List Exercises! (Please note how exercises is spelled…)

Welcome to Lists, in order to prove your chops (and before we begin a fun-er project), please work on the following 12 List exercises.

A couple of requirements:

- Make all your methods **static**
- You **MUST** only use variables of type List, i.e. List<String> listName = new ArrayList<String>();
    - You may use whatever data structure that implement the List interface, though I recommend ArrayLists.
- You **MUST** write two jUnit tests for every method listed here (except the first one)
- Please make sure you are adhering to documentation guidelines 1 & 2

Have fun!

navarroExercise

A method made by my old ADSA teacher, although I can't say I remember writing this one. Write a method called void navarroExercise, which takes as a parameter a String filename (including a .txt in the string) and does the following:

1. Read all the integers in the file (one per line) and insert each number directly into an ArrayList pointed to by a List variable..
2. Once inserted, print the number of numbers in the list.
3. Then call a method to print the list in a horizontal row using a for-each loop. (If you want to do this in the same method, that's fine.)
4. Then call a method to traverse the loop again, this time using a for loop, and remove all odd integers from the list. (If you want to do this in the same method, that's fine.)
5. Once again, print the number of numbers in the list and call a method to print the list in a horizontal row using an iterator.
6. Lastly, print out only the first and last integers in the list. Label each one.

Note: DON'T write a JUnit test for navarroExercise.

maxLength

Write a method maxLength that takes a List of Strings as a parameter and that returns the length of the longest string in the list. If your method is passed an empty list, it should return 0.

swapPairs

Write a method swapPairs that switches the order of values in a List of Strings (passed via parameter) in a pairwise fashion and returns the result. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on. For example, if the list initially stores these values: {"four", "score", "and", "seven", "years", "ago"} your method should switch the first pair, "four", "score", the second pair, "and", "seven", and the third pair, "years", "ago", to yield this list:{"score", "four", "seven", "and", "ago", "years"}

If there are an odd number of values in the list, the final element is not moved. For example, if the original list had been: {"to", "be", "or", "not", "to", "be", "hamlet"} It would again switch pairs of values, but the final value,"hamlet" would not be moved, yielding this list: {"be", "to", "not", "or", "be", "to", "hamlet"}

removeEvenLength

Write a method removeEvenLength that takes a List of Strings as a parameter and removes all of the strings of even length from the list.

doubleList

Write a method doubleList that takes a List of Strings as a parameter and that replaces every string with two of that string. For example, if the list stores the values {"how", "are", "you?"} before the method is called, it should store the values {"how", "how", "are", "are", "you?", "you?"} after the method finishes executing.

minToFront

Write a method minToFront that takes a List of ints as a parameter and that moves the minimum value in the list to the front, otherwise preserving the order of the elements. For example, if a variable called list stores the following values: {3, 8, 92, 4, 2, 17, 9, 2} and you make this call: minToFront(list); it should store the following values after the call: {2, 2, 3, 8, 92, 4, 17, 9} You may assume that the list stores at least one value.

Remember to consider the boxing necessary for this task.

removeDuplicates

Write a method removeDuplicates that takes as a parameter a sorted List of Strings and that eliminates any duplicates from the list. For example, suppose that a variable called list contains the following values: {"be", "be", "is", "not", "or", "question", "that", "the", "to", "to", "to"} After calling removeDuplicates(list); the list should store the following values: {"be", "is", "not", "or", "question", "that", "the", "to"}

Because the values will be sorted, all of the duplicates will be grouped together.

stutter

Write a method stutter, similar to *doubleList*, that takes a List of Strings and an integer k as parameters and that replaces every string with k copies of that string. For example, if the list stores the values ["how", "are", "you?"] before the method is called and k is 4, it should store the values ["how", "how", "how", "how", "are", "are", "are", "are", "you?", "you?", "you?", "you?"] after the method finishes executing. If k is 0 or negative, the list should be empty after the call.

markLength4

Write a method markLength4 that takes a List of Strings as a parameter and that places a string of four asterisks "$&*#" in front of every string of length 4.

For example, suppose that a variable called list contains the following values: {“Fall”, “is”, “the”, “best”, “season”, “of”, “the”, “year”} And you make the following call: markLength4(list); then list should store the following values after the call: {“$&*#”, “Fall”, “is”, “the”, “$&*#”, "best", “season”, “of”, “the”, “$&*#”, “year”}

Notice that you leave the original strings in the list, “Fall”, ”best” and “year”, just add the string “$&*#” before it.

removeShorterStrings

Write a method removeShorterStrings that takes a List of Strings as a parameter and that removes from each successive pair of values the shorter string in the pair. For example, suppose that an ArrayList called list contains the following values: {"four", "score", "and", "seven", "years", "ago"} In the first pair, "four"and "score", the shorter string is "four". In the second pair, "and" and "seven", the shorter string is "and". In the third pair, "years" and "ago", the shorter string is "ago". Therefore, the call:removeShorterStrings(list); should remove these shorter strings, leaving the list as follows: "score", "seven", "years". If there is a tie (both strings have the same length), your method should remove the first string in the pair. If there is an odd number of strings in the list, the final value should be kept in the list.

interleave

Write a method called interleave that accepts two Lists of integers a1 and a2 as parameters and inserts the elements of a2 into a1 at alternating indexes. If the lists are of unequal length, the remaining elements of the longer list are left at the end of a1. For example, if a1 stores [10, 20, 30] and a2 stores [4, 5, 6, 7, 8], the call of interleave(a1, a2); should change a1 to store [10, 4, 20, 5, 30, 6, 7, 8]. If a1 had stored [10, 20, 30, 40, 50] and a2 had stored [6, 7, 8], the call of interleave(a1, a2); would change a1 to store [10, 6, 20, 7, 30, 8, 40, 50].

writeYourOwn

Write your own method that does something interesting, please document your instructions for what to code up (like the snippets I have provided for you above).