

String Manipulation Project

By Jeremy Griffith

The purpose of this project is to give you some familiarity with Java loops and Strings.

To complete the project, you should only need to be familiar with the StringBuilder class. All other utilities are unnecessary. For the purposes of efficiency, avoid using APIs to clean up your code. In most cases, a direct solution will perform better.

# Problem Statement

For this project, you’ll be asked to implement your own StringManipulation class. The purpose of this class is to provide eight utility string methods. These methods are described below:

## isAlphabetical

The isAlphabetical method must accept an input string and determine if the string is in alphabetical order. Character case can be ignored. If the string contains characters other than letters, these characters may be ignored. The following contains a table of true and false cases:

|  |  |
| --- | --- |
| True Cases | False Cases |
| “abcd” | “dcba” |
| “ab-fg” | “abdc” |
| “ghi--- “ | “ ba” |
| “HLmnO” | “Zy17” |

## reverseString

The reverseString method must accept an input string and return a string in the reverse order of the input string. There are no rules beyond this behavior. Simply reverse the input string and return it. The following is a table of input strings and their expected outputs:

|  |  |
| --- | --- |
| Input String | Output String |
| “hello” | “olleh” |
| “Lebron James” | “semaJ norbeL” |
| “FiShY” | “YhSiF” |
| “A” | “A” |
| “” | “” |

## capitalizeVowels

The capitalizeVowels method must accept an input string and return a string with all of the vowels capitalized. The following table contains a set of input strings with their respective outputs:

|  |  |
| --- | --- |
| Input String | Output String |
| “hello” | “hEllO” |
| “Lebron James” | “LEbrOn JAmEs” |
| “FiShY” | “FIShY” |
| “A” | “A” |
| “” | “” |

## insertSpacesBetweenLetters

The insertSpacesBetweenLetters method must accept an input string and return a string with spaces placed between letters. In addition, this method must trim all leading and trailing spaces, and enforce a strict one-space-per-letter-pair rule. The following table contains a set of input strings with their respective outputs:

|  |  |
| --- | --- |
| Input String | Output String |
| “hello” | “h e l l o” |
| “Lebron James” | “L e b r o n J a m e s” |
| “FiShY” | “F i S h Y” |
| “A” | “A” |
| “” | “” |

## convertToHex

## removeChar

## generateAllChars

## containsSubSequence

# Tips & Tricks

* Read each problem carefully. Each description contains constraints which save you some coding.
* Remember, this problem set is rated for beginners. Each method should only require about 20-30 lines of code with comments.
* Comment your code. Don’t overdo it, but make sure to give every method a nice Javadoc comment.
* Test your code. You can use the JUnit tests included with the solution to test your own code, or you can write your own test cases.
* Use [The Renegade Coder](https://therenegadecoder.com/) as reference. Since a lot of this material is based on content from the site, you can expect the site content to be relevant to your solution.
* Don’t be afraid to reach out. I’m happy to work with anyone on their coding journey. Feel free to email me at [jeremy.griffith@therenegadecoder.com](mailto:jeremy.griffith@therenegadecoder.com).