Type char

The values of primitive type **char** are characters. There are no operations on characters. Characters can be written in several ways:

1. '$' A character within single quotes.
2. '\u0061' *Unicode representation* (explained below). This is character 'a'
3. '\n' An escape sequence within quotes. This is the new-line character.

Code points

Each **char** that you write in single quotes, like '$' and 'A', is represented by a 16-bit integer called its *code point*. The code point for 'A' is 65, the code point for 'B' is 66. Later, we show how to use casting to get the code point for a **char** and vice versa.

In a representation of a **char** like '\u0061', following the \u is a 4-digit hexadecimal number giving the code point of the character. Below are examples. The last five examples are Strings that show a greeting in the world's five most popular languages.

'\u0061' is 'a'
'\u00E4' is 'ä'
'\u03C3' is 'σ'

'\u0950' is 'ॐ'	Om, the sound of the universe (Sanskrit)
"\u4F60\u597D"	Chinese 你好, spoken as *ni hao*, meaning "a respectful hello"
"hola"	Spanish word for "hello"
"hello"	English word "hello"
"\u0928\u092E\u0938"	Sanskrit word नमस, or *namaste*, meaning "I bow to the divine in you"

"\u0627\u0644\u0633\u0644\u0627\u0645 \u0639\u0644\u064a\u0643\u0645"
Arabic words "السلام عليكم", or *as-salam alaykom*, meaning "peace be upon you"

Unicode and ASCII

Unicode encodes the characters of almost all the world's writing systems. Most characters are represented in 2 bytes (16 bits), but there are now so many characters that they don't all fit into the 16-bit format, and a some require two-character escape sequences. Unicode is not a fixed format given from heaven but a standard developed by the Unicode Consortium, a non-profit organization that works closely with other standards committees. It continues to evolve. Have a look at its website: www.unicode.org/. Get to know the history of character representations.

ASCII stands for *American Standard Code for Information Interchange*. This 7-bit representation of the standard Latin letters and digits and some punctuation and control characters was developed in the 1960s. It was based on an encoding used earlier in telegraph systems. Unicode uses the ASCII representation of the standard characters, with a few modifications. A list of these ASCII characters is given at the end of this document, on the next page.

Escape sequences

The escape sequences start with the backslash letter \. They include a few control characters and the sequences for a single quote, double quote, and the backslash itself. Here are the most frequently used ones.

'\t'	tab	'\n'	new-line	'\r'	carriage return
'\''	single quote	'\"'	double quote	'\\'	backslash

Casting to and from a character's code point

Type **char** is a number type, and a character can be cast to its code point and back. For example,

The value of (**char**) 65 is 'A' The value of (**int**) 'A' is 65

If a character appears in an arithmetic operation, it is automatically cast to its **int** representation. For example,

'A' + 1	evaluates to 66	(**char**) ('A' + 1)	evaluates to 'B'
'A' < 'B'	evaluates to true	'Z' +1 - 'A'	evaluates to 26

Type char

The tables to below gives the ASCII and Unicode representation of digits, Latin letters, punctuation, etc. For the numbers '0'.. '9', letters 'A'.. 'Z', and letters 'a'.. 'z', we show only the first and last since they follow in order. These code points can be used to process digits and letters in various ways. Here are examples.

Suppose character c contains a digit. To get this digit as an integer, use the expression c - '0'. Do *not* use the expression c - 48 because people don't remember that **(char)** 48 is '0'. Check out *magic numbers* in JavaHyperText.

The following loop prints all the lower-case Latin letters:

```
for (char k= 'a'; k <= 'z'; k= (char)(k+1))
   System.out.println(k);
```

Binary	Oct	Dec	Hex	Char
010 0000	040	32	20	space
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
…	…	…	…	…
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;

Binary	Oct	Dec	Hex	Char
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?
100 0000	100	64	40	@
100 0001	101	65	41	A
…	…	…	…	
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D	]
101 1110	136	94	5E	^
101 1111	137	95	5F	_
110 0000	140	96	60	‘
110 0001	141	97	61	a
…	…	…	…	
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~