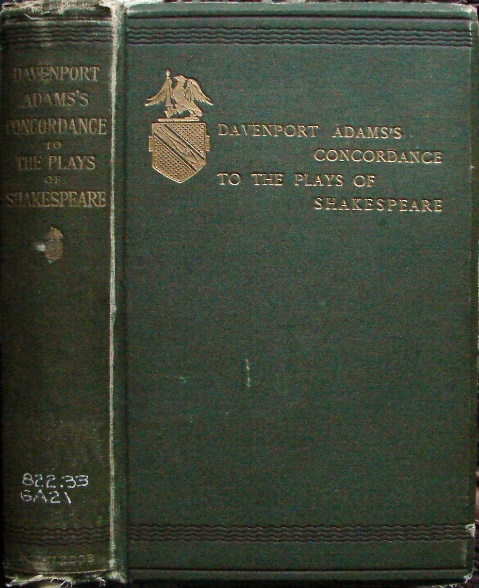
**CMSC204**

**Assignment #4**



A **concordance** is an alphabetical list of the principal words used in a book or body of work, listing every instance of each word with its immediate context. Only works of special importance have had concordances prepared for them, such as the Bible, Qur'an, the works of Shakespeare, or classical Latin and Greek authors, because of the time, difficulty, and expense involved in creating a concordance in the pre-computer era.

The first Bible concordance was compiled for the Vulgate Bible by Hugh of St Cher (d.1262), who employed 500 monks to assist him.

The reconstruction of the text of some of the Dead Sea Scrolls involved a concordance.

**Assignment Description**

Write a program that creates a concordance from some text. It will list all the words in alphabetic order followed by a list of line numbers where the word can be found in the text. Upload the initial files and your working files in the repository in GitHub you created in Lab 1, in a directory named Assignment4.

**Concepts tested by this assignment**

Hash Table,

Link List,

hash code, buckets/chaining,

exception handling,

read/write files using FileChooser

**Classes**

**Data Element -** ConcordanceDataElement,

ConcordanceDataElement implements Comparable<ConcordanceDataElement> and consists of a String (the word) and a reference to a LinkedList<Integer> (list of line numbers where word occurs). Follow the Javadoc provided for you.

**Data Structure -** ConcordanceDataStructure,

Implements the ConcordanceDataStructureInterface Interface that is provided.

You will be implementing a hash table with buckets. It will be an array of linked list of ConcordanceDataElements. The add method will take a word and a line number to be added to the data structure. If the word already exists, the line number will be added to the linked list for this word. If the line number for the word already exists, don’t add it again to the linked list. (i.e. if Sarah was on line 5 twice, the first line 5 would be added to the linked list for Sarah, the second one would not). If the word doesn’t exist, create a ConcordanceDataElement and add it to the HashTable. Two constructors will be required, one that takes in an integer that is the estimated number of words in the text, the other is used for testing purposes. Look at the provided Javadoc.

**Data Manager -** ConcordanceDataManager

Implements the ConcordanceDataManagerInterface interface that is provided.

The data manager allows the user to create a concordance file or a concordance list (ArrayList of strings). The input is read (from a file or string) and is added to the data structure through the add method. The add method requires a word and a line number. The line number is incremented every time a newline appears in the file or the string. Change all words to lowercase so that Now and now are considered the same word.

**Exception Classes**

IOException – created and thrown when user selects an input file that cannot be read (check out the methods of File).

**GUI Driver (provided)**

* User will only create a concordance file once they have entered an input file and an output file
* Show the text area only when the option to create from text is chosen.
* A FileChooser is used to select the input and output files. Use a filter so that user can only select .txt files.
* Inform the user if there is an error with the input file or the output file
* Use exception handling for the validity of the files.
* If creating a concordance from text, make sure the user has entered some text in the text area. Inform user if text area is empty.
* Display the concordance from the text in the text area.
* Provide a way for the user to “clear” the text area.

