Objektorientierte Programmierung

Datentypen und Variablen

Prof. Dr. Ulrike Hammerschall
Fakultät für Informatik und Mathematik

Variablen und Datentypen

- Eine Variable ist ein Platzhalter (Zwischenspeicher) für einen beliebigen Wert in einem Programm.
- Eine Variable hat einen Namen und einen Datentypen:
  - Name: Eindeutiger Identifikator für Variable
  - Datentyp: Art der Werte, die im Speicherbereich der Variable abgelegt werden dürfen.
- Technisch gesehen referenziert eine Variable einen konkreten Speicherbereich im Hauptspeicher, in dem ein Wert von einem bestimmten Typ abgelegt werden kann.

Variablen - Definition, Wertzuweisung und Initialisierung

- Definition von Variablen
  - Festlegung des Namens und Zuordnung des Datentyps.
  - `int max;`
- Wertzuweisung
  - Zuweisung eines Werts zu einer (zuvor definierten) Variable.
  - `int max;`
  - `max = 100;`
- Initialisierung einer Variable: Definition und Zuweisung.
  - `int max = 100;`

Syntaktische Regeln für Variablennamen

- Syntaktische Regeln für Variablennamen
  - Große und kleine Buchstaben, Dezimalziffern und Unterstrich.
  - Der erste Buchstabe darf keine Ziffer sein.
  - Leerzeichen im Namen sind nicht erlaubt.
  - Etwa fünfzig reservierte Wörter dürfen nicht als Bezeichner benutzt werden (beispielsweise *class*, *int*, *public*, *static*, …).
- Konventionen für Variablennamen
  - Beginn immer mit Kleinbuchstaben.
  - Neue Wörter im Namen beginnen mit einem Großbuchstaben (camelCase).
  - Sprechende Variablennamen verwenden.
  - Mehrdeutige Abkürzungen ausschreiben.
  - Englische Begriffe verwenden.

- Diese Regeln sind Teil der Grammatik.
- Werden vom Compiler überprüft!

- Allgemeine Coding Conventions für Java Programme.
- Empfehlungen, die helfen Java-Programme einheitlich zu gestalten und besser lesbar zu machen.
- Können ggf. über Stylechecker geprüft werden können.

Datentypen in Java

- Ein Datentyp legt die Darstellung eines Wertes im Speicher des Computers fest.
- Primitive Datentypen (Basisdatentypen):
  - Fest definierte Datentypen, die von der Sprache selbst unterstützt werden. Sie können nicht verändert werden.
- Referenztypen:
  - Jede Klasse in einer Bibliothek oder in einem selbstgeschriebenen Programm ist ein Referenztyp.
- Java als typisierte (typsichere) Sprache:
  - Jede Variable erhält bei der Definition einen festen Datentypen.
  - Dieser kann innerhalb des Programms nicht mehr verändert werden.
  - Der Compiler prüft, ob der zugewiesene Wert kompatibel zum Datentyp ist.

Primitive Datentypen in Java

Primitive Datentypen in Java	Werte
int, byte, short, long	Positive oder negative Ganzzahl (Für Berechnungen mit großen Ganzzahlen gibt es die Hilfsklasse BigInteger[1)])
double, float	Gleitkommazahlen (Für Berechnungen mit großen Werten gibt es die Hilfsklasse BigDecimal[2)])
char	Einzelnes Zeichen (Charakter). Kann Buchstabe, Ziffer oder Sonderzeichen sein.
String-Literale	Text beliebiger Länge. Im Kern eine Aneinanderreihung von Zeichen (Charakter).
boolean	Wahrheitswert (true oder false)

1) https://openbook.rheinwerk-verlag.de/javainsel/21_006.html#u21.6.1
2) https://openbook.rheinwerk-verlag.de/javainsel/21_006.html#u21.6.3

Positive und negative Ganzzahlen

Datentyp	Bits im Speicher	kleinster Wert	größter Wert
int	32	-2^{31}	$2^{31} - 1$
long	64	-2^{63}	$2^{63} - 1$
short	16	-2^{15}	$2^{15} - 1$
byte	8	-2^{7}	$2^{7} - 1$
Allgemein:	n	-2^{n-1}	$2^{n-1} - 1$

