**CS 110 hw#4 (classes)**
**Flesch – Kincaid Grade Level Introduction**:

The **Flesch Reading Ease** index is a tool for estimating the reading comprehension level necessary to understand a written document. For a given document, the Flesch readability index is an integer indicating how difficult the document is to understand, with *lower* numbers indicating *greater* difficulty. For example, the table below shows typical Flesch readability index values for some common(and some not-so-common) reading material:

| Material | Flesch Index |
|---|---|
| Comics | 95 |
| Consumer Ads | 82 |
| Sports Illustrated | 65 |
| Time | 57 |
| New York Times | 39 |
| Auto Insurance | 10 |
| IRS Code | -6 |

There is also an “improved” version called **Flesch/Kincaid Grade Level** which provides the grade level readability of a document (6.2 would be a 6th grade reading level in the US).

Some word processors are able to compute the Flesch/Kincaid Grade Level.
In Microsoft word:

1. Click the **Review Tab**, **Language, Language Preferences, Proofing** : check the box for show readability statistics, OK
2. Now choose the **Review Tab/Spelling and Grammar** and the readability stats will be displayed after any spelling or grammar errors are fixed or ignored.

-------------------------------------------------------------
**Formula for computing the Flesch/Kincaid grade level for a document:**

1. Count the number of words in the document.

[each group of continuous non-blank characters with beginning and ending punctuation removed counts as a word]

2. Count the number of syllables in the document.

[each vowel in a word is considered one syllable subject to: (a) -es, -ed and -e (except -le) endings are ignored; (b) words of three letters or shorter count as single syllables; and (c) consecutive vowels count as one syllable.]

3. Count the number of sentences in the document.

[periods, explanation points, colons and semicolons serve as sentence delimiters]

4. Compute the grade level as:

Flesch-Kincaid Grade level =
(0.39 * words/sentences) + (11.8 * syllables/words) -15.59

-----------------------------------------------------------

**Assignment Description -- **use folder name: **hw4_flesch**

**Collaboration:**

For this specific assignment, you may collaborate with any and all students in our class (either section of 110) and the TAs for this course for this semester. This does NOT mean that multiple students sit and work on the same files together then copy the file or parts of it to each student who worked on it. Students should NOT be passing code around. Each student should be working on his/her own but discuss with other students as they go. Consider the collaboration similar to what you would do if you asked a TA a question, only the "TA" can be another student in our class. Each student must turn in his/her own files and significant collaborators must be named in the comments at the beginning of the **Flesch.java** file.

------------------------------

In a nutshell, you will write and test two classes and write an application that uses your classes to compute the Flesch-Kincaid grade level for a document. Class **Word** represents a single word (as defined by the Flesch directions) and provides a method for counting the number of syllables in a word (also defined by the Flesch directions). Then you will write and test the **Sentence** class. The **Sentence** class represents a "sentence" string (possibly containing Flesch words). It provides methods for counting the number of words in the sentence and extracting each word from the sentence one by one. Then you will write an application ( a class with a main method) which uses **Word** and **Sentence** objects to compute the Flesch-Kincaid grade level of a text file document. Details below.

**Word.java, WordTest.java**

Implement the **Word** class as documented in **Word.html**. Notice that the rules for how syllables are counted should be described in the documentation for the countSyllables method. Write a test program for your **Word** class in the file **WordTest.java**. This test program should contain only a main method. The main method should create a number of **Word** objects and test each of the instance methods. Be sure to select test cases which are sufficient to fully test your **Word** class.
(What is a "word" according to Flesch?)

**Sentence.java, SentenceText.java**

Implement the **Sentence** class according to the description which appears in **Sentence.html**. The easiest approach to implementing the **Sentence** class is to use a StringTokenizer as part of the instance data. You may also find that the endsWith instance method in the String class is useful in creating your **Sentence** class.

Notice that the **Sentence** class uses the **Word** class. Thus, your **Word** class must be completed before you can write the **Sentence** class. Write a test program for your **Sentence** class: **SentenceTest.java**.

(What is a sentence according to Flesch?)

**Flesch.java (the application)**

In a class named **Flesch**, write a main method that uses your **Sentence** and **Word** classes to compute and display the Flesch-Kincaid grade level of a document. Your Flesch application should prompt the user for the name of the file to be analyzed and then compute and display to the console the name of the file processed, followed by the number of syllables, number of words, number of sentences, and the Flesch-Kincaid grade level.

**Additional requirements**: javadoc
Your program must be commented for javadoc. The only required tags are:
@param and @return, and of course the function documentation should appear quite like that provided in the files: Word.html and Sentence.html. (You will also have the file Flesch.html.)
See Handouts for javadoc and JGrasp. You will NOT turn in the documentation files.
<u>Questions to consider</u>:

1. In which part of this program will the sentences be isolated from the input file?
   (A "line" of the file is not necessarily a "sentence".)
   Hint: See p. 742. Reading one character at a time until you reach a sentence-ending character may be a good way to isolate a sentence from the file.

2. How many times should the program read through the input file?

3. Should all the words be stored in an array?

4. ?

**Input Files:**
Here are some files to test your program (is everyone else getting the same numbers as you?):
hanselgretel.txt
quote.txt

**Files to turn in (within the hw4.zip file):**

Word.java
WordTest.java
Sentence.java
SentenceTest.java
Flesch.java
(no javadoc created files)

**Grading** : see grading form

See:
http://flesh.sourceforge.net/

How do your results compare to the above grade level calculator?
How do your results compare to the Microsoft Word grade level calculator?