Assignment 4

CMSC 204

Spring 2016

**Concepts tested by this program**

Hash Table,

Link List,

hash code, buckets/chaining,

exception handling, read/write files (FileChooser)

A concordance lists every word that occurs in a document in alphabetical order, and for each word it gives the line number of every line in the document where the word occurs.

Write a program that creates a concordance. 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.**

**Data Elements** – ConcordanceDataElement, 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.

**Data Manager** – ConcordanceDataManager

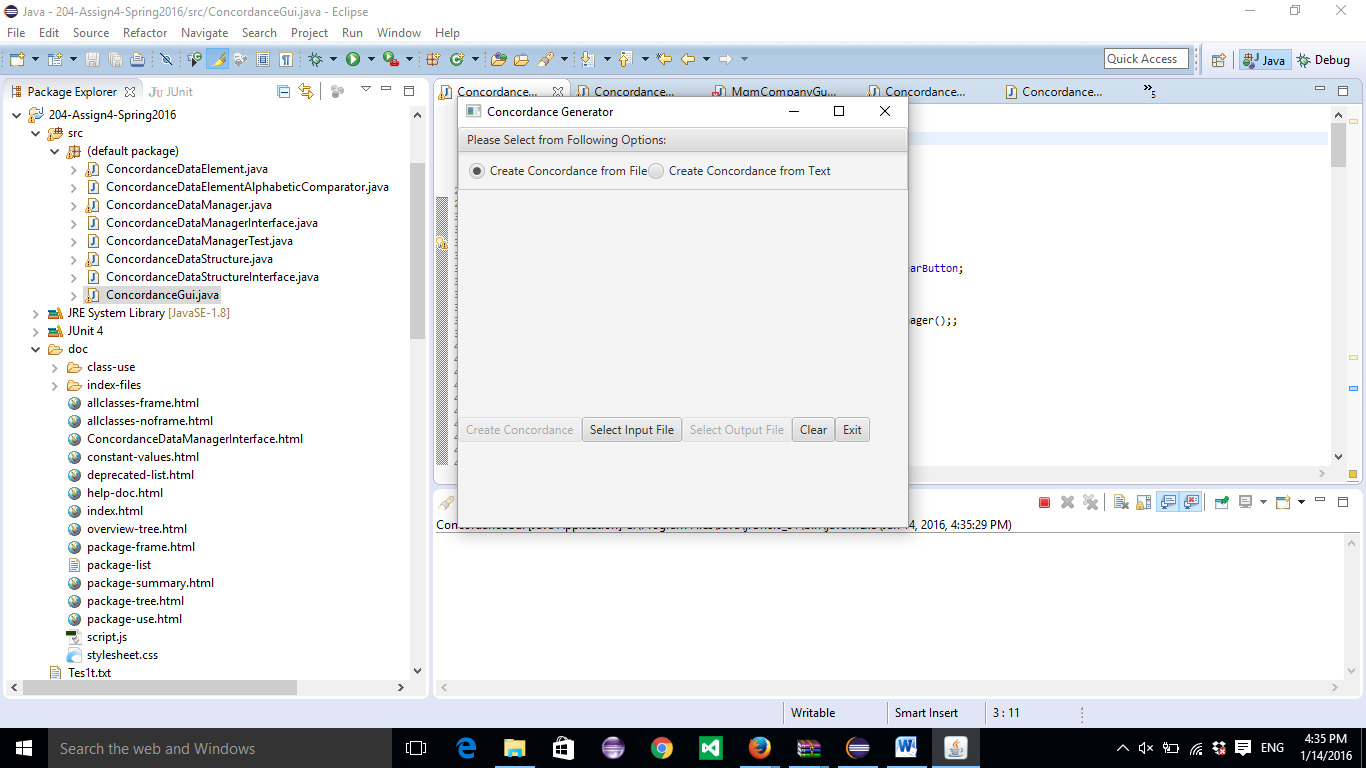
Implements the ConcordanceDataManagerInterface interface that is provided.

The data manager allows the client (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.

