CSCI 1301: Introduction to Computing and Programming

Lab 07 – Fun with Strings

Prerequisites

The lab deals mainly with material from Chapter 4 (loops and nested loops) but builds on material learned in Chapters 2-3 – especially the material on Strings.

Lab Details

The code you write for this lab should be put in a file named **`StringFun.java`** and in a public class named **`StringFun`**. In this lab, you will create an environment where the user can enter a sentence (as a string) and then manipulate that string using 5 basic commands. These manipulations will affect the successive commands that they make. Look at the output examples for a clear demonstration of this. You will implement these commands using loops and basic string methods. You are NOT permitted to use the StringBuilder class, Arrays class, or any class in java.util.stream (the use of any of these will result in a grade of zero on this assignment). Furthermore, do not use any break or continue statements in the body of your loops.

The ONLY String methods that you can use for this lab are:

length, concat, +, charAt, substring, and equals (or equalsIgnoreCase).

When the program begins, it asks the user to input a String. This input can range from a single letter to a complete sentence.

The program will then list the 6 possible commands that the user can choose from (see below). The user can use these commands to modify the original sentence. The program will continue to ask for commands until the user enters the "quit" command. The 6 commands are:

1) **Reverse**: If the user types "reverse" with any capitalization, your program will reverse the original string and print the result.
2) **Replace First**: If the user types "replace first" with any capitalization, your program will prompt the user to enter the character to replace and the new character (to replace the old character). Afterwards, it will replace the first instance of the old character with the new character (see example output below) and prints the modified string.
3) **Replace Last**: If the user types "replace last" with any capitalization, your program will prompt the user to enter the character to replace and the new character (to replace the old character). Afterwards, it will replace the last instance of the old character with the new character (see example output below) and prints the modified string.
4) **Remove**: If the user types "remove" with any capitalization, your program will prompt the user to enter the character to remove, and which instance it should remove (it can remove any instance of the character). The program will then carry out the remove and print the modified string.
5) **Remove All:** If the user types "remove all", with any capitalization, your program will prompt the user to enter the character to remove. Then, it removes all instances of the given letter in the sentence. This command also prints the modified sentence to the console.
6) **Quit**: stops the program.

Error Handling

The program should print a friendly error message as shown in the examples and continue to the next command if any of the following arises:

- The user tries to replace or remove a character not in the input string
- The user tries to remove the 3rd 'a' from the sentence but there is no 3rd 'a'
- There may be additional error situations that you can catch and handle gracefully. You should make every attempt to handle these situations when possible.

In all commands above, when the user enters a character, you should assume that **case matters**. For example, if the user says remove all 'A' characters, your program should not remove 'a' characters. Also, the user may enter numbers and special characters and they can be treated as any other character. In other words, no need to modify your code specifically to handle these values.

Hints

1) You have multiple days to complete this lab. Start by writing one command, convince yourself that it is working perfectly, then move onto the next command. Don't try to do all six at one time.
2) While this lab will require nested loops, **you should NOT start by trying to write nested loops**. Practice and review the textbook material about nested loops before attempting to nest your loops. Only after you are fully convinced that all six of the commands are all working should you proceed to wrap your program in a loop to make them execute multiple times.
3) Store your user's string in a variable and modify that variable with every command. If you are simply printing the correct value after the user enters a command but not modifying a variable, you will not have the updated value on the next loop iteration.

Sample Input and Output

Notice that in these examples manipulations of the input string are carried forward to successive commands.

Example 1: Testing Reverse
Enter the string to be manipulated
Go Dawgs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
reverse
The new sentence is: sgwaD oG
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
reverse
The new sentence is: Go Dawgs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
Quit
…Execution Ends…

Example 2: Testing replace first and last and capitalization of commands
Enter the string to be manipulated
Co Dawcs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
RePlAcE FiRST
Enter the character to replace
C
Enter the new character

G
The new sentence is: Go Dawcs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
replace LAST
Enter the character to replace
c
Enter the new character
g
The new sentence is: Go Dawgs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
Quit
…Execution Ends…

