Assignment 1: Frequency of Ingredients using ArrayList

Due Date: Friday September 28, before midnight

Weight: 7%

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 which represents a recipe book and outputs statistics on the frequency of ingredients in the file.

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

The finished program will be run like this:

cat recipebook#.txt | java -jar A1.jar > output.txt

recipebook#.txt is a text file containing a list of recipes. The first line of the file is *******. After this line, each recipe will start with a title line, followed by a line with three dashes, followed by a list of ingredients. Each ingredient will be on a separate line of the text file. After the ingredients are finished for a recipe, a line of three dashes will occur, then recipe instructions, then a line of three stars to end the recipe and separate it from the next recipe (if there is one).

For example, a short ***recipebook#.txt*** could look like this:

```
***
Apple Pie
---
1 Apple
1 Whole Wheat Crust
---
Cook apple in crust at 350°
***
Bean Burgers
---
1 can black beans
1 egg
1 cup breadcrumbs
----
Combine all ingredients.
Flatten into disks and cook.
***
```

A1 will maintain an ArrayList called **foodList** of Foods in the ingredient list and the number of recipes in which that word has occurred (its frequency). First it will add the common foods at the end of this file to **foodList**, each with frequency 0. The frequency will be incremented if they are seen again in a recipe. Then foodList will be sorted from the longest to the shortest length words. Finally **A1** will read ***recipebook#.txt*** one line at a time until it reaches the end of the file.

Assume that the string that has just been read be called **nextLine**.

1. If **nextLine** is "---", "***", the title line, or the instructions, nothing will be done
2. Otherwise, **nextLine** represents an ingredient. If the name of any Food currently in foodList is a substring (case insensitive) of **nextLine**, then its frequency is incremented. Only one Food frequency can be incremented per line. (i.e. if an ingredient was 1 banana squash, you would not increment both banana and squash and banana squash). The one that is seen first in the list will be incremented.
3. If 1 and 2 are not true, then a new Food will be created, with frequency equal to 1, and name equal to the last word in the string. (i.e. the word after the last space)
    - If the word ends in "es", the "es" will be removed.
    - Any spaces at the end or beginning of the word are removed
    - If the word ends in a single "s", then the single "s" will be removed. For example, "a handful of horseradishes" will be stored with the name "horseradish", and "two eggplants" will be stored with the name "eggplant". Unfortunately, "foie gras" will be stored as "gra", should it be in the list of ingredients.
    - The list should be sorted after every addition of a new ingredient, from longest to shortest, so that the longest ingredient name will always be chosen first.

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

- The total number of different ingredients in the **recipebook#.txt** file (not including foods with frequency 0).
- The 10 most common foods ordered from most to least. (If there are less than 10 foods that were used as ingredients, then only these foods will be on the list). If there is a tie (for example 10 potato and 10 apple), then they will be ordered alphabetically.
- All foods used in the recipe list, in alphabetical order.

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 ***Food*** class to represent a food. It should include code to maintain a count of how many times the food has occurred and a string to represent the name of the food.
- The word class should implement ***Comparable*** and override the ***.equals*** method.
- Your ***A1*** class should contain an **ArrayList** of **Food** objects named **foodList**, added from the list at the end of this file, and from the **recipebook#.txt** file.
- To find the top ten most frequent or least frequent words the **Comparator** interface should be used 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.

Testing and example files.

There are four sample input and output pairs named **recipebook1.txt**, **recipebook2.txt** etc and **output1.txt and output2.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.

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

For this program 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:

- All of your **.class** and **.java** files.
- 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.

Some Useful String Methods

trim

indexOf

lastIndexOf

substring

Outcomes

Once you have completed this assignment you will have:

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

Basic Food List

baking powder
baking soda
cheese
broth
tomato paste
tomato sauce
tomato
flour
egg
garlic
cheese
rice
onion
salt
pepper
vinegar
baking powder
carrot
sweet potato
potato
cream
milk
bean
green bean
beef
chicken
cumin
basil
oregano
oil
fish