README

First, there are two different Indexers (that's how I named the classes that create the index), one for URL and the other one for the text file; both are subclasses of `IndexerAPI`, the only difference is how they retrieve the lines of text. I provide a function to load stopwords into a `ConcurrentSkipListSet` since they are simple strings, and another function to load the dictionary into a `ConcurrentSkipListMap` with the following type `Map<String, List<Definition>>`, this is a map that has words as keys and a list of Definition as value; I use a list because a word can have various definitions that have a type (e.g. noun, verb, etc.) and its text.

For the index creation the program iterates through the lines of the source given by the user (file or URL) and stores the words in a `Map<String, WordDetail>` using the ConcurrentSkipListMap implementation that is thread safe, necessary in this case, because multiple threads will update this map at the same time. `WordDetail` has the definitions and pages where a single word appears.

Methods for the API

There are various methods to interact with the generated index, these are:

- Get all words in the text ordered alphabetically
- Get all words in the text reverse ordered alphabetically
- Get unique words count
- Get total words count
- Get Top N most frequent words
- Get Top N least frequent words
- Execute all methods (Summary), this functionality uses all the previous methods.