CSCI 2110 Data Structures and Algorithms
Laboratory No. 5
Release Date: Oct 11th
Due Date: Oct 18th
23h55 (five minutes to midnight)

Unordered Lists II

Introduction

In this lab you will have extra practice with reading from files and performing basic tasks on a list of data.

Marking Scheme

Each exercise carries 10 points.
Working code, Outputs included, Efficient, Good basic comments included: 10/10
No comments or poor commenting: subtract one point
Unnecessarily inefficient: subtract one point
No outputs and/or not all test cases covered: subtract up to two points
Code not working: subtract up to six points depending upon how many methods are incorrect.

Your code must compile. Failure to compile may result in a score of 0.

Your final score will be scaled down to a value out of 10.

Error checking: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.

Submission: All submissions are through Brightspace. Log on dal.ca/brightspace using your Dal NetId.

What to submit:
Submit one ZIP file containing all source code (files with .java suffixes) and a text documents containing sample outputs. For each exercise you will minimally have to submit a demo Class with a main method to be tested and your test code, sample outputs.

Your final submission should include the following files: *AverageAnalyzer.java, AverageAnalyzerTester.java, Exercise1Input.txt, Exercise1Out.txt, SetAnalyzer.java, SetAnalyzerTester.java, Exercise2List1.txt, Exercise2List2.txt, Exercise2Out.txt.*

You MUST SUBMIT .java files that are readable by your TAs. If you submit files that are unreadable such as .class, you will lose points. Please additionally comment out package specifiers. Your code must compile. Failure to compile may result in a mark of 0.

In both exercises, you are free to make additional helper methods as long as you stick to the requirements and functionality of this document.

Exercise1

For this exercise, you will complete the code to *AverageAnalyzer.java*. The purpose of this class is to calculate basic statistics on a list of values.

Complete the following methods and functionality in *AverageAnalyzer.java*:

- AverageAnalyzer(ArrayList<Integer> values): constructor which sets up the values list.
- mean():double method, which calculates the mean from the list of values
- mode(): int method, which calculates the mode from the list of values
  - You are guaranteed that the list of values will have a mode and will be in the range of [0,100]

<u>You may use ArrayList methods, but do not use ArrayList.sort().</u>

Create the class *AverageAnalyzerTester.java* which contains a main method and has the following functionality:

- Asks the user to enter a file name
- Populates an ArrayList of Integers with the file data
- Create an AverageAnalyzer object
  - Pass it the list of data from the file
  - Run the mean() and mode() methods
  - Print the results

Input/output

Your program should accept input in the following format:

- Prompt the user for the file name

Your program should provide output in the following format:

- Print the mean and mode
- No need to decimal format

Student Generated Test Cases

Run the file provided: *Exercise1Input.txt*
Save your results in *Exercise1Out.txt*

Sample inputs and outputs:

Exercise1.in Enter file name: [*file path here*]	*Exercise1.out* Mean: [*some value*] Mode: [*some value*]

Exercise2

In this exercise you will complete the code to *SetAnalyzer.java*. This class computes the set intersection and union of two <u>generic</u> sets of data. You will run your code against sets of integers and strings.

For example, given the following sets: Set1={A,B,C,D,E}, Set2={E,D,F,Z}

- Intersection of Set1 and Set2 = {D,E}
- Union of Set1 and Set2 = {A,B,C,D,E,F,Z}
  - No repeats in the union

Complete the following methods and functionality in *SetAnalyzer.java*:

- setAnalyzer(): empty constructor
- intersection(ArrayList<T> list1, ArrayList<T> list2):ArrayList<T> method which accepts two generic arraylists of type T, computes their intersection set, and returns a new arraylist.
  - Do not alter the original arraylists
- union(ArrayList<T> list1, ArrayList<T> list2):ArrayList<T> method which accepts two generic arraylists of type T, computes their union set, and returns a new arraylist containing the union.
  - Do not alter the original arraylists

You may use ArrayList methods, but do not use ArrayList.sort().

Create the class *SetAnalyerTester.java* which contains a main method and has the following functionality:

- Ask the user if they will be analyzing integers or Strings
- Asks the user to enter a file name
- Asks the user to enter another file name
- Populates the two ArrayLists with input data from the two files
- Create a SetAnalyzer object
  - Pass it the lists of data from the files
  - Run the intersection() and union() methods
  - Print the results

Input/output

Your program should accept input in the following format:

- Prompt the user to enter two files

Your program should provide output in the following format:

- Print the union and intersection of the two files' data

*Note: You may assume there are no repeated values per input file

Student Generated Test Cases

Test your code on the following files:

- *Exercise2List1.txt* and *Exercise2List2.txt*
- *Exercise2List3.txt* and *Exercise2List4.txt*

Run your code with the pairs above.
Save your results in *Exercise2Out.txt*

Sample inputs and outputs:

Exercise1.in Will you be analyzing Strings or Integers? [*some choice "strings" or "integers"*] Enter file name:[*some file*] Enter the second file name:[*some file*]	*Exercise1.out* Intersection: [*print the intersection here*] Union:

	[*print the union here*]

References:
https://en.wikipedia.org/wiki/List_of_dog_breeds