4.3 Die Datentypen char und String

- Ein weiterer einfacher Datentyp ist der Datentyp **`char`**.
  - Der Datentyp repräsentiert ganze positive Zahlen im Bereich 0,1,…,65535 (= 2^{16}-1). Es gelten also die typischen Regeln für ganze Zahlen:

```
char ch1 = 87;
int intch1 = ch1;
char ch3 = (char) intch1;
```

  - Zudem stehen als Operationen ++, -- und die Zuweisungsoperatoren =, +=, -=, … zur Verfügung.
- Es gibt aber **Unterschiede** zu den anderen ganzzahligen Datentypen.
  - Als Literale stehen zusätzlich noch die Zeichen des UCS/Unicode-Zeichensatz zur Verfügung.

```
char ch2 = 'x';
```

Das Zeichen steht zwischen 2 Apostrophen!

  - Zudem wird bei der Ausgabe eines Werts vom Typ char durch System.out.print das Zeichen des Unicode-Zeichensatzes verwendet.

Zeichensätze

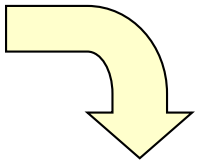
- Java nutzt den UCS/Unicode-Zeichensatz, der neben den lateinischen Schriftzeichen auch kyrillische, chinesische, japanische, koreanische und zahlreiche weitere Zeichensätze enthält.
- Die ersten 256 Zeichen entsprechen dem ASCII-Zeichensatz

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	à	166	ª	199	Ã	232	Þ
002	Start Of Text	035	#	068	D	101	e	134	å	167	º	200	╚	233	Ú
003	End Of Text	036	$	069	E	102	f	135	ç	168	¿	201	╔	234	Û
004	End Of Transmission	037	%	070	F	103	g	136	ê	169	®	202	╩	235	Ù
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	ï	172	¼	205	═	238	¯
008	Backspace	041	)	074	J	107	k	140	î	173	¡	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	Ê	243	¾
013	Carriage Return	046	.	079	O	112	p	145	æ	178	▓	211	Ë	244	¶
014	Shift Out	047	/	080	P	113	q	146	Æ	179	│	212	È	245	§
015	Shift In	048	0	081	Q	114	r	147	ô	180	┤	213	ı	246	÷
016	Delete	049	1	082	R	115	s	148	ö	181	Á	214	Í	247	¸
017	-- frei --	050	2	083	S	116	t	149	ò	182	Â	215	Î	248	°
018	-- frei --	051	3	084	T	117	u	150	û	183	À	216	Ï	249	¨
019	-- frei --	052	4	085	U	118	v	151	ù	184	©	217	┘	250	·
020	-- frei --	053	5	086	V	119	w	152	ÿ	185	╣	218	┌	251	¹
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	Ü	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	Ì	255	
025	End Of Medium	058	:	091	[	124	|	157	Ø	190	¥	223	▀		
026	Substitude	059	;	092	\	125	}	158	×	191	┐	224	Ó		
027	Escape	060	<	093	]	126	~	159	ƒ	192	└	225	ß		
028	File Seperator	061	=	094	^	127	⌂	160	á	193	┴	226	Ô		
029	Group Seperator	062	>	095	_	128	Ç	161	Í	194	┬	227	Ò		
030	Record Seperator	063	?	096	`	129	ü	162	ó	195	├	228	õ		
031	Unit Seperator	064	@	097	a	130	é	163	ú	196	─	229	Õ		
032		065	A	098	b	131	â	164	ñ	197	┼	230	µ		

Operationen auf char-Datentypen

- Inkrement/Dekrement-Operationen ++ und -- liefern Vorgänger und Nachfolger-Zeichen
- Zuweisungsoperationen (mit ganzer Zahl): +=, -=, ….
- Vergleichs-Operationen ==, !=, <, <=, >, >=
- Beispiel:

```
char c = 40;
while (c <= '9')
  System.out.print(c++);
System.out.println();
```



```
()*+,-./0123456789
```

- Warnung:

```
c++;  // ok
c += 1; // ok
c = c + 1; // nicht ok
```

Im 3. Fall bekommt die rechte Seite durch Typanpassung den Typ ***int***, der ***nicht*** implizit an den Typ ***char*** angepasst wird!

String-Datentypen in Programmiersprachen

- Der Typ String ist ein erstes Beispiel eines zusammengesetzten Typs.
  - Ein String ist eine Zeichenkette und besteht aus einer Folge von Zeichen (von Werten des Datentyps `char`).
- Mit der Menge aller solcher Zeichenketten wird die Datentyp `String` gebildet.
- Operationen sind z.B.:

`+: String × String → String`	Konkatenation
`length: String → int`	Stringlänge
`charAt: String × int → char`	Zugriff auf Zeichen
`indexOf: String × char → int`	Position eines Zeichens

Zeichen- und Zeichenketten-Literale

- Ein Zeichen-Literal (char-Literal) ist ein einzelnes, in einfache Apostrophe eingeschlossenes Unicode-Zeichen:

```
'a' '%' 't' 'a' 'W' 'α' 'Ω' 'æ' 'ç' 'Ã' 'ß'
```

- Ein Zeichenketten-Literal (String-Literal) ist eine Folge von Unicode-Zeichen, in Doppel-Apostrophen:

```
"Dies ist ein String."
```

- Ein String-Literal muss auf genau einer Zeile beginnen und enden.
- Allerdings können String-Literale mit dem + Operator verkettet (konkateniert) werden. Sie bilden dann ein zusammengefasstes String-Literal.
- Beispiele für weitere String-Literale:

```
"Dies ist ein String, der auf " +
  "zwei Zeilen verteilt wurde."
```

```
"Tar" + "tar" + " ist " +
    "keine Käsesorte!"
```

```
"Ввгдтњ Юњяы Швгд Итњяыш Жл"
```

Ersatzdarstellungen

- In Zeichen- oder String-Literalen können bzw. müssen Ersatzdarstellungen benutzt werden.
- Falls das eingeschlossene Zeichen selbst ein Apostroph oder ein \ sein soll, oder ein nicht-druckbares Zeichen ist, muss eine der folgenden Ersatzdarstellungen, auch Escape-Sequenzen genannt, verwendet werden.

\" für ein "
\' für ein '
**\\ für ein **
\t für einen horizontalen Tabulator (HT: ASCII-Wert 9)
\n für einen Zeilenwechsel (LF: ASCII-Wert 10)

Weitere Ersatzdarstellungen:

\b für einen Rückwärtsschritt (BS: ASCII-Wert 8)
\f für einen Formularvorschub (FF: ASCII-Wert 12)
\r für einen Wagenrücklauf (CR: ASCII-Wert 13)

```
'\t' '\\' '\''
```

```
"\tDieser Text\r\n\twurde formatiert."
```

Aufruf von Operationen

- Der +-Operator verbinden zwei Zeichenketten zu einer neuen Zeichenkette.
  - String str1 = "Uni";
    String str2 = "Marburg";
    String str3 = str1 + " " + str2;
- Die anderen Operationen werden anders aufgerufen.
  - Der erste Parameter (vom Typ String) kommt zuerst, dann folgt ein Punkt und der Aufruf der Methode ohne den ersten Parameter.
  - Beispiele:
    - int len = str1.length(); // Liefert die Länge der Zeichenkette str1
    - char ch = str2.charAt(3); // Liefert das Zeichen an Position 3
    - int pos = str3.indexOf(' '); // Liefert die Position des Leerzeichens
  - Vor dem Punkt darf ein beliebiger Ausdruck vom Typ String stehen.
    - len = (str1 + str2).length(); // Liefert die Länge der beiden Zeichenkette str1 und str2

Beispiel: Mustersuche in Text

```
/** Prüft, ob ein Textmuster in einer Zeichenkette an einer Position vorkommt.
  * @param text Zeichenkette
  * @param pattern Muster der Länge l
  * @param pos Position
  * @return text[pos, pos + 1, ... ,pos + l - 1] == pattern
  */
 boolean isSubStringAtPosition(String text, String pattern, int pos) {
        int i = 0;
        while (i < pattern.length()) {
                if (i + pos < text.length() && text.charAt(i + pos) == pattern.charAt(i))
                        i = i + 1;
                else
                        return false;
        }
        return true;
 }
```

Zusammenfassung

- Datentypen
  - Wertemenge
  - Operationen
- Primitive Datentypen in Java
  - boolean, ganze Zahlen, Fließpunktzahlen
- Ausdruck und Anweisung
  - Ausdrucks hat einen Typ
  - Auswertung unter Verwendung von Prioritäten der Operatoren
  - Seiteneffekte in Java
- Typumwandlung
  - Explizite und implizite