Data Structures and Algorithms **Spring '23 section 1** **Dr. Hristescu**

Lab2

Assigned: Thursday, January 26
Due: 10:00pm Monday, January 30

Problem1: Use the code from the textbook (presentation starting page 226, implementation starting on page 231):

- ListInterface for the List ADT
- ListArrayBased slightly modified textbook implementation of this interface (fixes 2 of the 3 logic errors - causing ArrayIndexOutOfBoundsException)
- ListIndexOutOfBoundsException
- ListException

a. Eliminate all 3 errors in the (modified) textbook code and comment the lines of code appropriately:

- 1 implementation error **Attach comment: //fixes implementation errors**
- 1 memory leak error **Attach comment: //fixes memory leak**
- 1 programming style error that would also prevent you from extending the class and using the inherited code properly (as described below) **Attach comment: //fixes programming style**

After fixing these do not make any more changes to the ListArrayBased.

b. Create a **subclass** of ListArrayBased called **ListArrayBasedPlus** and incorporate the resizing of the array when full [**using the frame outlined in class**] that also has the additional functionality of **reversing** the list.

Make sure that ListArrayBasedPlus has a **toString** method (note that ListArrayBased doesn't have one) that gathers the information starting from the first position to the last position in the list.
Therefore, **ListArrayBasedPlus** should contain the following 4 methods: **add**, **reverse**, **toString** and internal method **resize** (as discussed in class, resize should take no parameters).
Note:

- In the subclass you should **work with the underlying structure directly** (rather than inherited methods)
- For **reverse** write your own code that does the reversing of the collection
- For **add** you should not re-invent the wheel, i.e. you should invoke the superclass add method instead of duplicating the add code in your subclass after ensuring capacity. Use the frame discussed in class.

Test the functionality of your class using a menu-driven test application with the following options:

0. Exit program
1. Insert item into the list
2. Remove item from the list
3. Get item from the list
4. Clear the list
5. Print size and content of the list
6. Reverse the list

A sample run of the program on this input data can be found here.

Problem2: Implement the **ListInterface** using as the underlying structure an **ArrayList** collection instead of an array data structure. Name the class **ListArrayListBased**. Make sure that you only have data fields that are needed and do not introduce any redundancy. Implement the additional functionality in the subclass **ListArrayListBasedPlus**. ArrayList can be found in java.util.

Problem3: Use the internet to find the source code for the ArrayList class. Study how the different methods that are related to the functionality of the ListInterface are implemented and **attach the code for ONLY those specific methods to your lab**.

Problem4: Lab2Conclusions: Summarize in a few sentences what you learned from working on this lab.

Extra credit: (marked clearly_EC and noted in file header)
Implement a more efficient **add** method for the extended method and submit it in a class named **ListArrayBasedPlusMore**. Submit this class in addition to **ListArrayBasedPlus**, not in place of it.

Submit:
Problem1: corrected **ListArrayBased**, **ListArrayBasedPlus**, application program and sample runs
Problem2: **ListArrayListBased**, **ListArrayListBasedPlus** and sample runs.

Problem3: **ArrayList** implementation for the ListInterface methods and related methods
Problem4: Conclusions
If EC is submitted: **ListArrayBasedPlusMore** and sample runs.
Follow the instructions to turn in the lab electronically. Use Lab2 instead of Lab1 for the name of your directory.