IT 306 Spring 2022
Data Structures and Algorithms in Java
Assignment 4

Due: 3/16/2022

Assignment 4

This assignment is designed to introduce you Array-based list and divide and conquer-based sorting.

Description

You will be implementing an array-based list with dynamic resizing. You will also implement a sorting method using the array-based list.

Overview

A List interface is provided, implement the list interface using an array. The interface has defined in a way to only work for integer type. You can use the code directly from the textbook or write your own as long as it behaves the same and has the same public interface. In addition to implementing the interface methods, the data structure's existing behavior should be modified such that the size dynamically increases when needed. One of the interface methods increases capacity, and it should be "automatically" invoked as needed. You can use either doubling strategy or fixed-increment strategy.

Another part of your assignment requires you to implement the sort method of the list using a divide and conquer approach. You should use a randomized Quick sort method.

The following files are provided in the assignment package:

- List.java
- ArrayList.java
- TestArrayList.java

List.java is an indexed-based interface with the following methods:

size(): Returns the number of elements in the list.

isEmpty(): Returns a boolean indicating whether the list is empty.

get(*i*): Returns the element of the list having index *i*; an error condition occurs if *i* is not in range [0, size( ) − 1].

set(*i*, *e*): Replaces the element at index *I* with *e*, and returns the old element that was replaced; an error condition occurs if *i* is not in range [0, size( ) − 1].

add(*i*, *e*): Inserts a new element *e* into the list so that it has index *i*, moving all subsequent elements one index later in the list; an error condition occurs if *i* is not in range [0, size()].

remove(*i*): Removes and returns the element at index *i*, moving all subse- quent elements one index earlier in the list; an error condition occurs if *i* is not in range [0, size( ) − 1].

increaseCapacity(): copies the array into a new array with a size that is greater than its current size, and replaces the original array with the new one. You may choose to implmement either a constant or doubling algorithm. If the array is uncreated, then throws IllegalStateException.

minimize(): copies the array into a new array with a size that exactly matches the number of elements, and replaces the original array with the new one. array is empty, or hasn't been created yet, then throws IllegalStateException.

capacity():Returns the length of the array (not how many elements are in it). If the array has not been created, then throws IllegalStateException.

Sort(): sort the list using quick sort.

ArrayList.java is the implementation class for the interface.

TestArrayList.java provides a basic implementation to test and execute ArrayList.java. If implemented correctly, your code should pass all the testcases and show the proper ascending order of the list.

Extra points (10)

Implement the in-place version of quick sort.

Requirements

Complete implementation of all the methods in the ArrayList.java. You can add your own method for sorting implementation if needed.

Deliverables

You should submit a zip file named YourFullName-Assignment4 containing: • *only* your java source code

- no metafiles
- no .class files
- no unit test files (e.g. Test*.java)
- no package declarations in your source code files!

Note: any unimplemented methods in your classes should remain "stubbed" so that they compile against the interfaces, and run relative to the unit tests. Any programs that do not compile or run, or cause the test programs to fail, will earn zero points.

Stubbed means that the class will respond to the public interface even if the method is not otherwise implemented. For example:

```
public boolean isEmpty() {  return true; }
```

The method above will not work correctly (unless it's truly empty), but it will compile and it will allow unit tests to invoke it.

Grading

Implementation of Array list	50 points
Implementation of Quick Sort	50 points