**Computer Programming II  
Prof. Tom Wulf  
Lab 8: Tag/Keyword Extractor  
Spring Sem 2022**

**20 pts**

Learning Goals:

* Practice Java Swing GUI and text file I/O
* Use ADTs form the Java Collections Framework
  + Map
  + Set
* Computational linguistics
  + Tags
  + Stop or Noise Words

*This is a challenging multi-part assignment. I will discuss it in class. None of the programming is beyond your capability, but it has a bit more structural complexity that what I have asked you to do previously.   
Read the directions carefully, don’t just start coding. You have to plan your program carefully here.*

**Mini-lecture:**Computational linguistics is the study of linguistics using computers. It often involves large datasets. Related areas are Natural Language Processing, text-to-speech, etc.  
  
One way of looking at a document and comparing documents is to examine the frequency in which words occur. For instance, we would assume that an article with the word UFO in it many times is probably about UFOs… A document that does not contain the word would not be. (Early search engines used this approach to determine the documents to include in search results.)  
  
However, many words in the document are not relevant. We call these noise words or stop words. Pronouns, adverbs, prepositions, and commonly occurring words are noise words. Examples are: a, an, the, to, is, are, from, at, in, etc.  
  
After we exclude these words the remaining relevant words are called tags or keywords.  
  
We normalize our tags by making them all the same case or ignoring case when we process the documents.

**Directions:**

Create a Java swing GUI program that lets the user pick a text file (JFileChooser) and a stop or noise word file to scan for tag extraction.

* Display the name of the file you are extracting the tags from
* Display the tags and their frequency in a JTextArea/JScrollPane
* Allow the user to save the extracted tags and frequencies to a file.

**Scanning process:**

The program will open and scan the file and store the frequency of each word that occurs in the set of documents. (This is exactly like the Map example from our text book!) You will remove any non-letter characters and force the words to be all lowercase. You will end up with a map that contains <String> <Integer> where String is the word as the key and where the Integer value is the number of times that the word appears in the scanned documents. That is the frequency that the word occurs within the scanned files.   
  
However, one additional aspect is that you will implement a stop word filter. So before you add a word to the map, you first check to see if it is in the set of stop words and reject it if it is. You should load the stop word set from a local text file. It is likely that we will be able to find a good stop word file on the internet but if not it is easy to develop one by simply running the tag cloud generator several times as the stop words will appear with great frequency. Then simply add them to the stop words file. The stop words file is a text file with one word per line. It should be sorted in alphabetical order and all words should be lowercase. If you use a TreeSet or TreeMap it will be sorted as a by-product. You can also sort at the DOS cmd line or Shell using this cmd:  
  
 **type FILENAME | sort >NEWOUTPUTFILENAME**.   
  
CAUTION: There is no space between the >and the OutputFileName. If you do that you will have a leading space in the file name which is a real pain.

**Deliverables:**

1. Create an IntelliJ project named TagExtractor add it to GitHub source control.
2. Get a series of screen capture shots showing your program completes all the requirements here.
3. Download a Literature text file from Project Gutenberg run the extractor several times on it and adjust the stop words file as needed to get a good set of tags. Include the original file, the final stop words file, and the output tag file within the project.

A screenshot of a computer

Description automatically generated

1. Submit the link to your repo. Submit this document with the required screen captures, etc. Rename it as **LastName FistName\_Lab08.docx** using your name.   
   Submit this using the Canvas assignment mechanism.

A screenshot of a computer

Description automatically generatedA computer screen with a white square

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated