1. I wrote the indexer in Java and, like the previous assignments, found it useful to reuse a Tuple class I had been suing. This class was crucial in mapping various data types together such as the IDs of the scenes and plays as well as the positions of the occurrences of the terms within those documents. Using a Map of terms to an ArrayList of these tuples, I was able to index said terms and add a positional index to the tuple if the term was already found. If it was not found, I simply add a new tuple corresponding to the sceneID and positional index within the scene. This Tuple is then mapped to the term in the overall index. I was unsure of how to handle the fact that my output would add multiple positional indexes of the same value to one Tuple corresponding to one sceneID, despite the fact that I am checking the values of each Tuple found within an ArrayList of tuples corresponding to a term in the overall index, assuring that if the sceneID of the current tuple in the list matches the current sceneID in the outer loop, that I add the *new* positional index to the *existing* Tuple. It has to do with the fact that I am indexing the set of tokens I derive from each set of text. I index the term when needed but it always returns the first occurrence's index. I am aware of a second parameter in indexOf() that utilizes a start index but am unaware of how to make use of this in the code. This method also does not work on ArrayList<String>, but on Strings themselves. I output my result set to index.txt.
2. The only library I used was suggested by the instructions and is the json-simple library for Java. I used the JSONObject class for each entry within the overall JSONArray. This array was obtained by first parsing the json file by using the JSONParser class and parse method. This allowed me to use the array by invoking the get("corpus") method. Iterating over each object I was able to obtain the key and value pairs and parse the given text, mapping the appropriate values in the indexing process.
3. A count could be misleading for a scene because it determines nothing about the positioning. Therefore, a document could contain a plethora of one word that is contained in a query, yet that word could be used arbitrarily in ways that may not even make sense. For example, a word could be repeated a thousand times in a row and the word count would in crease the probability by which that page will appear as a result, but it may not be an exactly relevant result. This could be fixed by coupling the word count with a positional index set so that an engine could look at both delimiters as a decision maker in returning results.