Week 13 Lab – Ferguson

SentenceDemo.java-

This file is meant to take a string object and traverse through the string to see if the string object contains a certain group of characters specified by the user. While you could just use the .contains() method, for this lab we are supposed to use recursion. To do this, I take the substring of the object which is the length of the word the user wishes to find, and then if the object matches the word, it returns true, and if it does not, it traverses to the next letter and checks again. If the program reaches the end of the word before it returns true, it returns false. To do this, I had to first put an “if” statement incase the substring of the object that the program was trying to reach was too long, it wouldn’t error out. Next, the program checks to see if the substring of the object matches with the text. If it does not, it grabs a different substring of the object and repeats all over again, which is how you use recursion. Since this program has no user input, the output does not vary, so only 1 attempt is necessary. Since we were required to add extra test cases, one true and one false, I decided to see if “Mississippi!” contains “sss” (expected: false), and if “Mississippi!” contains “pi!” (Expected: True).

A screen shot of a computer

Description automatically generated

SentenceDemo2.java-

This file makes recursion a reasonable method to reach the output. For this file, we are supposed to return the index at which the sample text matches the string object, and -1 if the string object does not contain the sample text. To do this, I once again had to add an if statement to make sure I do not get an outOfBounds error. The second step was the same; The program checks to see if the next substring of the object matches with the text. To check, we had to include the index of which to start the substring, which made the recursion step much easier. I was able to call the same method but with index + 1 to grab the next substring in the string object. Since this program has no user input, the output does not vary, so only 1 attempt is necessary. Since we were required to add extra test cases, one positive value and one -1, I decided to see if “Mississippi!” contains “Mss” (expected: -1), and if “Mississippi!” contains “iss” (Expected: 1).

A screen shot of a computer

Description automatically generated