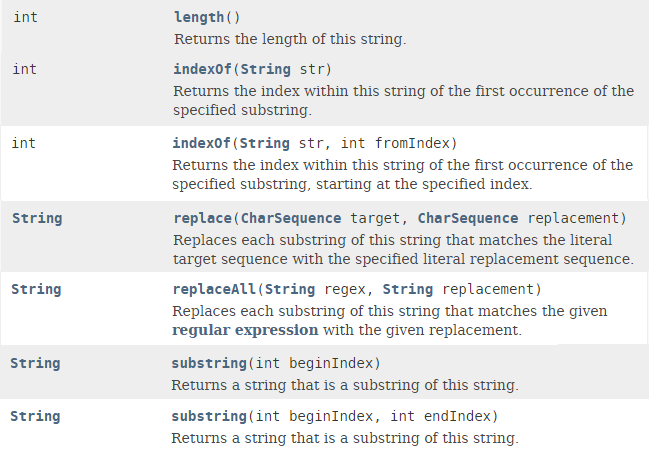
2.06 notes

## The Methods to Java's String Class Madness

Java's String class consists of over one hundred methods, a small segment of code, for handling a wide range of tasks related to alphanumeric data. Although you will only use a few String methods on a regular basis, be sure to scan the complete list in case you find a situation where one of the less common methods will be useful. The methods introduced in this lesson are powerful and may be used individually or in combination to solve many string handling tasks.

### Part 1

The String class provides many useful methods for processing strings of characters. For example, you can determine the length of a **String** object, locate the position of a specific character in a String literal, locate a specific string of characters within a larger string, replace one character with another, etc. The methods covered in this lesson are listed below. What can you learn about each one from their entries in the Method Summary table?



The Method Summary table tells you four important pieces of information about each method.

1. The name of the method
2. A description of the method’s purpose
3. How many parameters the method takes and the type of each one
4. The type of information the method returns after it completes its task

### Part 2

In order to use a method, it has to be called (or invoked) using [**dot notation**](javascript:void(0);) (the notation used to call methods or access variables). For example, the following code segment illustrates how to determine the length of a string of characters assigned to a String object.

String phrase = "Four score and seven years ago";  
int stringLength = phrase.length();  
System.out.println("Length of String: " + stringLength);  
System.out.println();

Dot notation is used in the second line of this code segment to separate the **String** object called phrase from the **length()** method of the **String** class. Since **phrase** is declared to be a **String** object, it can utilize the **length()** method of the **String** class. In this example, dot notation tells the phrase object to call or invoke the **length()** method to determine how many characters it contains.

Handling String literals is an important programming skill. The methods covered in this lesson are very useful. Pay close attention to parameters for the methods. Some numeric ones are inclusive and others exclusive. Others may appear to be "1 off" until you realize that programmers usually start counting index positions from 0, not 1. Also, some parameters are inclusive and others exclusive.

**String** methods can be tricky at first, so let’s put them to practical use in a simple demo program.

* Create a new project called 02.06 String Methods in the Mod02 Lesson folder.
* Open the [**02.06 Desk Check**](https://lti.flvsgl.com/flvs-cat-content/m84egbrd90osvgcqlpq4jm8ij1/flvs-cat-session/apcomputersciencea_v20/module02/lesson06/pop/03_06b/03_06b_pop01.htm) file.
* Use the Desk Check document to carefully trace the code before running the demo program.
* Download the [**StringPractice.java**](https://lti.flvsgl.com/flvs-cat-content/m84egbrd90osvgcqlpq4jm8ij1/flvs-cat-session/apcomputersciencea_v20/module02/lesson06/docs/03_06b/stringpractice.java) file to the newly-created folder and run it.

You will use String methods frequently, so study the demo program very carefully; there may be some surprises and challenges you don’t expect.

Now it’s your turn to experiment with String methods. Assign each string of characters in the Input column below to a String object. Use methods of the String class to produce the corresponding string of characters in the output column. Review the examples in the StringPractice class for hints.

**Input, Output and Solution**

| **Input** | **Output** | **Solution** |
| --- | --- | --- |
| 407–555–3326 | (407) 555–3326 | Solution  String phone = "407-555-3326"; int indexOfDash = phone.indexOf("-"); String newPhone = "(" + phone.substring(0, indexOfDash) + ")" + phone.substring(indexOfDash +1); System.out.println(newPhone); |
| 4075553326 | 407.555.3326 | Solution  phone = "4075553326"; newPhone = phone.substring(0,3) + "." + phone.substring(3,6) + "." + phone.substring(6); System.out.println(newPhone); |
| 03/06/2017 | 03-06-2017 | Solution  String date = "03/06/2017"; date = date.replace("/", "-"); System.out.println(date); |
| Tallahassee, Florida | TAllAhAssee, FloridA | Solution  String location = "Tallahassee, Florida"; location = location.replace("a", "A"); System.out.println(location); |
| Anna Wright | Wright, Anna | Solution  String name = "Anna Wright"; int indexOfSpace = name.indexOf(" "); String firstName = name.substring(0, indexOfSpace); String lastName = name.substring(indexOfSpace + 1); String newName = lastName + ", " + firstName; System.out.println(newName); |