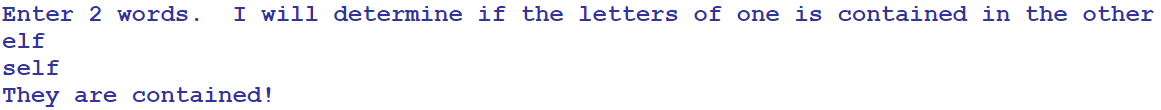
Lab 06 Report: To Do List

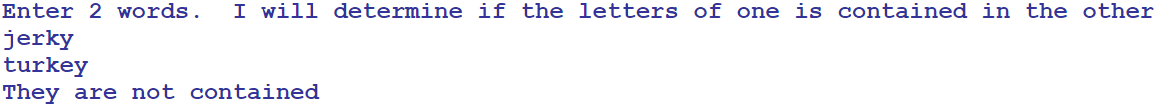
Problem

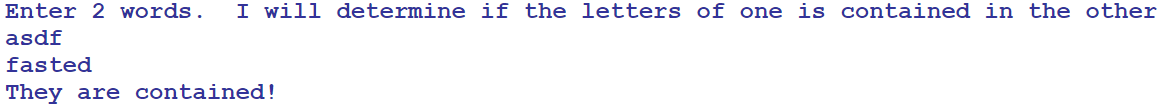
Write a program that takes in two words, and then it recursively determines if the letters of the first word are contained, in any order, in the second word.  If the size of the first word is larger than the second then it should automatically return false.  Also if both strings are empty then return true. Proposed Solution

1. Create a class that includes a main method
   1. In the main method prompt the user to “Enter 2 words. I will determine if the letters of one is contained in the other”
   2. Import and create a scanner which will except user input of type string for the next two lines
   3. Call the Boolean static method with the two strings the user entered and 0 for their indexs as it starts from 0 not at the end (which would use string.length(), and way easier to implement)
2. Create a static method inside of the class which returns a Boolean and takes in two strings, and the index of both strings (String a, string b, int aLoc, int bLoc)
   1. Set char to equal the string.charAt(loc);
   2. If the chars equal eachother, move on to next char of first word if there is a next one through a recursive call, else return true
   3. If chars don’t equal eachother, move on to next char of second word if theres a next one through a recursive call, else return false

Tests and Results







Problems Encountered

The toughest part of this lab was returning the correct true or false Boolean valued based on what occurred in the program. This was due to the program having to have recursive calls and if statements to handle what was going on, so you couldn’t return or end the program in some cases without adding more if statements with return values at different points within the program.

Conclusions and Discussion

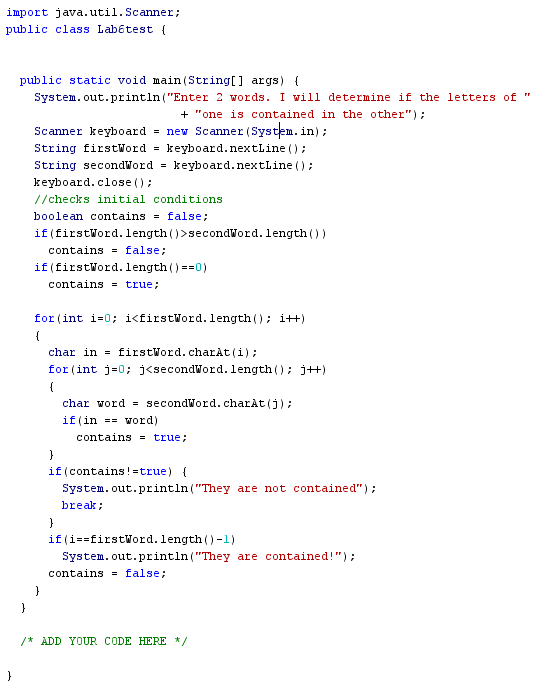
This lab showed the importance and efficiency of recursive calls over using for loops to either sort or do something to an input. The way this works is by calling a method in the main method and recalling the method within the method to keep doing the functionality.

While this solution works, another solution which I first did the lab was using Arrays.sort method built into java to take in both of the strings, sort them alphabetically and go through down the list comparing them. Another way is to start at the end of the words and sort down starting at .length -1.

Additional Questions

1. Write some pseudo-code to solve the same problem but with loops instead.

Technically not pseudo-code, but I had written out the problem in DrJava before attempting to solve the solution like the lab suggests.



1. Between the recursive version and the iterative version you solved in the previous question, which one would run faster in real time?  Why?

The for loop one solved in the previous question would run faster in real time as it does not have to call itself multiple times, but instead would just keep iterating throughout the nested for loop.