**GUI driver**

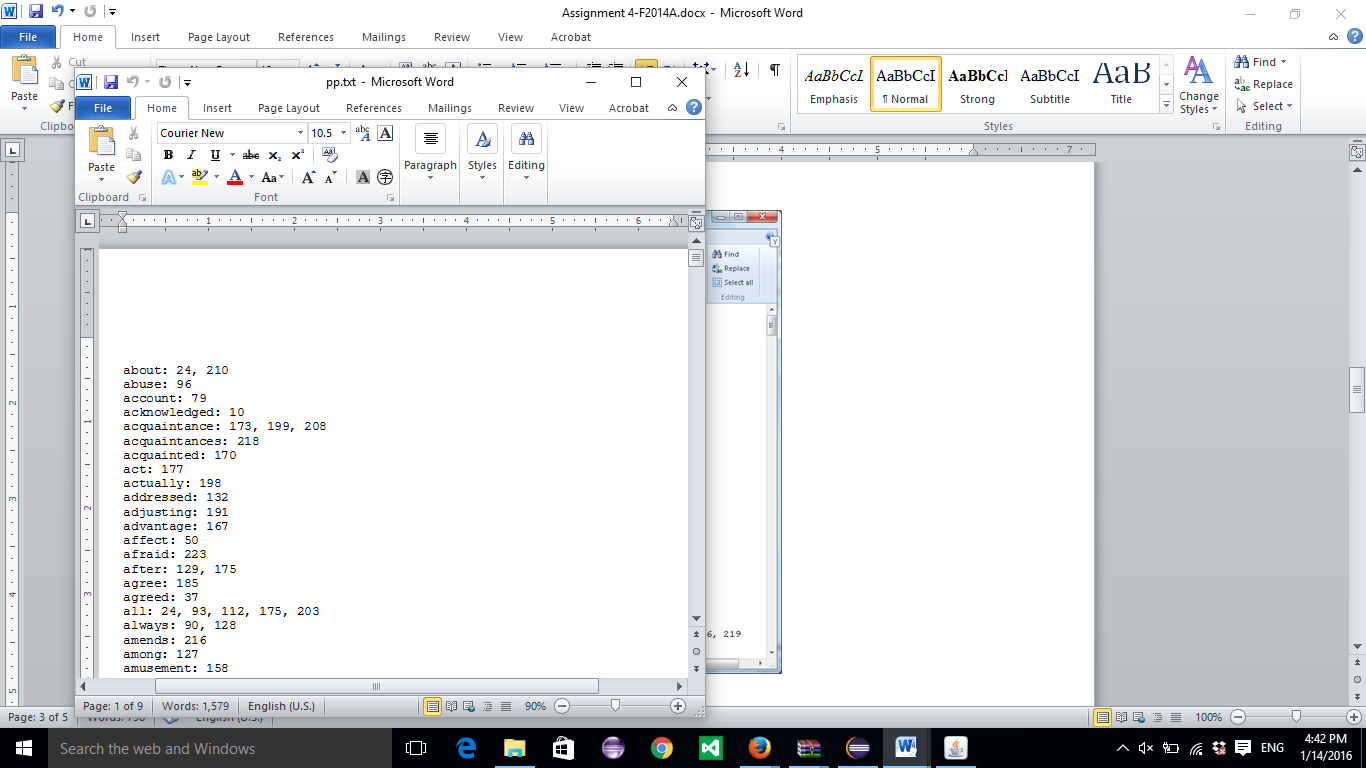
* Do not allow the user to create a concordance file until they have entered an input file and an output file
* Show the text area only when the option to create from text is chosen.
* Use a FileChooser for the user to select the input and output files.
* Inform the user if there is an error with the input file or the output file
* Use exception handling (try/catch) 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.

