Strings

By: Megan Avery
Updated March 2019

Before We Begin

- Make sure you have Java & Java JDK downloaded on your computer
  - can run **`java -version`** in terminal to check
- Make sure you have IntelliJ downloaded on your computer

- *Suggested:* Watch previous Java tutorials

What are Strings?

Strings are another **type** of variable. They aren't primitive though, they are **objects** which means they have methods associated with them. Strings are a collection of characters.

Objects will be discussed more in depth in a later tutorial.

Declaring, Assigning, and Updating Strings

```
// declare and assign
String s = "example string";

// update s
s = s + " after update";
```

Indexes of Strings

Indexes are used to keep track of characters in a String.

Strings are 0 indexed, this means the first character in a String has an index of 0, the second has an index of 1, and so on. The index of the last character in a String is the length of that String minus 1.

Example:

index	0	1	2	3	4	5	6	7	8	9	10	11	12
	p	e	n	g	u	i	n	s		r	o	c	k

String Methods

Method name	Description
s.indexOf(str)	index where the start of the given string appears in this string (-1 if not found)
s.length()	number of characters in this string
s.replace(str1, str2)	replaces occurrences of str1 with str2
s.substring(index1, index2) **OR** s.substring(index1)	the characters in this string from index1 (inclusive) to index2 (exclusive); if index2 is omitted, grabs till end of string
s.toLowerCase()	a new string with all lowercase letters
s.toUpperCase()	a new string with all uppercase letters
s.charAt(index)	the char at the given index

Note: all these methods return Strings except s.charAt(index) returns a char

Under the Hood

Whenever a String method is called the String isn’t edited, a completely new String is created and returned. The same happens when a String is added to using a “+”. This is because in Java Strings are **immutable**.

String Test Methods

Method	Description
s.equals(str)	whether two strings contain the exact same characters
s.equalsIgnoreCase(str)	whether two strings contain the same characters, ignoring upper vs. lower case
s.startsWith(str)	whether one contains other's characters at start
s.endsWith(str)	whether one contains other's characters at end
s.contains(str)	whether the given string is found within this one

Note: all these methods return booleans (true or false)

The null Keyword

null is a special value for Strings, and other objects, that means the variable is equal to nothing. If you try to run a method on a String that is equal to null you will get a **NullPointerException**, a type of run time error. We will talk about how to deal with null Strings and exceptions in later tutorials.

```
// setting a String to null
String nullString = null;
```

String Method Examples

The End