- Ein Zahlenwert ohne Punkt (ohne Typangabe) wird in Java automatisch als int interpretiert.

```
int max = 100;
```


Einschub - Speicherdarstellung von Integer

- Der Wertebereich vom Datentyp *int* wird festgelegt durch die Zweierkomplementdarstellung im Speicher.
- Zweierkomplement zur Darstellung von negativen Zahlen: Arithmetische Operation auf Binärzahlen
- Beispiel für n = 4 => Wertebereich: -2^3 bis $2^3 - 1$
  - Binärdarstellung der Zahl 5:
  - Inversion der Zahl
  - Addition von 1
  - Binärdarstellung der Zahl -5

```
0101  (0*2^3 + 1*2^2 + 0*2^1 + 1*2^0)
1010
+0001
1011
```

➢ Was geschieht wenn der Wertebereich überschritten wird?

Zweier-komplement	Dezimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Gleitkommazahlen

HM

Datentyp	Bits im Speicher	kleinster und größter negativer Wert	kleinster und größter positiver Wert
double	64	$-1.79769 * 10^{308}$	$+4.94065 * 10^{-324}$
		$-4.94065 * 10^{-324}$	$+1.79769 * 10^{308}$
float	32	$3.40282 * 10^{38}$	$-1.4 * 10^{-45}$
		$+1.4 * 10^{-45}$	$+3.40282 * 10^{38}$

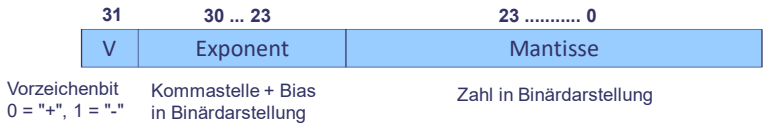
Ein Zahlenwert mit Punkt (ohne Typangabe) wird in Java automatisch als double interpretiert.

```
double price = 100.0;
```


Einschub - Speicherdarstellung von Gleitkommazahlen

HM

- Darstellung von positiven / negativen reellen Zahlen
  - **double**: 64 Bit im Speicher (double precision)
  - **float**: 32 Bit im Speicher (single precision)
- Abbildung im Speicher (float):



- Entspricht der Darstellung:
  (+/-) Mantisse * $2^{Exponent}$
- Beispiel[1)]: Speicherdarstellung von `6,75` wäre `0 1000000 11011000000000000000000`

1) Eine schöne Erklärung finden sie hier https://www.h-schmidt.net/FloatConverter/IEEE754de.html

Arbeiten mit numerischen Datentypen

- Varianten bei der Schreibweise bei double:

```
double value = 20.5;
double value = 0.0205E3;
double value = 205000E-4;
```

- Zuordnung des Datentyps zu Werten:

```
20      => 20 wird hier als int gespeichert
20.0    => 20 wird hier als double gespeichert
20d     => 20 wird hier als double gespeichert
20f     => 20 wird hier als float gespeichert
20l     => 20 wird hier als long gespeichert
```


Implizite Typkonversion

- Implizite Typkonversion findet immer statt, wenn:
  - ein Wert in einem bestimmten Typ benötigt wird,
  - der Wert nicht in diesem Typ vorliegt und
  - die Typkonversion ohne Informationsverlust möglich ist.
- Übung: Bei welcher Konversion würde kein Informationsverlust auftreten, d.h. eine implizite Typkonversion ist möglich (die Anweisung ist korrekt)?

1) int nach double:	`double price = 3;`
2) double nach int:	`int max = 3.0;`
3) int nach float:	`float  = 3;`
4) float nach int:	`int degree = 3f;`
5) long nach int:	`int number = 3l;`
6) int nach long:	`long seconds = 3;`

Character - Darstellung von einzelnen Zeichen

HM

- Primitiver Typ "char" (engl. "character") repräsentiert einzelne Textzeichen (Buchstaben oder Sonderzeichen).

```
char c = 'a';  // der Buchstabe a
char c = '5';  // das Zeichen 5
int i = 5;     // die Zahl 5
char c = '%';  // das Prozent-Zeichen
char c = ' ';  // das Leerzeichen
```

