CS 251 Intermediate Programming
Lab 4: Anagram Solver

Brooke Chenoweth

Spring 2014

This assignment will give you some experience using Java Collections and a nested class. You will also be introduced to a technique known as recursive backtracking.

Problem Description

You will write a class that uses a dictionary to find all combinations of words that have the same letters as a given phrase. In other words, you are going to find anagrams.

What do you need to do?

1. Write an `AnagramSolver` class.
   - The constructor should take a `List` of `String` object and use thegiven list as its dictionary. It should not modify the list.
   - `void print(String phrase, int maxWords)` – Prints to `System.out` all combinations of words from its dictionary that are anagrams of the phrase and that include at most `maxWords` words (or unlimited number of words if `maxWords` is 0). Throws an `IllegalArgumentException` if `maxWords` is less than 0. (This is an *unchecked* exception type, so you won't have to say that your method throws it. See the Java API for more details.)
   - You will need one or more additional methods to find all anagrams of the input phrase.
2. Inside the `AnagramSolver` class, write a public static nested `LetterInventory` class. This class will keep track an inventory of letters of the alphabet. It will ignore the case of the letters and ignore anything that is not a letter (punctuation, digits, whitespace, etc.).
   - A constructor should take a `String` and construct an inventory of the alphabetic letters in the given string, ignoring the case of letters and ignoring any non-alphabetic characters.

- `boolean isEmpty()` – returns true if this inventory is empty.
- `String toString()` – (This overrides the method in `Object`.) Returns a String representation of the inventory with the letters all in lowercase and in sorted order and surrounded by square brackets. The number of occurrences of each letter should match its count in the inventory. For example, an inventory of 4 a's, 1 b, 1 l and 1 m would be represented as "`[aaaablm]`".
- `LetterInventory subtract(LetterInventory other)` – Constructs and returns a new `LetterInventory` object that represents the result of subtracting the other inventory from this inventory (i.e., subtracting the counts in the other inventory from this objects counts). If any resulting count would be negative, your method should return null.
- You may need some additional methods to complete this program. (add? size? another constructor?)

3. Have fun generating anagrams!

   - I have provided you with an `Anagrams` class to test your `AnagramSolver` class.
   - `Anagrams` takes a single command line argument to specify the name of a dictionary file. (If no file name is given, it assumes "`words.txt`". You may want to create a small dictionary file with this name for easy testing purposes.) It will read the words in this file, place them in a list, and use that list to construct an `AnagramSolver`.
   - After constructing the `AnagramSolver`, the user will be prompted for a phrases and a max word count. The `AnagramSolver` will generate and print all the anagrams for the given phrase with up to the requested number of words. This will repeat until the user wants to stop.
   - I strongly suggest you use a small dictionary file and/or restrict the number of words requested for a given phrase. Searching a large dictionary for all possible anagrams can take a *long* time.

What is this? I don't even know where to start!

Take a deep breath and start by writing the `LetterInventory` class. We'll go over the recursive backtracking algorithm that you can use to find the anagrams in class.

Turning in your assignment

Once you are done with your assignment, use UNM Learn to turn in `AnagramSolver.java`.