**Testing**

Create a JUnit Test - ConcordanceDataManager\_STUDENT\_Test.

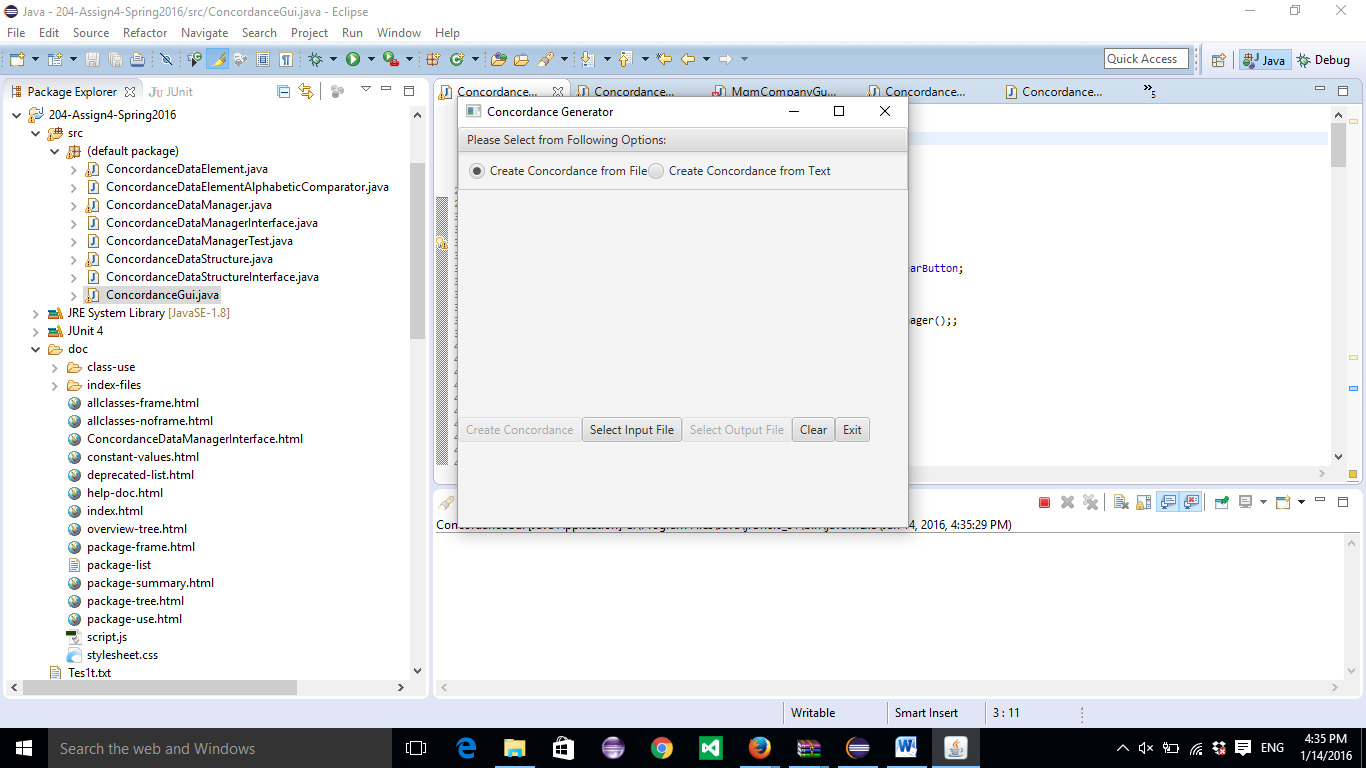
**Assignment Details**

There will be two ways to create a concordance. The first requires a document to be read from an input file, and the concordance data is written to an output file. The second reads the input from a string and returns an ArrayList of strings that represent the concordance of the string.