- Jeder Charakter wird intern auf eine positive Zahl (Unicode) abgebildet:

```
'a'      =>    97
'5'      =>    53
```


Einschub – Encoding von Zeichen

HM

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.overcoded.net

Implizite Typkonversion int und char

- Abbildung von Charakter auf eindeutige Zahlenwerte erlaubt teilweise implizite Typkonversion zwischen char und int:

```
int i = 'i';                // i = 105
int a = 'a';                // a = 97
int v = '5';                // v = 53
char c = 97;                // c = 'a'
int j = c;                  // j = 97
char x = -2                 // Fehler
```

Übung: Welchen Wert hat die Variable *value* in folgender Anweisung?

```
int value = a  +  v  + 5  + c;
```


Darstellung von Text mit String

- Der Datentyp "String" dient zur Darstellung von beliebig langen Zeichenfolgen.
- Initialisierung von Strings (String-Literale)

```
String word = "Haus"
String text = "Geh ins Haus!"
```

- Verkettung von Strings mit + Operator:

```
String compiler = "Java-" + "Compiler ";
```

- Verkettung von String mit anderem Datentyp liefert wieder einen String

```
int version = 17;
String text = "Wir arbeiten mit Java " + version;
```

Wahrheitswerte

- Auch als Boolesche Werte bezeichnet. Ursprung ist die Boolesche Algebra von George Boole.
- Wertebereich enthält genau zwei Werte:
  - true (trifft zu)
  - false (trifft nicht zu)
- Keine gültigen Werte sind: wahr/falsch oder 0/1
- Beispiele zur Initialisierung von Boolean-Variablen:

```
boolean isEmpty = true;
boolean isReady = false;
```


Arithmetische Operatoren

Operator	Mathematik	Java	Priorität der Auswertung	Reihenfolge bei Auswertung der Operatoren	Operanden
Pos. Vorzeichen	+1	+1	1	Rechts	1
Neg. Vorzeichen	-1	-1	1	Rechts	1
Multiplikation	1(1 * 1)	1 * (1 * 1)	2	Links	2
Division	$\frac{1}{2}$	1 / 2	2	Links	2
Modulo Operator	1 mod 1	1 % 1	2	Links	2
Addition	1 + 1	1 + 1	3	Links	2
Subtraktion	1 – 1	1 – 1	3	Links	2

Ganzzahlige Division versus normale Division

- Der Divisionsoperator / verhält sich unterschiedlich, je nach Typ seiner Operanden!
- Ganzzahlige Division: Beide Operanden sind vom Typ *int*. Der nicht-ganzzahlige Rest wird abgeschnitten:

```
5 / 2        =>    2
```

- Normale Division: Beide Operanden sind vom Typ double (oder float):

```
5.0 / 2.0    =>    2.5
```


Ganzzahlige Division und Modulo

- Ganzzahlige Division wird häufig in Kombination mit dem Modulo Operator verwendet.
- Fragestellungen, die man lösen möchte:

```
x / y    (wie oft passt y vollständig in x?)
x % y    (was bleibt danach noch übrig?)
```

Beispiel: Sie haben 20 Meter Tapete und eine Wand von 3 Metern Höhe. Wie viele Bahnen bekommen Sie aus der Tapete und wieviel Verschnitt bleibt? Ausgedrückt in Java:

```
int wallpaper = 20;
int wall = 3;
int parts = 20 / 3;
int rest = 20 % 3;
```

Zusammenfassung

- Variablen sind Platzhalter für Werte. Datentypen legen fest, wie diese Werte im Speicher abgelegt werden.
- Java kennt nur wenige Basisdatentypen. Diese sind fest in der Sprache verankert. Alle anderen Datentypen sind Referenztypen (Klassen).
- Bei impliziter Typkonversion wandelt der Compiler automatische Werte in einen bestimmten Datentyp in einen anderen um. Dies ist nur möglich, wenn kein Informationsverlust erfolgt.
- Java kennt die typischen arithmetischen Operatoren. Diese erlauben die Formulierung von üblichen mathematischen Algorithmen.
- Anwendung und Auswertung orientiert sich weitgehend an den Regeln der Mathematik. Nur die Schreibweise weicht teilweise ab.