* An input file contains sentences, one per line. Each sentence contains words delimited by <space> character. Each sentence terminates with a period.
* Each sentence is made up of alphanumeric words (characters in the set [a-zA-Z0-9]).
* The program should process the input file word by word.
* The program should do the following.
  + Rotate each word in a sentence to the right by x places where x is the index of the word in the sentence.  
    *Note: Indexing starts from 1. So first word is rotated by 1 place, second by 2 places and so on.*  
    *Note: Only the characters of a word should be rotated. The order of words in the sentence should remain as is.*  
    *Note: The rotation should be case senstitive. An upper case character in the input should remain in upper case in the output and lower case character should remain in lower case.*  
    *Note: Period characters remain unchanged.*  
    For example, consider the sentence "Welcome to the course.". As it is mentioned that indices start from 1, the index of "Welcome" is 1, "to" is 2, "the" is 3 and "course" is 4.  
    We therefore need to rotate "Welcome" by 1 position, "to" by 2 positions, "the" by 3 positons and "course" by 4 positions to the right.  
    After performing rotation, the sentence would now read "eWelcom to the urseco.". This rotated sentence is to be written to the output file.
  + Calculate the following metrics and write them to the metrics file (one metric per line).
    - AVG\_NUM\_WORDS\_PER\_SENTENCE - Average number of words per sentence. Round to 2 decimal places. Format: **AVG\_NUM\_WORDS\_PER\_SENTENCE = <value>**
    - AVG\_WORD\_LENGTH - Average length (number of characters) of a word in the input file. Round to 2 decimal places. Format: **AVG\_WORD\_LENGTH = <value>**

The following rules MUST be followed.

1. FileProcessor code has been given to you. This should not be altered. You should use the FileProcessor for reading in the input file word by word. Read the documentation to understand how the FileProcessor works.
2. The input file should be processed one word at a time.
3. The program should not read in all the input and store it in a data structure.
4. You should implement your own function for rotating a word.
5. You should implement your own function for calculating each of the metrics.

General Requirements

* Separate out code appropriately into methods, one for each purpose.
* You should document your code. The comments should not exceed 72 columns in width. Use javadoc style comments if you are coding in Java. Include javadoc style documentation. It is acceptable for this assignment to just have the return type described for each method's documentation.
* Do not use "import XYZ.\*" in your code. Instead, import each required type individually.
* **All objects, in Java, that may be needed for debugging purposes should have the "toString()" method defined. By default, just place a toString() in every class.**
* **Every class that has data members, should have corresponding accessors and mutators (unless the data member(s) is/are for use just within the method.).**

FileDisplayInterface should declare a method called writeToFile().

StdoutDisplayInterface should declare a method called writeToStdout().

Results should implement both these interfaces.

The word rotation code and metrics calculation code should not directly write to the corresponding output/metric files. Instead, they should use an instance of Results class to store the results.

The driver code should also hold a reference to the shared Results instance, cast it to the appropriate interface and, after all the input has been processed, call the appropriate function to persist the data.

Note that the metrics calculator will hold a reference to a different Results instance compared to the reference held by word rotation code.

The directory structure has been updated. Please refresh the assignment description page.

You will have to create two new packages - wordPlay.handler and wordPlay.metrics.

wordPlay.handler package should contain the WordRotator class that holds code pertaining to the word rotation.

wordPlay.metrics package should contain the MetricsCalculator class that holds code pertaining to the metrics calculation.

Another important note regarding the assignment - DO NOT create static methods. Share references and use them to call methods.

Boundary condition confusion with Assignment-1

I hope this post clarifies any questions you may have regarding reading in of the input file and satisfying the boundary conditions.

READING IN THE INPUT FILE

- This has to be done only once and with the use of FileProcessor.

DETECTING END OF SENTENCE WHEN CALLING poll()

- The last word in a sentence returned by poll() would include the period character. You have to write your logic around this to detect end of sentence and line.

BOUNDARY CONDITION: MULTIPLE SENTENCES PER LINE

This one is our mistake. The FileProcessor code is written such that it is impossible to detect whether there are multiple sentences per line. Therefore, we will not test this condition and you are free to assume that the input file WILL have just one sentence per line.

BOUNDARY CONDITION: MISSING PERIOD CHARACTER IN A SENTENCE/LINE

This boundary condition is not possible to check with the existing code in file processor. Therefore, we won't be testing for this.

Note that apart from these two boundary conditions the remaining 4 boundary conditions will be tested for.

