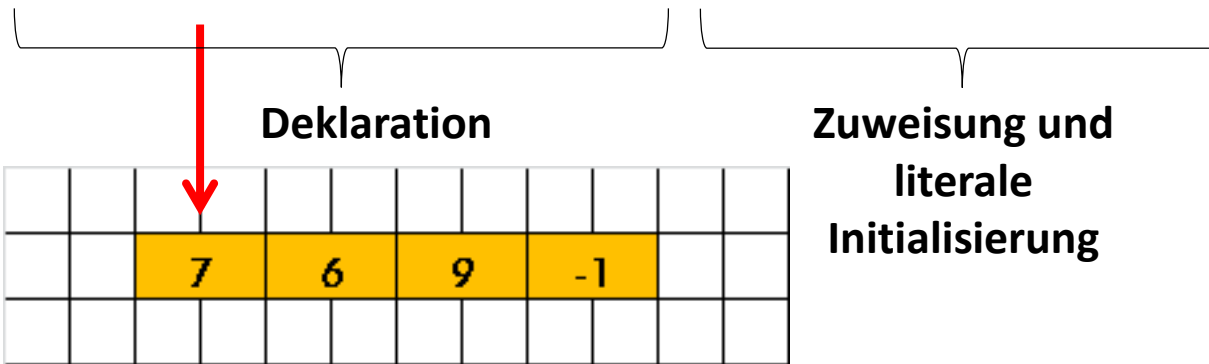


Arbeitsspeicher (1 Byte pro Zelle)

[Class7.java]

- Arrays sind eine Reihung (vgl. Liste) von Elementen eines Datentyps
- Arrays haben eine feste Größe, die man zur Laufzeit festlegen kann (Initialisierung auf eine Länge)

```
short[] aNumberArray = {7, 6, 9, -1};
```

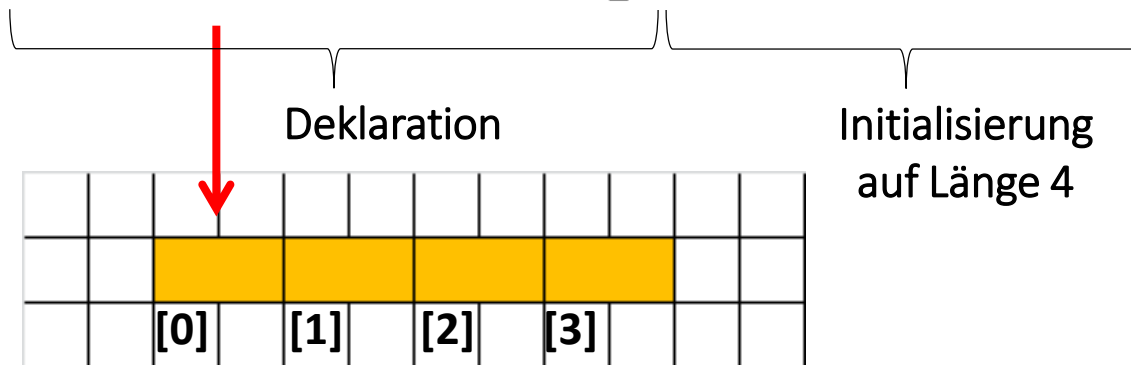


Arbeitsspeicher (1 Byte pro Zelle)

[Class7.java]

- Initialisierung mit dem new Operator

```
short[] bNumberArray = new short[4];
```



Arbeitsspeicher (1 Byte pro Zelle)

- Zuweisung von Elementen mit Hilfe des Index

```
bNumberArray[0] = 7;  
bNumberArray[1] = 6;  
bNumberArray[2] = 9;  
bNumberArray[3] = -1;
```

- Auf ein Array Zugreifen:

7	6	9	-1
---	---	---	----

Index    [0]    [1]    [2]    [3]

```
System.out.println(bNumberArray[0]);
```

```
System.out.println(bNumberArray[2]);
```

```
short elementTwo = bNumberArray[2];
```

- Länge des Arrays abfragen:    `bNumberArray.length`
- Was tun, wenn die Array-Länge nicht ausreicht?