Example 3: Letter not found with replace first

Enter the string to be manipulated
co dawcs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
replace FIRST
Enter the character to replace
C
Enter the new character
G
The letter was not found in the word
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
replace first
Enter the character to replace
c
Enter the new character
G
The new sentence is: Go dawcs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
Quit
…Execution Ends…

Example 4: Testing remove and remove all
Enter the string to be manipulated
cazzzzooooo dawggggoooos
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
remove ALL
Enter the character to remove
o
The new sentence is: cazzzz dawggggs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
remove
Enter the character to remove
z
Enter the z you would like to remove (Not the index - 1 = 1st, 2 = 2nd, etc.):
4
The new sentence is: cazzz dawggggs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
remove
Enter the character to remove
g
Enter the g you would like to remove (Not the index - 1 = 1st, 2 = 2nd, etc.):
4
The new sentence is: cazzz dawgggs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)

remove
Enter the character to remove
g
Enter the g you would like to remove (Not the index - 1 = 1st, 2 = 2nd, etc.):
5
Error: the letter you are trying to remove does not exist
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
remove
Enter the character to remove
a
Enter the a you would like to remove (Not the index - 1 = 1st, 2 = 2nd, etc.):
1
The new sentence is: czzz dawgggs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
remove all
Enter the character to remove
z
The new sentence is: c dawgggs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
remove all
Enter the character to remove
c
The new sentence is: dawgggs
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
Quit
…Execution Ends…

Example 5: Command invalid
Enter the string to be manipulated
computer science
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
none
Command invalid. Try again
Enter your command (reverse, replace first, replace last, remove all, remove, quit)
…Execution Continues…

Additional Requirements

These are things that make the graders lives easier, and ultimately, you will see in the real world as well. Remember the teaching staff does not want to touch your code after they gave you requirements; they want to see the perfect results they asked for! Here is a checklist of things you can **lose points** for:

- (-1 point) If the source file(s)/class(es) are named incorrectly (case matters!)
- (-1 point) If your source file(s) have a package declaration at the top
- (-1 point) If any source file you submit is missing your Statement of Academic Honesty at the top of the source file. All submitted source code files must contain your Statement of Academic Honesty at the top of each file.
- (-1 point) If you have more than one instance of Scanner in your program. Your program should only have one instance of Scanner.
- (-1 point) Inconsistent I/O (input/output) that does not match our instructions or examples exactly (unless otherwise stated in an assignment's instructions). Your program's I/O (order, wording, formatting, etc.) must match our examples and instructions.
- (-2 points) If your submission is late

- (-7 points) If the source file(s) are not submitted before the specified deadline's late period ends (48 hours after the deadline) or if they do not compile.
- If your (-1 point) comments or (-1 point) variables are "lacking"
    - Here, "lacking" means that you or a TA can find **any** lines of code or variables that take more than 10 seconds to understand, and there is no comment, or the variable name does not make sense (variable names like **b**, **bb**, **bbb**, etc. **will almost never be acceptable)**
- (-1 point) Indentation is not consistent throughout your source code
    - Refresh your memory of indentation patterns in chapter 2 in the course textbook
    - Be careful of a combination of tabs and spaces in your files (use one or the other)
- (-7 points) If you use a non-standard Java library or class (StringBuilder class, Arrays class, any class in java.util.stream) or other code specifically disallowed by the lab/project. For this assignment, you are **NOT** permitted to use a **break** or **continue** statement inside of the body of a loop, and using one of these statements in the body of a loop will result in a grade of zero on this assignment.

If any of the above do not make sense to you, talk to a TA or your lab instructor.

eLC Submission and Grading

After you have completed and thoroughly tested **`StringFun.java`** submit it to ***eLC*** in order to receive credit for the lab. Always double check that your submission was successful on ***eLC***!

The lab will be graded according to the following guidelines.

- Students may earn 3 points for lab attendance and 7 points if their program passes various test cases that we use for grading. **Some test cases may use values taken from the examples in this document and some test cases may not**. You should come up with additional test cases to check that your program is bug free.
- All source code must adhere to good Java programming style standards as discussed in lecture class and in the course textbook readings.
- Points will be deducted for not following any of the aforementioned instructions or requirements.