Because they are so common, **don't include the words "the" or “and”** in your concordance. Also, **do not include words that have length less than 3**. **Strip out all punctuation, except apostrophes that occur in the middle of a word, i.e. let’s, we’d, etc.**

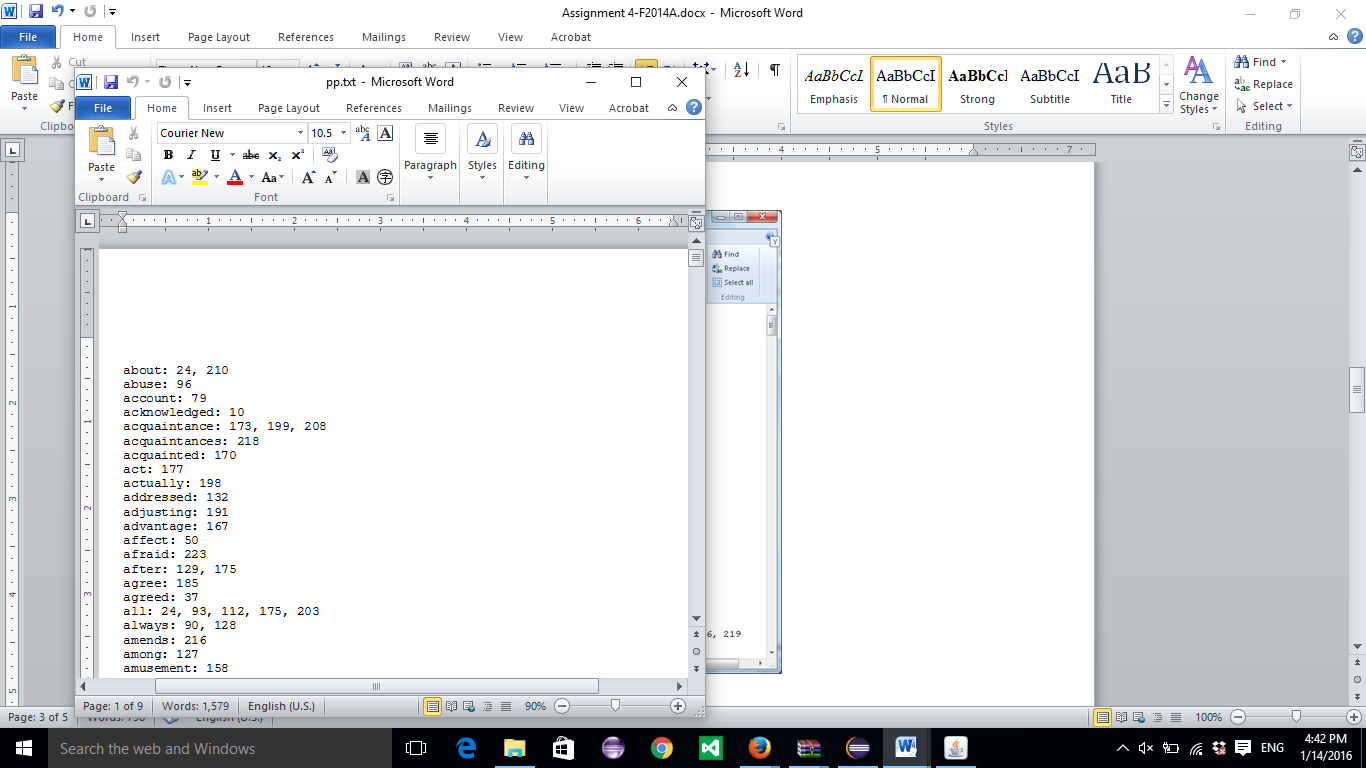
**Examples**

Example of creating a Concordance from an input file

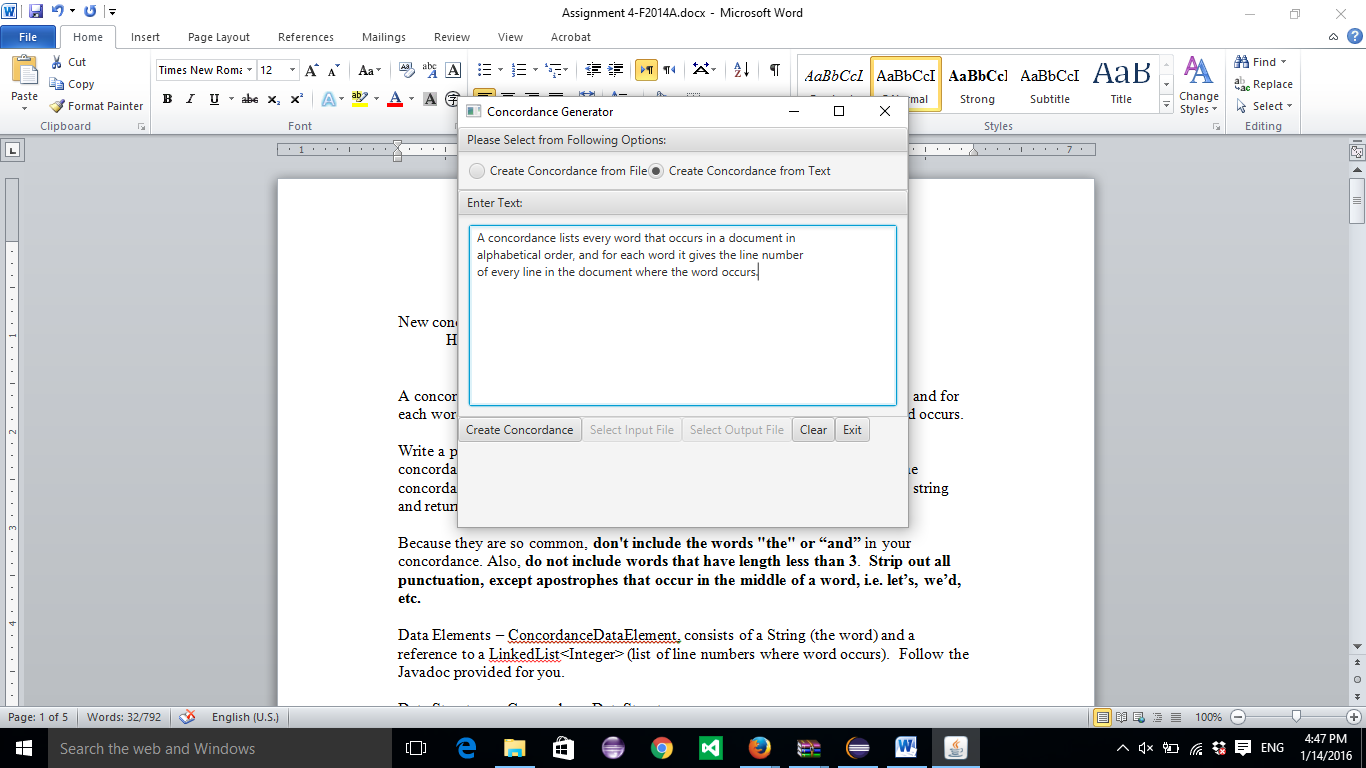


Select an input file and an output file. PrideAndPrejudice.txt was used.

