Assignment	Write a program that computes the *Flesch Readability Index* for the contents of a file or URL. The program can be run from either the command line or a graphical user interface, as described below.
What to Submit	1. Create a project named readability and commit it to Github classroom using the repository that will be created for you. To create a repository go to this URL adn "accept" the assignment: https://goo.gl/evwTof. 2. Create a *runnable JAR file* of your application with the name `Readability.jar` in the top level folder of your readibility project.
Evaluation	1. Implements the requirements and computes the Flesch Index correctly. 2. Good OO design. Separate UI from application logic; use polymorphism, state machine, and State Design Pattern instead of "if". 3. Code quality: follows *Java Coding Convention* for this course, code is well-documented and easy to read.

Requirements

1. The application can be run from either the command line or a GUI interface. You can get partial credit just for implementing only the command line version.

1.1 **Command Line Mode**: If the user supplies any arguments to the program's Readability.main() method, then run in command line mode. **Each** of the command line argument (args[]) isthe name of a file or URL.

Process **each** file or URL and output the readability statistics for each argument separately (see example below). If a file or URL cannot be processed, print an error message and *continue to the next command line argument*. Exit after all command line arguments have been processed. (See example below.)

> Command line arguments are passed to main as the args String array: main(String [ ] args).

1.1 **GUI Mode**: If there are no command line arguments then the program should run a graphical user interface and process commands interactively.

2. **Files and URLs**: The program should read data from either a file or URL.

An argument is a URL if it begins with "http://", "https://", "ftp://", or "file:/". Examples of URLs are http://se.cpe.ku.ac.th/raven.txt and file:/temp/raven.txt. In general, a URL has the form "protocol://host/path" where "protocol" is any string of alphabetic characters. If an argument is not a URL then treat it as a filename.

See below for hint on how to open a URL as an InputStream.

3. The program processes each input file and computes the Flesch Readability Index.

4. The files may be of any length (even very big) so that program **must not** try to buffer the entire file in memory! Points deducted for programs that try to hold the entire file in memory.

The program should scan (read) each input file **only once** (don't make multiple passes over the file).

5. After analyzing the file, print results. In Command Line mode, print results to System.out. In GUI mode, show the results in a GUI form.

6. In GUI mode include your name in the title bar, e.g. "`Readability by Fatalai Jon`". You might also have a Help => About menu item that displays your name.

Package Requirement

The **base package** for the application should be: **`readability`**.
The **main class** for running the program is: **`readability.Readability`**.

If you use the wrong package, your program will not be graded.

Requirement for Original Work

All work must be your own. You can discuss the design for implementing the program, including UML diagrams and general programming ideas (but not actual code from your program), but you may not provide or accept help in coding or debugging, except from the instructor or TA. Don't share your code with other students. If you have problems with either design or coding, please ask the instructor or TAs for help.

I'm sure no one will copy, but for the record: the penalty for copying is a grade "F" for the course and to be reported to the Faculty of Engineering for disciplinary action (which may include suspension).

The Flesch Readability Index

The Flesch Readability Index is a measure of how easy or difficult a document is to read. The Flesch Index helps authors to write documents at a level appropriate for their intended readers.

For any text document, the Flesch index computes a score based on sentence lengths and word lengths in the file. The higher the score, the *easier* the file probably is to read. The index scale is:

Index Value	Readability
Above 100	4th grade student (elementary school)
90+ - 100	5th grade student
80+ - 90	6th grade student
70+ - 80	7th grade student
65+ - 70	8th grade student
60+ - 65	9th grade student
50+ - 60	High school student
30+ - 50	College student
0 - 30	College graduate
Less than 0	Advanced degree graduate

Some examples are:

	Flesch Index
Comic Books	95
Consumer advertisements	82
Sports Illustrated magazine	65
New York Times newspaper	39
Auto insurance policy	10
Internal Revenue (tax) Code	-6

Formula for the Flesch Readability Index

The Flesch Readability Index is computed using the formula:

Readability Index = 206.835 - 84.6*(Number of Syllables / Number of Words)
- 1.015*(Number of Words / Number of Sentences)

Words

A *word* is any sequence of characters delimited by whitespace or punctuation such as , . ; -- ! ? ( ) [ ] /. A long dash (like -- or ---) separates words but not a single dash, as in anti-theft (one word).

A *word* must contain only letters, hyphen (-), and apostrophe (it's) **and** it must contain at least one vowel **and** must begin with a letter (can't begin with - or '). If a group of characters does not meet these requirements, then **don't count it** as a word.

Examples of Words: hello the I a am thy it's its' anti-static anti- me

Not Words: Dr me2 thx F anti_static

A whitespace character is any char such that Character.isWhitespace(c) is true. In most languages, whitespace means space, tab ('\t'), carriage return ('\r'), newline ('\n'), vertical tab ('\v'), or formfeed ('\f').

Syllables

Count *syllables* in *words* using these rules: each group of *adjacent* vowels (a,e,i,o,u) counts as one syllable. 'y' counts as a vowel if its the first vowel in a sequence (as in sky, tythe) but counts as a consonant if it comes after another vowel (layout). A trailing "e" at the end of a word doesn't count as a syllable unless its the *only vowel* in the word (the, he, me). These are the same rules as in Lab 9 Syllable Counter.

A hythen can be treated as a consonant (separates vowel sequences). "anti-aging" is 4 syllables and "re-install" is 3 syllables.

