Book Analysis Program
Focus: Using `String` and `ArrayList` Classes, Sorting, Searching, and Calculating Basic Statistics

Name _____ **Total** _____/105

Date _____ **Period** _____ **QA Name** _____

`Project`

___/ 1 Embeds the author's unique name in the directory/project name

`Word` Class

___ Includes comments with name, date, and summary using Javadoc, indicate other's help, if appropriate

___ Includes two private instance variables whose names are self-documenting

___ Has a constructor with at least a `String` parameter that sets an instance variable to the parameter and sets frequency to 1

___ Gets the text (`String`) of the `word` object

___ Gets frequency of the `word` object

___ Adds to frequency (either with a method such as `addOne` or `setFrequency`)

___ Includes method `compareTo(Object obj)` that compares two words' text

___ Compares frequencies that has an `Object` parameter

- Document the return to indicate what a positive, negative and 0 return value means

___ Includes a method `toString`

- Returns a `String`
- Uses `String.format` method

___ CheckStyle – correctly includes Javadocs for all methods and constructors, including:

- Summary
- `@param`, one `@param` per parameter
- `@return`, if appropriate
- `@precondition` and/or `@postcondition`, if appropriate

`WordAnalysis` Class

___ Includes comments with name, date, and summary using Javadoc, plus help, if appropriate

___ Includes a private `ArrayList` of `Word` objects as an instance variable

___ Has a constructor, accepting the file's name as a parameter

___ Reads words from a file, accepting the file's name as a parameter

___ Strips punctuation, excluding apostrophes, which are part of the word

___ Sorts the `words` list lexicographically ascending

- Loses all the points for not using a recursive merge sort
- Labels the base case
- Uses `Word`'s `compareTo` method

___ Sorts the `words` list by frequency descending

- Loses all the points for not using a recursive merge sort
- Labels the base case
- Uses `Word`'s compare frequency method

___ Searches for a word

- Loses all the points for not using a binary search.
- Labels the two base cases

Book Analysis Program
Focus: Using `String` and `ArrayList` Classes, Sorting, Searching, and Calculating Basic Statistics

___ Outputs a menu
- Loses points if it is difficult to understand
- Loses points if any grammatical mistakes exist

___ Interacts with the user
- Labels the `boolean` return correctly. If unsure how to do this, see the Harker Style Guide.

___ Contains a `main` that is short: it mainly reads the file and loops to interact with user

Menu allows these chores to be tested

___ Returns the total number of words in the file (QA person writes the number here _____)

___ Returns the number of different words (QA person writes the number here _____)

___ Sorts <u>all</u> of the words by frequency in a descending order but does not print

___ Sorts <u>all</u> of the words in lexicographical order in an ascending order but does not print

___ Outputs <u>all</u> of the words by frequency in a descending order

___ Outputs <u>all</u> of the words in lexicographical order in an ascending order

___ Outputs a list of the top "number" of words
- Asks user for a number and outputs that number of words
- Note: the top words change to correspond to the way the words list is sorted (lexicographically or by frequency)

___ Searches for a given word and outputs the word and its frequency
- Asks user for a word and outputs that word and its frequency

___ Outputs the percentage of the `num` most frequent words compared to the the total number of words in the book
- Asks user for a number and uses that number
- Here is a fact that hopefully illustrates this task. The first 25 most commonly used words make up about 33% of all printed material in English. The top 100 words make up 50%.
  - Originally retrieved at http://www.duboisic.org/EducationWatch/First100words.html. Web site is no longer easily available

___ CheckStyle – correctly includes Javadocs for all methods and constructors, including:
- Summary
- `@param`
  - One `@param` per parameter
  - If the parameter is an index, the comment states if the index is included or excluded.
- `@return` if appropriate
- `@precondition` and/or `@postcondition`, if appropriate

Additional features – The author may have added more features. Author is to list them below and have them verified by the QA person:

Book Analysis Program
Focus: Using `String` and `ArrayList` Classes, Sorting, Searching, and Calculating Basic Statistics

Name _____

Date _____ **Period** _____ **QA Name** _____

Moby Dick Testing Check Off Sheet

Search for the following words and their frequencies*:

- ___ moby 79
- ___ whale 955
- ___ the 14321
- ___ tale 1
- ___ tail 76
- ___ he 1748
- ___ she 116
- ___ a 4628
- ___ an 589
- ___ in 4143
- ___ ishmael 18
- ___ harpoon 70
- ___ death 77
- ___ computer – outputs that the word is not in the file

___ Top 10 most used words*

the	14321
of	6578
and	6362
a	4628
to	4577
in	4143
that	2940
his	2520
it	2368
i	1943

___ Top 10 words when sorted lexicographically*

a	4628
a'lee	1
a'low	1
a'mosti	1
a'ready	2
a'shiverout	1
a'top	2
a-begging	2
a-calling	1
a-going	1

Before testing the following, make sure that the file is sorted by frequency.

___ The top 25 words make up 34% of the total number of words in Moby Dick.
Total number of the 25 most frequent words in Moby Dick is 71938*.

___ The top 50 words make up 42% of the total number of words in Moby Dick.
Total number of the 50 most frequent words in Moby Dick is 89561*.

___ The top 100 words make up 51% of the total number of words in Moby Dick.
Total number of the 100 most frequent words in Moby Dick is 107489*.

___ Number of unique words in Moby Dick is 20015 [Close to 20,000 is fine*]
Total number of words in Moby Dick is 211806 [Close to 200,000 is fine*]

*Note that numbers should be close but do not have to be identical. Try to be within 3 to 6%. The numbers reflect how apostrophes, hyphens and dashes are handled.

