Assignment 1 Word Frequency

Due Date: Friday Jan 26, before 3:59pm
Weight: 6%
No lates will be accepted. This assignment should be completed individually.

Description

In this assignment you will create a program that reads a text file and outputs statistics on the frequency of words in the file.

It will use Java standard data structures, `ArrayList`, to perform some of the things that we will be doing for ourselves later on like linked lists, searching, and sorting.

It will also introduce you to the world of running your program from the command line and using input/output redirection.

The finished program will be run like this:

```
cat input.txt | java -jar A1.jar > output.txt
```

`input.txt` is any text file containing English text. `A1` will read the standard input, one word at a time. It will maintain a list of words and how many times that word has occurred in the text.

For each word read it will:

1. remove any trailing or leading blanks
2. convert the word to lower case
3. remove any punctuation or digits; e.g. '2day' would become 'day'; 'don't' would become 'dont'; 'end.' would become 'end'; '9,234' would be skipped
4. if the word is on the list of stop words (see below), do nothing else with it
5. if the word is not a stop word:
   (a) Check your list of words
   (b) If the word is already on the list, add to the count of how many times it has occurred
   (c) If it is not already on the list, add it (and set the count to one, obviously)

Once all of the input has been read, `A1` will print:

- The total number of words in the file. This total should not include words that are all digits or punctuation.
- The total number of stop words.
- The number of unique words in the file.
- The 10 most common words ordered from most to least.
- The 10 least common words oredered from least to most.
- All words ordered alphbetically.

The output from your program must be *Exactly* as shown in the examples.

Implementation Details

There are always a number of ways any problem can be solved. For this assignment I will be very prescriptive.

- Create a class called `A1`. It will have the `main()` method and do the bulk of the processing. Be sure to set it up to instantiate an `A1` object.
- Input is all from standard input. Create a `Scanner` object and use that for all input:

  ```
  Scanner inp = new Scanner(System.in);
  ```

  All output should be to standard output. Use `System.out.print();` and `System.out.println();` for all output.
- Create a `Word` class to represent a word. It should include code to maintain a count of how many times the word has occurred.

  The word class should implement `Comparable` and override the `.equals` method.
- Your `A1` class should contain an `ArrayList` of `Word` objects.
- To find the top ten most frequent or least frequent words I want you to use two different `Comparator`s and re-sort your `ArrayList`, using `Collections.sort()` then print the first 10 (if there are 10) items.

  Remember that a `Comparator` must provide a *total ordering* for the objects. That includes some clearly defined behaviour in case of a tie. For example, if there are two words, 'foo' and 'bar' and they both occurred 10 times, which should come first in the list? For this application the secondary ordering should be alphabetical, (a-z), by word.

- Testing and example files. There are four sample input and output pairs named `inp1.txt`, `out1.txt` etc. Use these to determine if your program is running correctly. You should use the `diff` command to compare your output to the samples.

  ```
  diff (cat sampleout.txt) (cat myout.txt)
  ```

  If `diff` gives no output, your program is working correctly.

  You should also devise a test plan and create a series of test files to fully exercise your program. For example, what about an empty file? or one with only some punctuation in it? Or a single word repeated 100 times? or?

  I will be evaluating the program against files you have not seen yet that will be meant to *test* your code.

- Stop words. Stop words are common, short or otherwise uninteresting words that are usually skipped in this kind of analysis. Here is the list for you to use:

  ```
  a, about, all, am, an, and, any, are, as, at, be,
  been, but, by, can, cannot, could, did, do, does,
  else, for, from, get, got, had, has, have, he, her,
  hers, him, his, how, i, if, in, into, is, it, its, like,
  more, me, my, no, now, not, of, on, one, or, our, out,
  said, say, says, she, so, some, than, that, the, their,
  them, then, there, these, they, this, to, too, us, upon,
  was, we, were, what, with, when, where, which, while, who,
  whom, why, will, you, your
  ```

- This is a fairly short assignment. You really only need two classes. Focus on writing compact, efficient code.

Documentation and Coding Standards

Your program should follow all of the coding rules and guidelines outlined in the document provided. In particular, include `Javadoc` style comments for all classes and substantial methods.

What to Hand In

Hand in a single file, `A1.jar`.

Submit this to the Blackboard drop box provided. The jar should be executable and contain:

1. All of your `.class` and `.java` files.
2. A class called `A1` which has a `main()` method.

Grading

A detailed grading rubric will be provided.

I will run your program with the command line given above on three text files and compare your output to the specifications.

Outcomes

Once you have completed this assignment you will have:

- Used standard Java data structures to solve a problem;
- Used polymorphism;
- Used the Java Comparable interface;
- Implemented a Comparator object;
- Used standard input and standard output re-direction.