ESS 201 Programming II (Java)
Term 1, 2019-20
Collections: Exercise

This assignment is meant as an exercise in using Collections. This need not be submitted, and will not be graded.

The purpose of this exercise is to see how much of certain common and useful computations can be accomplished using Collections and operations available on them, so that you, the programmer, do not have to work on detailed implementation and manipulation issues. You are strongly encouraged to use existing methods in the String class and in the Collections framework. Try to identify the best Collections type to use and invoke the appropriate methods of those objects

Given a set of English sentences, we would like to analyze the text to look for different patterns of usage of words - what words are used frequently, how many unique words are used. whether words from a specified set are used, etc.

The input is a text string as well as a list of “reserved” words. Your program should output the following:

1. The unique words in the input:
    a. In the order they **appeared** in the input string
    b. In alphabetical order (normal **lexicographic** order)
    c. In order of increasing word **size**. That is all 1 letter words first, then 2 letter words, then 3 letter words etc. If there are multiple words with the same length, then these should be listed in lexicographic order.
    d. In order of increasing frequency of occurrence. If two words have the same frequency count, then they appear in the same order as the initial input
2. The count of the words in the input text that **start** with each letter of the alphabet
3. The usage of the **reserved** words in the input
    a. in the order they **appeared** in the input
    b. In order of **decreasing frequency** of occurrence in the input

The program should consider the following:

- You can assume that the input text does now have any punctuations (edit the sample text given below to remove punctuations). Of course, you could build that into your program.
- All comparisons and output should consider only the lower-case versions of the text.

Example:

If the input text is:

A collection — sometimes called a container — is simply an object that groups multiple elements into a single unit.. A collections framework helps manipulate collections, creating sets of elements etc.

and the reserved words are

groups collections elements

The output should be:

1a: a collection sometimes called container is simply an object that groups multiple elements into single unit collections framework helps manipulate creating sets of etc

1b: <above words in sorted order>

1c: a an is of etc into that unit ...

1d: collection sometimes called container is simply an object that groups multiple elements into single unit framework helps manipulate creating sets of etc collections a

2a:
a 5
c 5
e 1
f 1
g 2
i 2
<and so on>

3a: groups elements collections
3b: elements collections groups

Notes (read this after you have spent some time on thinking through the problem!)

You are strongly encouraged to use existing methods in the String class and in the Collections framework. Part of this is to identify the best Collections type to use and invoking the appropriate methods of those objects. In fact, with the use of the appropriate bulk methods of these classes, you can avoid having to explicitly iterate through the collections. In case you need to explicitly iterate through a collection, use an Iterator to do so.

Some of the available methods/classes you can look at:

- String class: split, replaceAll, toLowerCase etc.
- Conversion from arrays to lists and vice versa Arrays.asList and toArray for collections
- Set, SortedSet, HashMap, SortedMap, TreeMap, ArrayList, etc. For the Sorted variants, use an appropriate Comparator at construct time. Make sure you create Collections with the correct type of its elements
- Methods on collections like retainAll, removeAll apart from the more common ones, and appropriate use of sort using Comparators
- Initializing one type of Collection from another.

Try to design the code on the following lines:
An **Analyzer** class that is provided the input string and a list of reserved words, and can be queried for each of the outputs needed above. All these interface methods should return an **ArrayList** (of the right type of element). Thus, any processing as well as use of other Collections should be hidden within Analyzer

The main/driver method creates an instance of Analyzer, reads in the input, initializes the Analyzer with this data, and queries it for each of the questions above, and prints out the elements of the list received. To keep the design clean, no printing should be done from the Analyzer class, and no processing/computation should be done in the main class.