Project
Check Point 1
CSCI 204
Due Sept 16

In this checkpoint we will focus on the following topics: object oriented layout, I/O streams, Error handling, and Algorithm Analysis.

When you first start a problem it is a good idea to figure out all the objects and how they are all interrelated to each other. You will see much more of this in CSCI205. Here, I will provide you will an overview. We can first break down the design into three areas. These areas are namely: User Interface, Algorithms, and Utilities. The User Interface is how and what the user will normally interact with and will be mostly command line and/or GUI based in this project. This will be one of the last areas we look at in this project. The Algorithms will be our methods for classification and identification. These will depend deeply on the Abstract Data Types we will learn in this course. The last is Utilities. These tend to be the “meat-and-potatoes” of most codes but are not the most fun to program. This checkpoint will focus mostly only the Util at this time, since all other areas need the Util.

A **Document** is going to be our primary data type. A Document is an object that contains a **DocumentStream** and an array of **Sentences.** Both **DocumentStream** and **Sentences** are objects and both or either can be “none” is the Document. **Sentences** here will be simply a wrapper to a python str. The **DocumentStream** will be our wrapper to our “buffered” I/O on reading in and printing out a document.

A **DocumentStreamError** derives from an I/O exception. It will be our method to catch exceptions that may come from DocumentStream.

Two visualization methods will be used in our program. The first is old fashion commandline. The second will utilize the matplotlib in python. Right now we will only focus on the commandline methods which we will use to make 2D scatter and bar charts. These classes will be called **CommandLinePloter** and **MatPlotPloter.**

In order to get full credit for this checkpoint you must complete the following.

1. Rough out the classes for: ~~Document, DocumentStream, Sentence~~, DocumentStreamError, CommandLinePloter, MatPlotPloter, and UserInputError which is an exception. Make sure to place them in the correct files.
2. ~~Add the following attributes to Document. Add private list of Sentence call lines with getters and setters. Add a reference number id, number wordcount, number linecount, number charcount. Add all needed getters and setters~~.

3. ~~Add the following attributes to Sentence: wordcount charcount, and punctuation which stores a string of how the sentence ends.~~
4. ~~Design DocumentStream for a whole read option. This will include a method called readWhole that will not take a filename (filename should be an optional attribute of DocumentStream) and return a list of sentences. The sentences should be divided based on ending line punctuation, such as ".", "!", "?", ";", and more than two empty spaces.~~
5. ~~Add a method to Document call generateWhole(), that will use the method readWhole in DocumentStream along with~~ parse the title information from the text if one exists given a filename.
6. ~~Add methods getWordCount and getLineCount in Document that finds these values if not already found and returns them. If already generated, return that value~~ (Unless we generated a new document!!!!)
7. ~~Write a method in DocumentStream that called writeWhole, that will write the whole document out using the sentences. (One Sentence per output line)~~
8. ~~Write a method in Sentence called parseWords that will return an array of Words.~~
9. ~~Write a method in Sentence called parseChar that will return an array of characters in the Sentence.~~
10. ~~In the commandline plotting class add a method that will generate a 2D coordinate plan given a start and end point for both the x and y-axis.~~
11. ~~In the commandline plotting class add a method called 2dScatter that takes two list with the second list having a default value of none. If both list are given, it will use the first list as x-coordinate and the second list as y-coordinate. If only 1 list is given, it will treat the given list as the y-coordinates and the none list will be the x-coordinate of value 1 to len(givenlist)~~ .
12. Create class called BasicStats. In this class define a static method call createFreqMap that will take a list and return a dictionary with keys being unique entries in the list and the values the frequency of entry types in the list. In the comments of this function perform algorithm analysis for your code.
13. In the BasicStats class, create a method called topN that takes a dictionary and a value n, and it returns a dictionary with keys being the top keys in the given dictionary and the values those the values of the top keys. Performance algorithm analysis for your code in the comments.
14. Do step 13 for the bottom. In comments, answer if there is a faster way to do both at the same time???
15. Put in the correct error handling using exception for the I/O.
16. Create a main function and file. The main function should ask the user for a filename, create a document from this name, read it in, find the top 10 words using the methods made above and plot them in the 2Dscatter.