Lab 7: Oops! I did it again.

Objectives

- Practice with structural recursion.
- Get more experience using testing methods.

Step 0: Preliminary Setup

For this lab, you will be creating several methods for doing things with lists, using recursion. Start by doing the following:

1. Create an IntelliJ project.

2. Add the starter code from the Google drive folder to your src folder:
    a. **Main.java** has some tests that you'll use to test your list methods
    b. **Tester.java** is used by **Main**.
    c. **ListProcessor.java** that has some starting implementations for the methods you'll write.

Step 1: Rewrite minimum-finding methods recursively

In **ListProcessor**, notice two methods **getMin** and **getMinIndex**. Read the javadocs explaining their behavior. Also, **read getMinTests in Main** to see how they are called.

I've provided *non-recursive* implementations for both methods. Your job is to ***write them recursively***, using the approaches we've discussed in class.

Recall that a recursive method has two parts:

1. deal with the base case(s), then
2. deal with the recursive case(s).

For list problems, the recursive case often looks like:

1. split the list into the first element and the rest of the elements.
2. recursively solve for the rest of the elements.
3. combine with the first element we set aside.

So, finding the minimum element in a list is, in the recursive case:

1. split the list into the first element and the rest of the elements.
2. find the minimum element for the rest of the elements.
3. if that minimum is less than the first element we set aside, return it; otherwise return the element we set aside.

So, rewrite getMin and getMinIndex recursively, and make sure they work by running the tests in **Main**. Feel free to add more tests if you'd like.

Step 2: Fix recursive sort

Now, let's look at the **sort** method in **ListProcessor**. It's supposed to sort a given list, but doesn't quite work. The general structure of the algorithm is what we call ***selection sort***, where we *select* the minimum element from the list, swap it into its correct position, and then (recursively) sort the rest of the list.

Note that the version I've provided doesn't work, even though the overall logic is sound. If you look at the tests in **Main** and what gets output, you should see that it **does** swap the smallest element into position, but then doesn't seem to sort the rest of the list.

The reason this doesn't work is because the recursive call

sort(new ArrayList(aList.subList(1, aList.size())))

is sorting a *newly-created* list that contains all elements of **aList** other than the first element. So, we recursively sort *that new list*, without modifying the original **aList** at all.

There is a way to fix this: create a helper method that takes an additional parameter to indicate what part of the list to sort. So, your job is to write this method:

```
/**
 * Sorts the portion of 'aList' from index 'start' to the end of
 * 'aList'. The list is sorted in place.
 *
 * @param aList the list to sort
 * @param start the index of the first element of interest.
 */
private void sortSublist(ArrayList<String> aList, int start)
```

using the general Selection Sort algorithm. This helper method only sorts a part of the list (namely, only those elements with index **start** or higher). Some questions to ask yourself as you write this:

- How do we tell if the part of the list we are trying to sort is empty? We can't just use **aList.size() == 0** like we did before because the part we are sorting is not the entire list.
- In the recursive case, we need to recursively sort the rest of the list. Recall that we are sorting the elements with index **start** or higher. How do we recursively sort all of those elements except the start element?

Once you've written **sortSublist**, write **sort** so it calls **sortSublist** with appropriate parameters to sort the entire list. **Hint**: at what index should we *start* if we want to sort the entire list?

Another Hint: Your **sort** should just be a single call to **sortSublist**.

Make sure it works by running the tests in Main. Feel free to add additional tests.

How to turn in this lab

Before turning in any program in this class, remember this mantra:

> ***Just because it works doesn't mean it's good.***

Part of your grade will also come from things like how understandable and readable your code is. You will also be graded on the neatness, presentation, and style of your program code.

Turn in the lab by uploading your .java files to gradescope by 4pm the day after lab.

Ask for help if you're having problems!