# Programming Assignment: How Easy to Read is Your Writing?

## Before you begin: Getting help

Before you begin, don't forget that you should **use the discussion forums to get help anytime you are stuck on this assignment**. Also, please visit the forums to help your fellow learners by contributing answers to their questions. We're all in this together!

We also have included an FAQ for this assignment as a reading immediately following this assignment, so check out that document for more help if you get stuck.

## Assignment Overview

In this first programming assignment, which is the first part of the project you will develop throughout this course, you will write code to analyze the reading level of a piece of text. The Flesch Readability Score is a measure of the reading complexity of a piece of text. Developed by author Rudolf Flesch, it is a measure that approximates how easy a piece of text is to read based on the number of sentences, words and syllables in that text. Higher scores indicate text that is simple to read, while lower scores indicate more complex text.

In this programming assignment you will write the back-end code for calculating Flesch readability scores, and then you will integrate this back-end capability with the front-end application we provide that you will be adding to throughout this course.

## Getting Set Up

**1. (If not already done) Download and set up the starter code**

Before you begin this programming assignment, you need to download and set up the starter code. You should have already followed the instructions for setting up Java and Eclipse. Make sure you have the most recent version of the starter code by checking the date the starter code was last updated. You are welcome to use any IDE of your choice, but we provide the starter code as an Eclipse project, so you'll probably find it easiest to work in Eclipse.

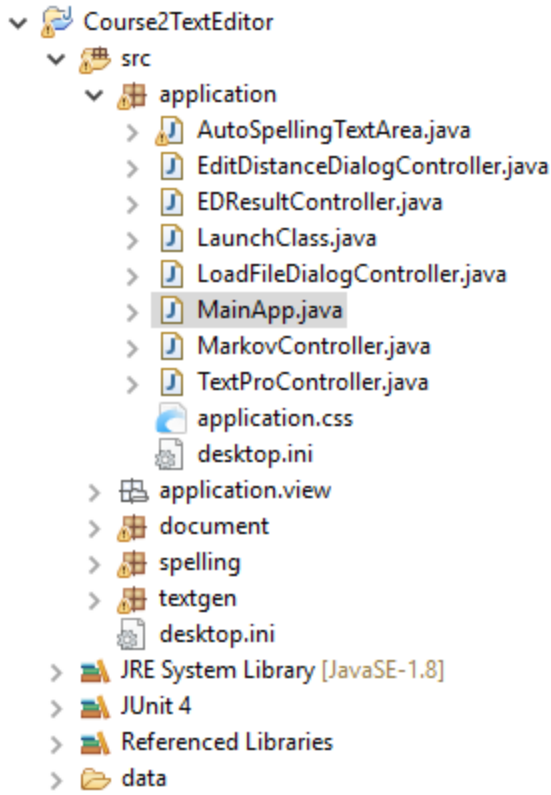
Make sure you've followed the setup instructions even if you've previously worked with Eclipse, and even if you took the first course in our series, because you'll need to make sure you have some additional pieces installed and set up (e.g. At least Java 1.8, JUnit).

**2. Verify that the front-end runs**

Make sure the project you downloaded and set up for this course is open in eclipse.

As you saw in Mia's demo video, all of the functionality you will implement in this course will be integrated into a text editor application. You can already run this application, but it won't do much.

Run the text editor by running the MainApp class inside the application package. You can do this by expanding the application package in the package explorer and then selecting the MainApp.java file and running it.




You will see a text editor window open. You can type in the text window and load text using the "Load" button. None of the other buttons work yet (you can click them, but they don't do anything). But they will soon...

**3. Orient yourself to the starter code**

Open the starter code for this assignment by expanding the document package in the Package Explorer window. There you will see several files, but the two that are relevant to this assignment are Document.java and BasicDocument.java. Open these by double-clicking on them.

Notice that Document is an abstract class. BasicDocument will implement the abstract methods in the Document class using the guidelines below described in the parts below.

# Assignment Details

This assignment is divided into two parts. You will submit a separate file for each part, as described below.

## Part 1: Implement the missing methods in BasicDocument.java

**1. Implement getNumSentences, getNumWords and getNumSentences** following the comments about how they are supposed to work that you will find in your starter code (the version uploaded June 22, 2016 or later has the most detailed comments). You'll probably also want to implement countSyllables(String) in Document.java to be called in getNumSyllables.

You must follow the definitions of what constitutes a syllable, word and sentence given in the documentation for each function exactly to pass the graders.

We have provided you will several test cases in main as well as a helper method you can use to write your own tests.

If you get stuck or want any hints about this section, you can find some helpful tips linked from the online version of these instructions.

If you get totally stuck on getNumSyllables, you can find the solution code for the countSyllables helper method from Document.java and the getNumSyllables method linked from the online version of these instructions.

**2. Part 1 Bonus (purely optional, just for fun):**

Our REGEX are pretty naive. For example, the word 7.5 causes real problems for our expressions as a “.” usually denotes the end of a sentence. If you want to learn more about REGEX, you can improve upon our approach. But if you want to do this, put it in a separate class (like ImprovedDocument) and make sure you don't use it to produce your grading output.

## Part 2: Implement the getFleschScore method in Document.java

**1. Fill in the method getFleschScore() in Document.java** to calculate the Flesch Score for the text in the document. You should use the following formula, and make calls to the getNumSyllables, getNumWords, and getNumSentences you just implemented.

$$Flesch\ score = 206.835 - 1.015\left(\frac{\#\ words}{\#\ sentences}\right) - 84.6\left(\frac{\#\ syllables}{\#\ words}\right)$$

You should test your code by calculating the Flesch score by hand on some very basic documents and then calling your method from main to make sure it's giving the same output. Or you can go ahead and run our grader before you submit, which you can find in the same package.

## Part 3 (optional, nothing to submit): Have fun with calculating the readability of text that you find

**1. Make your code work with the text editor application**

Currently, the text editor application is not set up to use the BasicDocument class (it uses EfficientDocument, which you will implement next week). So if you want your text editor application to calculate the Flesch score (after you finish part 2), you must change one line of code in the file LaunchClass.java, which is in the application package. In the method

public document.Document getDocument(String text)

change "document.EfficientDocument" to "document.BasicDocument". Remember to change this back next week!

Now that you've implemented the Flesch score, your GUI interface will automatically include this functionality. Try running it again, and playing around with calculating the Flesch score of various documents, either that you have produced or that you find on the web. We'd love to see fun things you discover on the discussion board!

# What and how to submit

**1. Create a zip file containing only Document.java and BasicDocument.java.** You can find these files in the workspace directory you set up on your computer's filesystem.

**2. Upload this zip file (that contains both Document.java and BasicDocument.java) for BOTH part 1 and part 2.** In part 1 we are grading you on only the methods you wrote for part 1, while in

part 2 we are grading you on only the methods you wrote for part 2. But we need both files for both parts of the grading.

**3. Submit!** Once you submit, grading will take a couple of minutes. During this time you will see a score of 0 displayed. This does not mean you got a 0! Your correct score will refresh once grading is done.

You can see exactly what test cases we used by running the code for the grader that we provided with your starter code (BasicDocumentGrader.java) and checking the file the grader is reading from. If you get errors, use these grader files to help you fix your errors and resubmit.

At this point our graders are very well-tested and are extremely unlikely to have bugs. For this assignment and all assignments in this course, if you have a problem with your submission we suggest you seek help in the forum rather than submitting a bug report.