Tutorial 9

Objectives

Practice with Exception Handling and File I/O.

Attendance Quiz

Please log on to cuLearn using one of the computers in the tutorial room and complete the attendance quiz. You can only access the quiz if you log in using one of the computers in the lab. You cannot use a laptop for this. This is a time limited quiz. Be sure to do this as soon as you arrive.

At the end of the tutorial a TA will assign you a grade, call it G, which is 0, 1 or 2, depending on the progress you make during the tutorial. If you spend your time reading mail and chatting with your friends you will receive 0. If you have completed the attendance quiz on time then G will be your tutorial grade. If you have not completed the attendance quiz on time, then your tutorial grade will be max(0, G - 1/2). Each tutorial grade will therefore be one of 0, 0.5, 1.5 or 2.

In order to receive full marks for this tutorial, you must make significant progress in part 2. If you intend to leave in the first hour of the tutorial then you are expected to complete part 2 and part 3.

---

0) Make a directory called **comp1406t9**. Download all the tutorial files to this directly. Run the **javadoc** program to generate the API for the classes.

1) The **ReadFile** class opens and reads a text file. The **WriteFile** class opens and writes to a file. Compile and run these programs.

There are several ways to read/write text files. The examples shown here are just one way of achieving this.

Look at the API for the **BufferedReader** class. The **readline()** method is a simple way to read the text file line by line. It returns **null** when the end of the file has been reached.

https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html

Look at the API for the **PrintWriter** class. Notice that we can print to a file just like we print to the terminal. There are overloaded **print()** and **println()** methods that print with or without adding a newline.

https://docs.oracle.com/javase/8/docs/api/java/io/PrintWriter.html

2) Using the ReadFile and WriteFile classes as a starting point, write a program that reads all the contents of a file (the provided text.txt, for example) and finds the 10 most frequently used words.

Your program will read every line in the input file and record each *word* in the line. A word is a sequence of consecutive non-whitespace characters.

Note that the **split()** method in the **String** class will break up a string into words.

```
String s = ...
String[] words = s.split("\\s+");
```

Here, the input **"\\s+"** tell split() to split the string s by whitespace characters.

Use a dictionary (**HashMap**) to store the words. The keys should be your words and the values should be the frequency count. Each time you see a word that you have already seen, you will need to update the value for that work (that key).

How do you use a HashMap? The API gives you lots of information!

https://docs.oracle.com/javase/8/docs/api/java/util/Hashtable.html

Once you have built your dictionary, you will need to extract all the key value pairs. You can, for example, use the **keySet()** method to obtain a set (use a **HashSet**) of all the keys in the dictionary. You can then use this set to build list (**ArrayList**) of key-value pairs. You can use the helper class **KeyValue** for this if you wish. You can then sort the list using **Collections.sort()** since KeyValue implements comparable.

Once you have sorted the list, you write the last 10 (the 10 with the highest frequency) to a file called "best10.txt".

Note: Java provided a **Map.Entry** interface to help do this same thing (in place of the KeyValue class). Feel free to use this other approach for this problem.

3) Notice that "cat.", "cat,", "cat!" and "cat" will all appear as different words in your dictionary instead of one "cat" (which should be the case). Modify your code from the previous problem to strip trailing punctuation from all the words before you add them to the dictionary. You should remove things like ",", ";", "!", """, "?", etc.

4) There are situations where more than 10 words should be saved. For example, consider a file with 9 unique words that are the 9 most frequent and then there is 5 words with the 10th largest frequency. All 5 words should be included in the saved file. Modify your code so that all words with the 10 highest frequencies will be saved to the output file.