Example of creating a Concordance from an input file

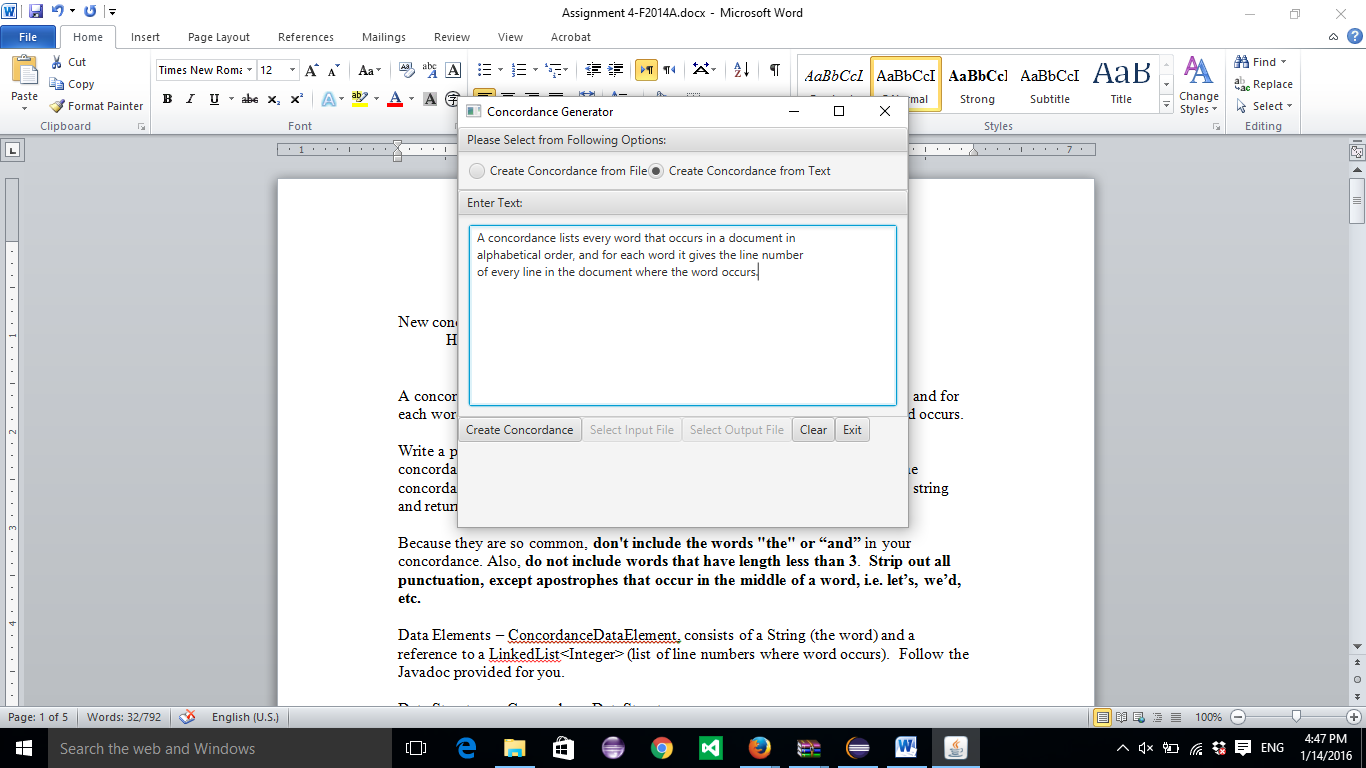


Select an input file and an output file. PrideAndPrejudice.txt was used.

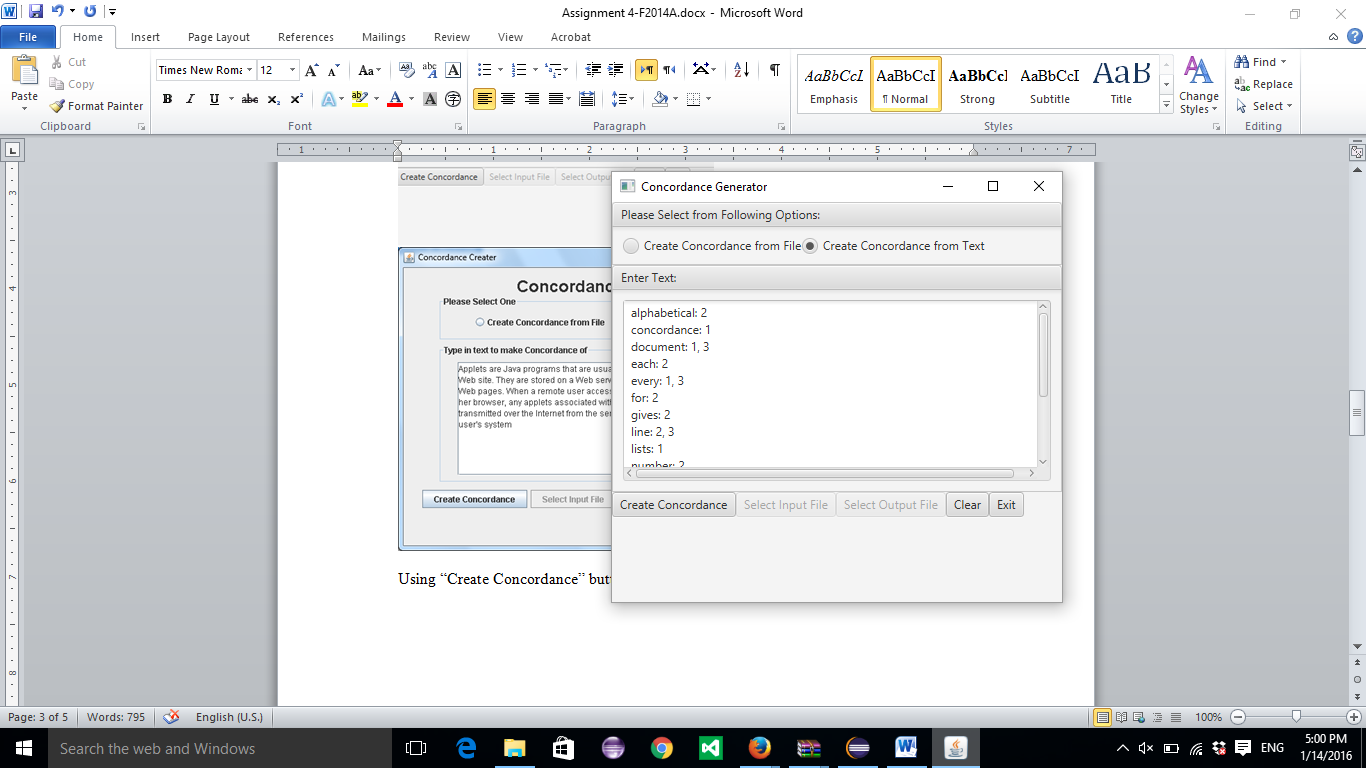
Sample of output file:



