**Chapter 9 - The String Class**

A String is not a primitive data type, like int, double, char and boolean. It is its own class, like those we created in Chapter 8. The String Class has 13 constructors and more than 40 methods for manipulating Strings. Assume in the following definition and coding examples that s1, s2, s3 are all String objects.

**Checking Equality of Strings**

if (string1.equals(string2)) - Good idea. This checks if the text contained in these two different Strings is the same.

if (string1 == string2) - Bad idea. This only checks to see if these two strings reference the same object.

\*Be reminded that Strings are case-sensitive, so if s1 = "cat" and s2="CAT", they will **not** be seen as equal.

**AP String Methods**

s1 = "hotdog" and s2 = "hamburger"

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Purpose** | **Syntax** |
| compareTo | Returns an integer value when comparing two strings based on the location of their characters in the ASCII table. | s1.compareTo(s2) (Returns -14) |
| length() | Returns the length of the String as an integer. | s1.length() (Returns 6) |
| substring(int) | Returns a new String that starts at the index of the provided integer through the end of the String. | s3 = s2.substring(3)  s3 = "burger" |
| substring(int, int) | Returns a new String start at first integer index, ending at the second integer index. | s3 = s2.substring(3, 6)  s3 = "bur" |
| indexOf(String) | Return the index (int) of the first occurrence of the provided String within this String. If not found, -1 is returned. | int x = s1.indexOf("dog")  x = 3; |
| concat(String) | Usually to essentially "glue" two String together. Alternately, you could just use the + operator to do the same thing. | s3=s1.concat(s2) or s3 = s1 + s3  s3 = "hotdoghamburger" |

**Other String Methods**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Purpose** | **Syntax** |
| toLowerCase() | Returns a new String will all character in lowercase. | s3=s1.toLowerCase() |
| toUpperCase() | Returns a new String will all character in uppercase. | s3=s1.toUpperCase() |
| trim() | New String with all leading and ending whitespace removed. | s3 = s2.trim() |
| replace(char1, char2) | Replaces all char1 occurrences with char2 | s2.replace("h","y") |
| replaceAll(String1, String2) | Replaces all String1 occurrences with String2 | s2.replaceAll("ham","horse") |

**Character and Substring Methods**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Purpose** | **Syntax** |
| indexOf(char) | Returns the index (int) of the first occurrence of the char in the String or -1 if not found. | int x = s1.indexOf("do");  x=3. |
| indexOf(char, int) | Returns the index (int) of the first occurrence of the char in the String starting after the (int) index. (or -1 if not found) | int y =s1.indexOf('o', 2);  y = 4; |
| lastIndexOf(char) | Returns the index (int) of the last occurrence of the char in the String or -1 if not found. | int x = s2.lastIndexof("xo");  x= -1; (not found) |
| lastIndexOf(char, int) | Returns the index (int) of the last occurrence of the char in the String starting after the (int) index. (or -1 if not found) | int y =s2.lastIndexOf('r', 5);  y = 8; |

**valueOf() Method**

*valueOf()* is an overloaded method. (many methods of same name with different parameter lists) We use it to convert a primitive data type, such as int, double, char or boolean to a String.

int x = 17;

String s1 = String.valueOf(x); s1 = "17";

**LiveLab Labs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Program Name** | **Description** | **User Input** | **Output** | **Points** |
| GoBigTime | User enters a String. It is outputted with all characters converted to upper case. | Hi there! | HI THERE! | 3 |
| NumChars | User enters a String. The program simply outputs the number of characters in that String. | To be, or not to be. | 19 | 3 |
| Polish | User enters a single word String. The program outputs that String with "ski" added at end. | poker | pokerski | 3 |
| GamerTag | User enters a single word String. The program outputs that String with "xX" added at beginning and "Xx" at the end of the String. | FPS\_God | xXFPS\_GodXx | 3 |
| ContainsU | User enters a String. The program outputs "true" if the name contains either a lower or upper case U. ('u' or 'U'). If it doesn't contain a U, have it output "false". | I'm in debt  Under the rainbow | false  true | 4 |
| Bookends | User enters a String. The program will output "true" if the first and last letter in the String is the same character, otherwise, it simply outputs "false". | a wonderful idea  home improvement | true  false | 4 |
| DoubleLetter | User enters a String. The program will output "true" if there are consecutive, identical characters, otherwise, it will output "false". | butter  supercalifragilistic | true  false | 5 |
| DeleteVowels | User enters a String. The program will remove all lower case vowels (aeiou) from the String and output it. | Optimus Prime | Optms Prm | 5 |
| ShiftThree | The user enters a String. The program will shift all letters three to the right in the alphabet. Make sure x, y, z "wrap" back around to the beginning of the alphabet. Example: Bill Dixon becomes Eloo Glarq. Spaces and non-letters remain unchanged. | Bill Dixon | Eloo Glarq | 10 |
| Palindrome1 | The user enters a single word String. The program determines if the phrase is a palindrome. (contains same letters in the same order going forward or backward). Output "true" or "false" | radar  prop | true  false | 10 |
| Palindrome2 | The user enters a String. The program determines if the phrase is a palindrome, however, this time, it doesn't factor in the spaces or the case of the letters. Output "true" or "false". | Madam, I'm adam.  Hello, Oprah! | true  false | 20 |
| Total |  |  |  |  |