[Class7.java]

Array-Inhalt als String darstellen:

```
short[] array = {1, 2, 3, 4};  
String string = Arrays.toString(array);
```

Arrays kopieren und dabei die Länge ändern:

```
short[] longerArray = Arrays.copyOf(array, 8);
```

Arrays auffüllen:

```
Arrays.fill(longerArray, (short) 99);
```

Das Parameter-Array der main-Methode:

```
public static void main(String[] args) {
```

```
C:\Users\wot39648\workspace\PG1 project\src>java Class8
Ausgabe des Arrays (ohne Konvertierung):
[Ljava.lang.String;@19821f
Ausgabe des Arrays (mit Konvertierung):
[1, 2, 3, 4]
Ausgabe des System-Parameter-Arrays (mit Konvertierung):
[]

C:\Users\wot39648\workspace\PG1 project\src>java Class8 15 Test
Ausgabe des Arrays (ohne Konvertierung):
[Ljava.lang.String;@19821f
Ausgabe des Arrays (mit Konvertierung):
[1, 2, 3, 4]
Ausgabe des System-Parameter-Arrays (mit Konvertierung):
[15, Test]
```

## Zweidimensionale Arrays

7	6	9	-1
[0][0]	[0][1]	[0][2]	[0][3]
-10	3	2	-3
[1][0]	[1][1]	[1][2]	[1][3]
0	4	-6	9
[2][0]	[2][1]	[2][2]	[2][3]

```
int[][] twoDim = new int[3][4];  
twoDim[0][0] = 7;  
twoDim[0][1] = 6;  
twoDim[0][2] = 9;  
twoDim[0][3] = -1;  
  
twoDim[1][0] = -10;  
twoDim[1][1] = 3;  
twoDim[1][2] = 2;  
twoDim[1][3] = -3;  
  
twoDim[2][0] = 0;  
twoDim[2][1] = 4;  
twoDim[2][2] = -6;  
twoDim[2][3] = 9;
```

[\[Class9.java\]](#)

## Verschiedene Sub-Array-Längen

7	6	9	-1
[0][0]	[0][1]	[0][2]	[0][3]
-10	3		
[1][0]	[1][1]		
0	4	-6	
[2][0]	[2][1]	[2][2]	

### Literale Initialisierung

```
int[][] otherArray =  
{ {7,6,9,-1}, {-10,3}, {0,4,-6} };
```

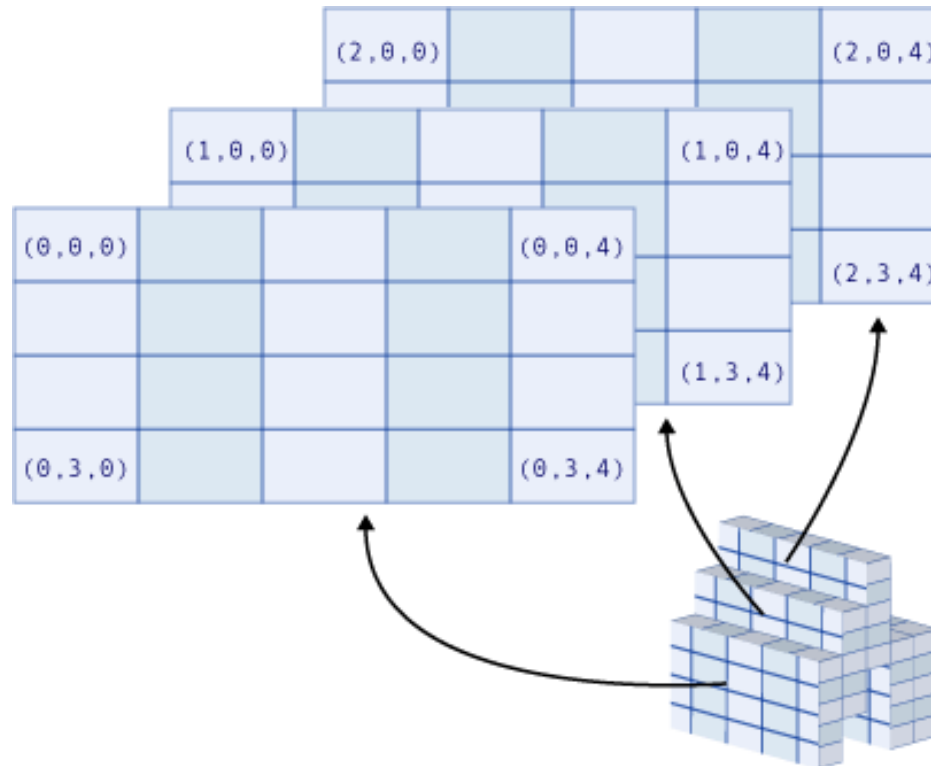
### String-Darstellung deepToString

```
Arrays.deepToString(otherArray);
```

[\[Class9.java\]](#)



## Drei Dimensionen, usw.



Quelle: Microsoft

```
String[][][] threeDimString = new String[3][5][4];
```

[\[Class9.java\]](#)