readabilityCalculator.java – System Manual

This program parses an input file, analyzes its structure, and prints out readability statistics.

Program Structure

This program is made of seven methods, including the main.

Main()

Return :void

Arguments: String[] args

The main method handles the sentinel loop that listens for the string QUIT to exit the program. If the use types something that is not the sentinel, the functions lineReader() and lineAnalyzer() are called. Then cleanup is done for the next file, the user is prompted for another filename, and the sentinel loop repeats. If QUIT is input, the program exits with status 0.

LineReader()

Return: Void

Arguments: List<String> linesInFile, String input

The lineReader() method is responsible for reading each line of the file. It uses a try-catch-finally statement to handle the opening of the file. In the try section, it scans each line into a string variable called line. Line is then split on spaces, and assigned to an array called parts. Then each item in parts is put into linesInFile, a List<String> that contains one word in each cell. Due to array limitations, in the program's current form, each paragraph must be 100,000 words or less. If a FileNotFoundException is thrown, an error message is printed, and the program checks the file extension. If the extension is wrong, a message is printed and execution is passed to the finally statement. If the file extension is okay, a generic message is displayed and execution is passed to the finally statement. In the finally statement, the fileReader scanner is closed to prevent resource leaks.

LineAnalyzer()

Return: Void

Arguments: List<string> contents, String input

The lineAnalyzer() method is responsible for doing prepwork for vowelCounter() and sentenceCounter(), calling these methods and handling the final analysis. Empty entries from spurious line breaks are removed, the List<String> is converted to an array for easy handling, and then each item in the array is sanitized by the following procedure:

- The string is converted to lowercase.

- Ending e characters are removed.
- Double vowels are converted to “a” characters.
- Words without any vowels are converted to “a”

This allows us to count variables by the number of “a” characters present in the strings. VowelCount() is then called, followed by sentenceCount(). The scores are calculated, and then FlechReadingEasePrint is called, followed by FlechKincaidPrint.

VowelCounter()

Returns: Int

Arguments: String[] contentsArray, int vowelCount

This function iterates through each string in the String array by character, checking if it is an 'a'. If it is, the variable vowelCount is incremented. After iterating though the entire array, the variable vowelCount is returned.

SentenceCounter()

Returns: Int

Arguments: String[] contentsArray, int sentenceCount

This function iterates through each string in the String array by character, checking if it is a punctuation mark. If it is, the variable sentenceCount is incremented. After iterating though the entire array, the variable sentenceCount is returned.

FlechReadingEasePrint()

Return: Void

Arguments: double rawScore, String filename

This function takes the rawScore calculated in LineAnalyzer() and determines the grade level against a predetermined grade scale. It prints a human-readable string, substituting the file name, rawScore and gradeLevel where appropriate.

The formula for this score:
FRES = 206.835 - 84.6 * (Number of Syllables) / (Number of Words) - 1.015 * (Number of Words) / (Number of Sentences)

FlechKincaidPrint()

Return: Void

Arguments: double rawScore

This function takes the rawScore calculated in LineAnalyzer() and determines the grade level against a predetermined grade scale. It prints a human-readable string, substituting the rawScore and gradeLevel where appropriate.

The formula for this score:
FKRA = 11.8* (Number of Syllables) / (Number of Words) + 0.39 * (Number of Words) / (Number of Sentences) - 15.59

Program Limitations

This program is designed for English, because the only adjacent vowels allowed in English are pairs. To modify this behavior, modify the regular expression in the lineAnalyzer function. As stated above, the paragraphs must be 1000,000 words or less.