The Renegade Coder
CODE FIRST. ASK QUESTIONS LATER

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:

1. 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"

2. 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"
""	""

3. 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”
“”	“”

4. 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”
“”	“”

5. convertToHex

The convertToHex method must accept an input string and return a string where each of the characters has been converted to a two-digit hex code. The following table contains a set of input strings with their respective outputs:

Input String	Output String
“hello”	“68656C6C6F”
“Lebron James”	“4C6562726F6E204A616D6573”
“FiShY”	“4669536859”
“A”	“41”
“”	“”

6. removeChar

The removeChar method must accept an input string and a character and return a string with all occurrences of that character removed. The following table shares a few examples:

Input String	Input Character	Output String
“hello”	‘h’	“ello”
“Lebron James”	‘e’	“Lbron Jams”
“FiShY”	‘j’	“FiShY”
“A”	‘A’	“”
“”	‘h’	“”

7. generateAllChars

The generateAllChars method must accept two characters and produce a string which contains all characters in between. In other words, the resulting string will be the inclusive set between the two terms. The sort order should be ASCII codes, so it's perfectly legal to use symbols. In the case that the first character is greater than the second character, produce the same list running least to greatest. The following table contains a few short examples:

Start Character	End Character	Output String
'a'	'h'	"abcdefgh"
'h'	'a'	"abcdefgh"
','	'?'	",-./0123456789:;<=>?"
'A'	'A'	"A"

8. containsSubSequence

The containsSubSequence method must accept a string and a substring and return the status of the search as a boolean. The method should only return true if the input string contains the characters in the substring in the order the appear. The characters do not have to appear consecutively. The following table gives a few examples:

Input String	Sub Sequence	Result
"hello"	"hlo"	true
"Lebron James"	"brony"	false
"world"	"worldy"	false
"A"	"A"	true

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 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.