Christian M Rapp
Promineo Tech BE Week 3 Research: Arrays and Methods

Writing Prompt #1:

Select **five** methods from the **String JavaDocs** and describe the following for each:
(a) What is the method signature? (b) What does the method do? (c) Why would this method be useful (how could you use it)?

Java provides many methods for the String object. Using methods will allow the programmer to perform various actions on an object that will help with whatever particular problem the programmer is trying to solve. I will highlight 5 methods and describe them.

For this example, we will use the declaration **`String sampleText = "Sample Text";`**

- **charAt() –** returns the char value at a specified index. This method would be helpful if the programmer was trying to locate what letter is the 5th letter of a String. A method signature using the above example:

  `System.out.println(name.charAt(6));` would print **a space** to the console as position 6 is the space between SAMPLE and TEXT. It's at 6 because of Java's zero-based indexing.

- **length() –** returns the length of a string. This method is useful if the programmer needs to determine the length of a string (handy if comparing strings). The method signature using the above example:

  `System.out.println(name.length());` would print **11** to the console.

- **toUpperCase()** – converts a string to all uppercase letters. This method would be useful as a first step in comparing strings. The programmer could convert the string to uppercase before comparing. The method signature using the above example:

  `System.out.println(sampleText.toUpperCase());` would print **`SAMPLE TEXT`** to the console

  NOTE: *there is also a toLowerCase method which converts the string to lowercase*

- **concat()** – allows the programmer to concatenate a string. This could be used if the programmer were making a program to conjugate words (adding –ed, for example). The method signature using the above example:

  `System.out.println(sampleText.concat("!!"));` would print **`Sample Text!!`** to the console

- **replace()** – allows the programmer to replace a character (char) in a string. This could be used to not only replace a letter in a string, but delete a letter totally. The method signature using the above example:

  `System.out.println(name.replace('e', 'E'));` would print **`SamplE TExt`** to the console

Christian M Rapp
Promineo Tech BE Week 3 Research: Arrays and Methods

Writing Prompt #2: Select **five** methods from the **Array JavaDocs** and describe the following for each:
(a) What is the method signature? (b) What does the method do? (c) Why would this method be useful (how could you use it)?

Java also provides many methods for the Array object. Using methods will allow the programmer to perform various actions on an object that will help with whatever particular problem the programmer is trying to solve. I will highlight 5 methods and describe them.

For this example, we will use the declaration `int[] numbers = {1,27,99,32,2};`

- **length –** returns the length of an array. This method is useful if the programmer needs to determine the number of elements in an array. Especially helpful if the array is dynamic. The method signature using the above example:
  `System.out.println(numbers.length);` will print 5 to the console.

- **Arrays.toString –** prints a string representation of your entire array. Useful for displaying the contents of your entire array. The output may not be desired for the final program, but would be useful for debugging.

  *Note: this method requires **java.util.Arrays** be imported*
  The method signature using the above example: `System.out.println(Arrays.toString(numbers));`

- **clone –** makes a copy of an array, instead of another instance that shares the same reference. Instead of iterating over an entire array to copy, duplicating the array can be done in one line of code. The method signature using the above example:

```
int[] numbers2 = numbers.clone();
```

- **Arrays.equals –** a quick and easy way to compare arrays, especially ones made by using the **clone** method. Method is typically used in a comparison statement.

  *Note: this method requires **java.util.Arrays** be imported*
  The method signature using the above examples:

```
if (Arrays.equals(numbers, numbers2)) {
      System.out.println("The arrays are equal");
      } else {
            System.out.println("The arrays are not equal.");
}
```

- **Arrays.fill –** fills an array with a specified value. While you wouldn't initialize an array with this method, it would be useful if the programmer needed to reset an existing array to a default value.

  *Note: this method requires **java.util.Arrays** be imported*
  Example:

```
Arrays.fill(numbers,8);
System.out.println(Arrays.toString(numbers));
```

CONSOLE OUTPUT: `[8, 8, 8, 8, 8]`

Sources:

https://docs.oracle.com/javase/7/docs/api/java/lang/String.html
https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html
https://www.w3schools.com/java/ref_string_touppercase.asp
https://www.javatpoint.com/java-string-concat
https://www.geeksforgeeks.org/arrays-tostring-in-java-with-examples/
https://www.geeksforgeeks.org/array-copy-in-java/
https://www.studytonight.com/java-util/java-arrays-fill-method