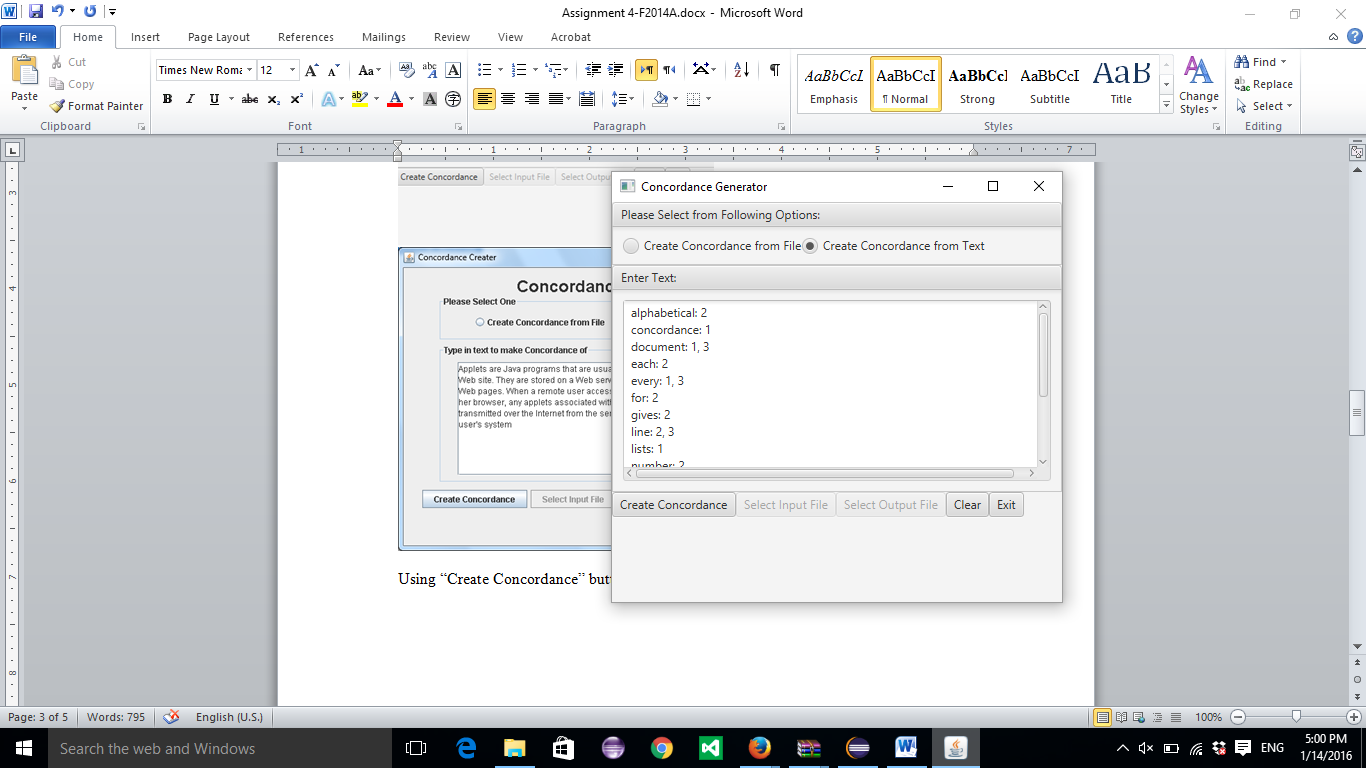
Sample of output file:



Example of Creating a Concordance from text:



Using “Create Concordance” button displays Concordance in text area



**Deliverables**

**Deliverables / Submissions:**

Design: UML class diagram with algorithm (pseudo-code) for methods

Implementation: Submit a compressed file containing the follow (see below): The Java application (it must compile and run correctly); Javadoc files in a directory; a write-up as specified below. Be sure to review the provided project rubric to understand project expectations. The write-up will include:

* UML diagram
* In three or more paragraphs, highlights of your learning experience

**Deliverable format:** The above deliverables will be packaged as follows. Two compressed files in the following formats:

* LastNameFirstName\_Assignment4\_Complete.zip, a compressed file in the zip format, with the following:
  + Write up (Word document) - reflection paragraphs
  + UML Diagram - latest version (Word or jpg document)
  + doc (directory) - Javadoc
    - * File1.html (example)
      * File2.html (example)
  + src (directory) 
    - * File1.java (example)
      * File2.java (example)
* LastNameFirstName\_Assignment4\_Moss.zip, a compressed file containing one or more Java files:
  + File1.java (example)
  + File2.java (example)

This folder should contain Java source files only