Hints and additional information for Programming Assignment 2: Efficient Flesch Score calculation with EfficientDocument.java

Part 1: EfficientDocument.java

Help with getTokens:

Notice that we have given you the pattern string to use to split the text already written as the first line in processText. Make sure you understand what this regular expression will split the text into. You might find it useful to print out the list of tokens for some examples to help you get started. In main, you can create new EfficientDocument objects containing different text strings (vary the words and the punctuation). Then call getTokens on these efficientDocument objects and print out the resulting list of tokens.

Help with isWord:

Notice also that we've provided a method that checks to see if a given string is a word.

boolean isWord(String s)

The idea behind this method is that it takes a string that either contains sentence ending punctuation or only contains alphabetic characters (i.e. it takes a token that is in the list returned by getTokens). It will return true if the input string only contains alphabetical characters, and false if it contains sentence-ending punctuation.

Why is this method useful? Well, once you get back the list of tokens, you need to count the total syllables, the total words, and the total sentences. One way to do this is to iterate through the tokens you get back from the call to getTokens. For every token that is a word, you count and accumulate its syllables (hint: remember countSyllables from Document.java, if you didn't implement it, you can find it in our provided solutions), and then increment a counter keeping track of the total number of words. But if the token you see is not a word, that means it is a sentence-ending punctuation, so instead of incrementing the number of syllables and words, you increment the number of sentences!

However, make sure you pay special attention to the case where the last sentence does not end in a sentence-ending punctuation. It should still be counted as a sentence, but you'll need to handle that case carefully...

Part 2: Benchmarking

- The main method in the starter code provides pseudocode for this part in the comments.
- You can print the time in any units you want, but recall that the unit of System.nanoTime() is nanoseconds, so to convert these to seconds you can divide by 1000000000 (10^9). Be sure to use type long to store the results of System.nanoTime()! ints don't give you room to store big enough numbers.
- Your data will be too noisy if you create only one BasicDocument and one EfficientDocument for each size, so we recommend you use a loop to create and call fleschScore on many

BasicDocument and EfficientDocument objects and measure the total time for the whole loop. This is what we have provided the 'trials' variable for: it is the number of trials that you do for each type of document at each size. Don't worry about dividing the total time by the number of trials. You can just output the total time. Play around with different values of trials until you get data that is smooth.