Example of Creating a Concordance from text:



Using “Create Concordance” button displays Concordance in text area



Deliverables:

Java files - The src folder with your driver (javafx application), data element, data manager and Junit Test (.java) files

Javadoc files - The entire doc folder with your javadoc for student generated files

UML Class Diagram (an image, not the proprietary format, must be a .jpg)

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

LastNameFirstName\_AssignmentX\_**Complete**.zip [a compressed file containing the following]

UML.jpg

Assignment 4 Checklist (filled in with YES or NO or ?)

doc [a directory] *please include the entire doc folder with the javadoc for*

*student generated files*

file1.html (example)

file2.html (example)

src [a directory] *contains your driver (javafx application), enumerated*

*class, data element, data manager and Junit Test (.java) files*

File1.java (example)

File2.java (example)

File\_Test.java (example)

LastNameFirstName\_AssignmentX\_**Moss**.zip [a compressed file containing only

the following]

*contains .java file which includes the driver (javafx application),*

*enumerated*

*class, data element, data manager and Junit Test (.java) files – NO*

*FOLDERS!!*

File1.java (example)

File2.java (example)

**Assignment 4 Grade Sheet – Spring 2016**

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Blackboard Date/Time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **DOCUMENTATION 20 points**

# CheckList for Assignment 3 is included and completed 1 pt \_\_\_\_\_

# Class documentation - Javadoc for all user created classes 4 pt \_\_\_\_\_

Class Description, @author

Method description, @param, @return

Test Cases 6 pt \_\_\_\_\_

JUnit Test Class

Implement STUDENT test for ConcordanceDataManagertTest

Create test for ConcordanceDataStructureTest

UML Diagram 4 pt \_\_\_\_\_

Lessons Learned 5 pt \_\_\_\_\_

In 3+ paragraphs, highlight your lessons learned and learning experience from working on this project. How did you do? What have you learned? What did you struggle with? How will you approach your next project differently?

## **PROGRAMMING 80 points**

Internal class documentation (Javadoc) 5 pt \_\_\_\_\_

Description of what class does

Author’s Name

Methods commented properly

# Description

# @param and @return

Compiles and Runs without runtime errors 5 pt \_\_\_\_\_

Accuracy

Public tests – Those I gave you 6 pt \_\_\_\_

Your Junit tests – STUDENT tests for 4 pt \_\_\_\_\_

ConcordanceDataManagerTest

and ConcordanceDataStructureTest

Private tests 10 pt \_\_\_\_

Program Detail

1. Data Element Class – ConcordanceDataElement 5 pt \_\_\_\_\_
   * Defines hashCode
   * Add a line number to a word
   * Follows Javadoc
2. Data Manager Class – ConcordanceDataManager 15 pt \_\_\_\_\_

* Implements ConcordanceDataManagerInterface
* Reads from a file/Writes to a file

1. Data Structure Class – ConcordanceDataStructure 20 pt \_\_\_\_\_
   * Implements ConcordanceDataStructureInterface
   * Uses an array of linked list as the primary data structure

to create concordance

1. Create a GUI driver 10 pt \_\_\_\_\_
   * User must enter input file and output file before creating a

Concordance file.

* + Use FileChooser to select the input and output files.
  + File validation:
    1. Use exception handling (try/catch) for the validity of the files.
  + Display concordance for text in text area , use scroll bar

**Program user interface**

* + Clear to user how data is to be entered
  + Output is easy to understand

Total 100 pt \_\_\_\_\_

Comments: