char

reading: 4.3

Type `char`

- **`char`** : A primitive type representing single characters.
  - A `String` is stored internally as an array of `char`

```
String s = "nachos";
```

index	*0*	*1*	*2*	*3*	*4*	*5*
value	`'n'`	`'a'`	`'c'`	`'h'`	`'o'`	`'s'`

  - It is legal to have variables, parameters, returns of type `char`
    - surrounded with apostrophes: `'a'` or `'4'` or `'\n'` or `'\''`

```
char initial = 'J';
System.out.println(initial);              // J
System.out.println(initial + " Joyce");   // J Joyce
```

The `charAt` method

- The `char`s in a `String` can be accessed using the `charAt` method.
  - accepts an `int` index parameter and returns the `char` at that index

```
String food = "cookie";
char firstLetter = food.charAt(0);    // 'c'

System.out.println(firstLetter + " is for " + food);
```

- You can use a `for` loop to print or examine each character.

```
String major = "CSE";
for (int i = 0; i < major.length(); i++) {     // output:
    char c = major.charAt(i);                  // C
    System.out.println(c);                     // S
}                                              // E
```

Comparing `char` values

- You can compare `char` values with relational operators:

  `'a' < 'b'` and `'X' == 'X'` and `'Q' != 'q'`

  - An example that prints the alphabet:

```
for (char c = 'a'; c <= 'z'; c++) {
    System.out.print(c);
}
```

- You can test the value of a string's character:

```
String word = console.next();
if (word.charAt(word.length() - 1) == 's') {
    System.out.println(word + " is plural.");
}
```

`char` vs. `String`

- `"h"` is a `String`, but `'h'` is a `char` (they are different)
- A `String` is an object; it contains methods.

```
String s = "h";
s = s.toUpperCase();          // "H"
int len = s.length();         //  1
char first = s.charAt(0);     // 'H'
```

- A `char` is primitive; you can't call methods on it.

```
char c = 'h';
c = c.toUpperCase();                // ERROR
s = s.charAt(0).toUpperCase();      // ERROR
```

  - What is `s + 1` ? What is `c + 1` ?
  - What is `s + s` ? What is `c + c` ?

`char` vs. `int`

- Each `char` is mapped to an integer value internally
  - Called an **ASCII value**

`'A'` is 65	`'B'` is 66	`' '` is 32
`'a'` is 97	`'b'` is 98	`'*'` is 42

  - Doing "math" on a `char` causes automatic conversion to `int`.

    `'a' + 10` is 107, `'A' + 'A'` is 130

  - To convert an `int` into the equivalent `char`, type-cast it.

    `(char) ('a' + 2)` is `'c'`

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

Suppose I am getting a char from a string and want to tell if the char I extracted is alphabetic?

Can I do the following?

```
if ( c >= 'a' && c <= 'Z' )
```

…no because in the ascii table lowercase comes after upper
…*AND* there are other/punctuation characters between the
…two sets…

Best thing to do is get a copy of the string in lowercase and compare to the range of lowercase char:

```
String s1 = s.toLowerCase();
c = s1.charAt(i);
if ( c >= 'a' && c <= 'z')
```

`String/char` question

- A *Caesar cipher* is a simple encryption where a message is encoded by shifting each letter by a given amount.
  - e.g. with a shift of 3, A → D, H → K, X → A, and Z → C

- Write a program that reads a message from the user and performs a Caesar cipher on its letters:

Run 1:

```
Your secret message: Computer science is awesome
Your secret key: 3
The encoded message: frpsxwhu vflhqfh lv dzhvrph
```

Run 2:

```
Your secret message: abc xyz
Your secret key: 3
The encoded message: def abc
```

Run 3:

```
Your secret message: AbC xYz
Your secret key: -3
The encoded message: xyz uvw
```

Plan

- main()
  - Create scanner
  - Get message
  - Get key
  - Encode message
  - Display message
- encode()
  - Input: String msg, int key
  - Output: encoded string
  - Convert msg to lowercase
  - For every char in msg
    - Get next char
    - if (alphabetic) {
      - Shift char
      - If (shifted char > ‘z’)
        - Shifted char – 26
      - Else if (shifted char < ‘a’)
        - Shifted char + 26
    - Add to encoded string
  - Return encoded string

Strings answer part 1

```
// This program reads a message and a secret key from the user and
// encrypts the message using a Caesar cipher, shifting each letter.

import java.util.*;

public class SecretMessage {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("Your secret message: ");
        String message = console.nextLine();
        message = message.toLowerCase();

        System.out.print("Your secret key: ");
        int key = console.nextInt();

        System.out.println("The encoded message: " + encode(message, key);
    }

    ...
```

Strings answer part 2

```
    // This method encodes the given text string using a Caesar
    // cipher, shifting each letter by the given number of places.
    public static String encode(String text, int shift) {
        String lowercase = text.toLowerCase(); // gonna ignore case
        String encoded = “”; // build the encoded string in here
        for (int i = 0; i < lowercase.length(); i++) {
            char letter = lowercase.charAt(i);

            // shift only letters (leave other characters alone)
            if (letter >= 'a' && letter <= 'z') {
                letter += shift;

                // may need to wrap around (include a negative shift!)
                if (letter > 'z') {
                    letter -= 26;
                } else if (letter < 'a') {
                    letter += 26;
                }
            }
            encoded += letter;
        }
        return encoded;
    }
}
```