ESS 201 Programming in Java
T1 2020-21
Lab 6
16 Sept 2020

Part A. (To be worked on during the lab class). Part B, the mini-project/assignment, will be published later in the week.

The main purpose of this exercise is to get familiar with the use of Collections and operations available on them, so that you, the programmer, do not have to work on detailed manipulation issues!.In all the following, we need not be concerned about the run-time efficiency of the approach, but focus on "re-use". So, the less code, the better. Of course, where possible, choose the best implementation class you can find.

A good resource for Collections is https://docs.oracle.com/javase/tutorial/collections/TOC.html
For a list of the implementation classes available, see
https://docs.oracle.com/javase/tutorial/collections/implementations/index.html
For a list of useful algorithms, see
https://docs.oracle.com/javase/tutorial/collections/algorithms/index.html

For most of the cases below, you should be able to find an appropriate implementation class and/or an algorithm that will get you the result needed without having to write too much code.

I. Assume you have a String containing multiple words. Implement methods or program fragments that would output each of the following:

1. The number of words in the input string
2. The words in the input that would be first and last word in alphabetical order
   [Hint: see how you can use max and min from Collections]
3. The words in the input list in random order
   [Hint: look for methods in Collections that would help with this]
4. The unique words in the input:
   a. In any order
      [Hint: use a (concrete derived class of) Set, constructed with the input words as parameter to the constructor]
   b. In alphabetical order (normal **lexicographic** order)
      [Hint: Check the default Collections.sort]
   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.
      [Hint: Implement a Comparator that uses the above rule]
   d. In the order they **appeared** in the input string
   e. In order of increasing frequency of occurrence. If two words have the same frequency count, then they appear in appear in alphabetical order

[Hint: This may require Map and MapIterators, which we have not yet discussed in class]

You can assume there will be exactly one blank (whitespace) between words
All comparisons and output should consider only the lower-case versions of the text.

To the extent possible, 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.

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

- String class: split, toLowerCase etc.
- Set, SortedSet, HashMap, ArrayList, LinkedHashSet, Initializing one type of Collection from another.
- Appropriate use of Comparators

II. Implement a class LoginManager that manages logins for a website. The class has a database (list/table) of (username, password) pairs. It provides the following methods:

1. boolean addUser(String name, String password) - adds a user to its database if that user does not already exist, and returns true. Else returns false
2. boolean checkLogin(String name, String password) - checks if this credentials pair exists in the database and returns true if successful
3. Lists out all usernames in the database

Explore implementing this using ArrayList, HashMap and an appropriate implementation of Set.

As a side note: typically, we never store passwords in plain text in a database/collection. Passwords are typically encrypted and stored. How would the above change if encryption were to be used? You can assume the availability of a static method *encrypt* that, given a String, returns a unique encrypted string.