Sentences

A *sentence* is any sequence of words ending with a period, semi-colon, question mark, or exclamation mark **and** followed by a whitespace character or EOF. A sentence must contain at least one word. This avoids counting empty sentences. The trailing whitespace (after .!?;) is important to avoid counting I.B.M. or "apple." in "apple.com" as sentences.

A blank line also ends a sentence. A blank line is two newlines separated by zero or more white-space characters. A regular expression is: `\n\s*\n` or (better) `[\m\n]+\s*[\m\n]+`. Note that in Strings you must use 2 backslashes: `"\\n\\s*\\n"`. For example, the first 2 lines below count as one sentence because they are followed by a blank line:

Alice in Wonderland
by Lewis Carroll

One day Alice went for a walk...

Examples

After computing the index for a file or URL, output all values and the readability level, like this:

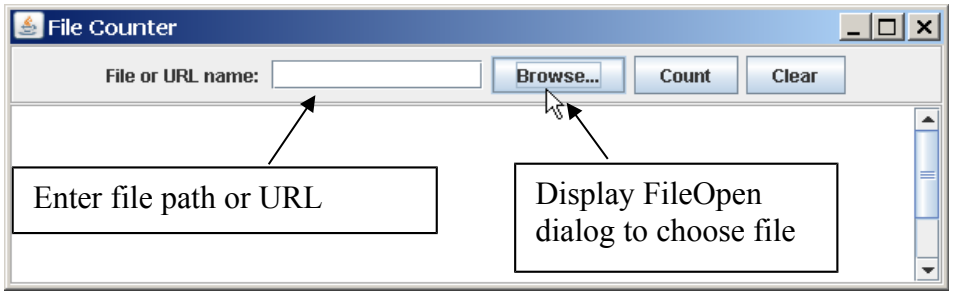
```
c:\>  java readability /temp/sample.txt

Filename:                /temp/sample.txt
Number of Syllables:     172
Number of Words:         96
Number of Sentences:     10
Flesch Index:            45.5
Readability:             College Student
```

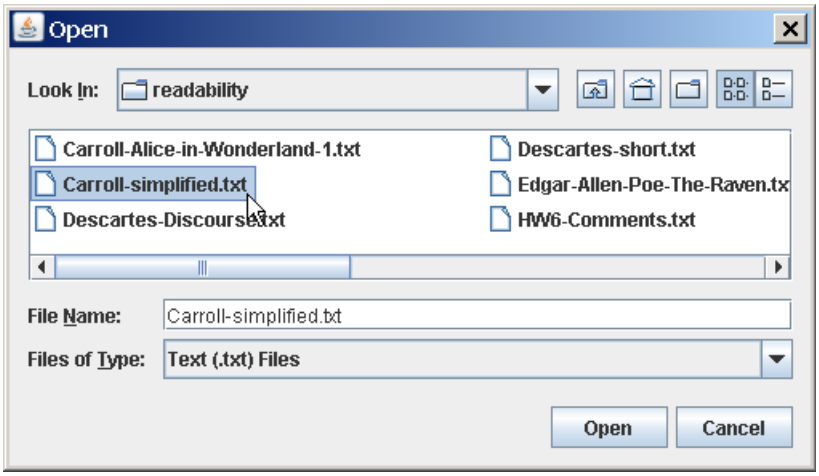
Graphical User Interface

In GUI mode, all input and output should be on the graphical interface. That means the results of processing a file are shown in the GUI. You can design the appearance as you like. For opening a file, please provide a File-Open dialog as in most applications. See resources below for help on this.

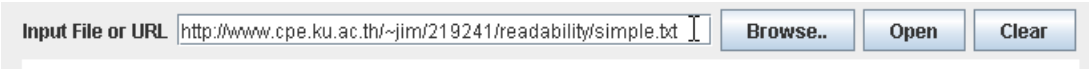
1. The program displays a Window where user can type in a URL or file path. User can also open a local file by clicking the "browse" button:



2a. If the user clicks "Browse..." then open a FileOpenDialog. Use a filter so that only `.txt` files and directories are displayed by default.



2b. The user can also type the file path or a URL in the window. If he enters a URL then read the file contents from the URL (see reference below for how to).



3. You can distinguish between a filename and a URL by the format. A URL has the format:

```
protocol://location/path
```

For example:

```
https://www.cpe.ku.ac.th/~jim/219141/readability/somefile.txt
file:///C:/temp/somefile.txt  (a local file can be specified as a URL)
file:/temp/somefile.txt       (for file: you can omit the //host part)
ftp://se.cpe.ku.ac.th/docs/howto.txt
```

Program Structure and Operation

1. The text file that the program reads may be **huge**, including an entire book.

Therefore, the program ***must not*** buffer the entire file in memory. Using an iterator design or state design, you shouldn't need to buffer more than one line of input.

2. The program should read the input **only once**. Don't make multiple passes through the same file/url.

3. Use a *State Machine* for counting syllables in a Word.

4. Divide your application into multiple classes. At least you should have separate classes for the "views" which are Console UI and Graphical UI, and separate class for the Readability calculation. A separate class to do the analysis (counting) of an input stream would be a good idea.

5. The program must follow the coding conventions for the course and have meaningful, useful Javadoc comments, esp. class Javadoc comments.

Programs that don't follow these conventions won't be graded.

Resources

1. Sample text files and example code: **http://www.cpe.ku.ac.th/~jim/219141/readability/**

2. JFileChooser and FileFilter: see "How to Use File Choosers" in the *Creating a GUI with JFC/Swing* Trail of the Java Tutorial.

3. Opening and Reading a URL: PowerPoint slides on the class web page, Read-File-Or-URL.ppt.