**Online Submission Assignment Cover Page**

Name: Richard Break
ID#:
Date: 10/04/2012
Course #: CIS*3190
Course Name: Software for Legacy Systems
Instructor: Dr. Michael Wirth
Assignment #: 3
Assignment Name: Legacy Software (FLESCH READABILITY INDEX)
# Of Pages (including this one): 3

| | |
|---|---|
| | (For instructor's use)<br><br>Grade: |

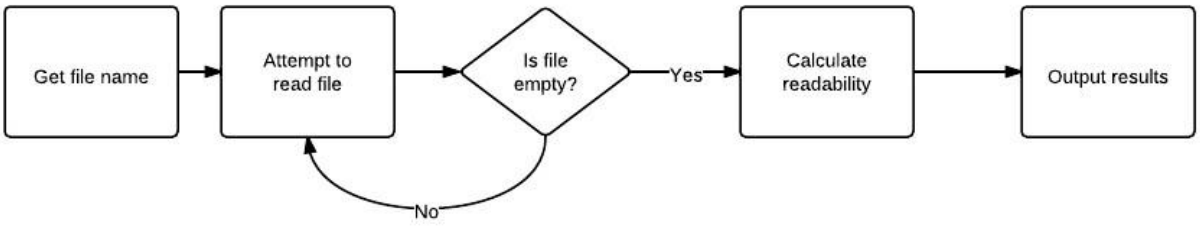
# Design Document: Flesch Readability index using Ada

When designing this program, there were many decisions that that lead to Ada versus other legacy languages. Since our class focused on Fortran, Ada and Cobol, the selection on which language was going to be implemented was limited to keep in the spirit of the course. The decision of what language that was going to be implemented was based on how the language read and stored a file's contents and how a language is able to be partitioned into separate sections.

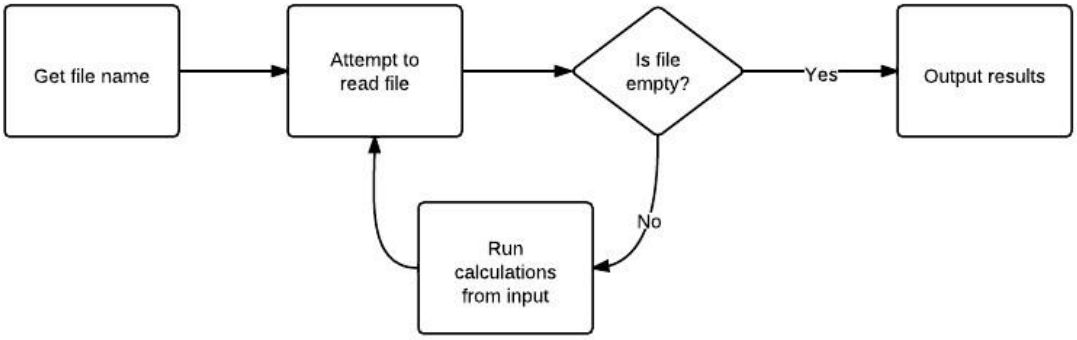
The assignment required that the program would be able to read from a file. Since Ada has the ability to implement unbounded strings, Ada became a prime candidate for the project. The most effective way to use Cobol in this manner was to read the file character by character, and process each character before continuing since storing the entire file at once without knowing the file's format is impossible. Similarly, Fortran file I/O would have required reading the file character by character, since storing the file contents into a single string seemed complex.

The next issue to address was to divide the problem into coherent sections. Separating a Cobol program into paragraphs is a tedious task, and since the program would be read character by character, it increased the amount of coupling between the file I/O and the calculations the program needed to complete, which dismissed Cobol from being used. Fortran's level of coupling in regards to the calculations and the file I/O was the same as Cobol's, however the coherence of the functions that could have been created would have been much better than Cobol. Ada was ultimately chosen from the ability of being able to divide the entire program into separate packages, which resulted in a low level of coupling.

## Flow chart: Ada program



Flow chart: FORTRAN and COBOL program




When comparing character and string manipulation between Ada, Cobol, and Fortan, all 3 of these languages fail to exceed each other. This was the main downside of using any of these languages in contrast to using a new language. If a modern language could be used in the implementation of this project, Java would have been chosen for implementation mainly because of the simple file I/O and string manipulation. The project could have been entirely written in a single class and would have been shorter and more coherent. The downside to using Java is that it is a slow language and older machines cannot run Java programs.

This program was implemented into 3 sections: rbreaka4.adb, fileIO.adb and readability.adb. These sections were divided into different tasks that the program was required to accomplish. The package "fileIO" simply read and stored the entire contents of a file into a string. The package "readability" took in a string and calculated its readability value. This leaves the procedure "rbreaka4" which does the necessary function and procedure calls from the previously mentioned packages to link them together.

The main issue with the actual implementation of the assignment was not the coding segment, but the actual problem itself. First, it is impossible to have a word that contains 0 syllables, which was eventfully tested for. The file could contain 2 end-of-word/sentence characters (ex 2 spaces) in a row, which also needed to be check so that the program did not count an extra sentence/word. There was also the problem of a dash ('-') counts as an end of a sentence as well as a hyphenated word. Since there is no particular way to handle a dash, comparing the results with other sources proved to be problematic, as this implementation treats two dashes in a row as an end-of-sentence, where as other sources ignore them. Finally, there are many different methods of counting syllables. They are revolve around vowels, however, in situations like "Skywalker", there are three syllables ('y', 'a', 'e') but the program will report two syllables for this word('a', 'e') because a 'y' only counts as a syllable at the end of a word. Another example is the word "they." This program counts two syllables, instead of one. From writing this program, the current system of syllable counting contains some flaws, but is still accurate beyond the edge cases.