**Description**

Build three classes that conform to the following interfaces. Use arrays in creating your classes (e.g., do not use the built-in ArrayList class when creating the ArrayList-like interface). Extend the sample driver below to completely test the FIFO nature of a Queue, the LIFO nature of a stack and the arbitrary inserts and removes in an ArrayList-like structure.

Sample Driver: [ArrayBasedDataStructuresDriver.javaPreview the document](https://canvas.uw.edu/courses/1331881/files/58017714/download?wrap=1)

**ArrayList-like Interface**

* void insert(Object, int index):  Add the object at the specified index.
* Object remove(int index):  Remove and return the object at the specified index.  (This should behave like the corresponding method in Java's built-in ArrayList class; see Savitch's description.)
* int size()
* String toString()
* boolean isEmpty()
* int indexOf(Object):  Returns -1 if not found
* boolean equals(Object):  Compare sizes and elements in the data structure.
* Object get(int index):  Returns the object at index specified.

**Stack Interface (LIFO)**

* void push(Object)
* Object pop()
* int size()
* String toString()
* boolean isEmpty()
* boolean equals(Object)

**Queue Interface (FIFO)**

* void enqueue(Object)
* Object dequeue()
* int size()
* String toString()
* boolean isEmpty()
* boolean equals(Object)

**Hints and Notes**

* You should practice building drivers and even more advanced testing harnesses yourself.  The above driver is merely a starting point that you can use to initially exercise your code.  In lecture, we will talk about what a testing harness should do.  For this assignment, you can just display the results of your test to screen.  Once we've gone through exception handling, you will have all the skills needed to write the harness the way I described in class.
* A reminder that Stacks should be LIFO, queues should be FIFO, and ArrayLists can add and remove in arbitrary order.
* The data structures should hold objects of class Object.
* Your code should not depend on absolute path specifications or any other environment-specific specifications.  We will take off 3 points from any submission that has such a specification.  (To those using Eclipse, this is a particularly easy mistake to make because Eclipse will often add in a package statement automatically.)  The grader should not have to change your code to run it on his/her machine.  If you're not sure how to make your submission non-environment-specific, please see me.  The program should **not** ask for user input from the console.
* The toString methods should show the contents of the data structure in a "natural" ordering.  Assuming the method writes data structure elements from left to right, for a stack, this starts from top on down, for a queue, this is from front to back, and for an ArrayList, this is from first index to the last index.
* Q:  The list of methods I'm supposed to write seems to be missing methods I need in order for me to actually create and use an instance of the class.  For instance, my ArrayList-like class, if it does not have an add or append method, cannot be used for anything.  Can I create additional methods?  A:  Yes, you should create any methods you believe necessary in order to make the class work.  The list of methods given above are just the ones we're looking for to grade.
* Q: Do these structures hold a fixed size set of items (i.e., static), or can they grow at runtime? A: Your software should be able to automatically resize your array once capacity is reached, and may be tested beyond 100 elements. Remember that stacks, queues, etc. are (in the abstract) infinite capacity data structures.
* Q: What is FIFO and LIFO? A: Good question.
* Q: How will you test this code? A: Tests may use the same strategies and techniques shown to you in the first few HWs, but more complete.
* Q: Are comments (file headers, method headers, inline comments) important? A: Comments are now worth almost 2 letter grades, so I'd include them.
* Q: I think there is a way to relate these classes via inheritance.  Should I do this? A: We'll cover this later in the quarter; you should use no inheritance for this assignment.
* Q: If I'm not to use inheritance, then how can I code the equals methods in such a way that it can accept input of class Object? A: We'll see later on (pp. 465-471 to be precise) how to utilize inheritance principles to properly write the equals method. For now, while you are to accept anything of class Object as input, you don't have to check what the input types are. That is to say, you don't have to return false if the input is a different type than the type of the ArrayList, etc. class. To enable the comparison, convert the input into ArrayList (or whatever class is appropriate).

**What to Submit**

You should turn in four .java files, three are for your classes and the fourth is your test harness (driver).  Your data structure classes **have to be named** ArrayList, Stack, and Queue and your driver named the same as the sample driver.

Please **submit your files inside a zip archive called DataStructures.zip.**  If you are not sure how to create a zip archive, do a Google search on "create zip archive windows" (swap out your computer's operating system, as needed).  Make sure you give yourself enough time before the due time to learn about and create the archive!

**About This Document**

Original assignment by Rob Nash. Minor edits by Johnny Lin, January 2016-September 2019.

**Rubric**

Data Structures V1 (1)

| Data Structures V1 (1) | | |
| --- | --- | --- |
| **Criteria** | **Ratings** | **Pts** |
| This criterion is linked to a Learning Outcome Code compiles and driver executes |  | 4.0 pts |
| This criterion is linked to a Learning Outcome Clean, descriptive, and comprehensive comments |  | 8.0 pts |
| This criterion is linked to a Learning Outcome List class |  | 6.0 pts |
| This criterion is linked to a Learning Outcome Queue class |  | 4.0 pts |
| This criterion is linked to a Learning Outcome Stack class |  | 4.0 pts |
| This criterion is linked to a Learning Outcome Extended driver tests every substantive method of the classes |  | 6.0 pts |
| Total Points: 32.0 | | | |

**Submission**

Submitted!

Oct 17 at 11:57pm

[Submission Details](https://canvas.uw.edu/courses/1331881/assignments/4923143/submissions/3729452)

[Download DataStructures-1.zip](https://canvas.uw.edu/courses/1331881/assignments/4923143/submissions/3729452?download=59069076)

Grade: 30 (32 pts possible)

Graded Anonymously: no

[View Rubric Evaluation](https://canvas.uw.edu/courses/1331881/assignments/4923143/submissions/3729452#rubric)

**Comments:**

Your ArrayList throws an ArrayIndexOutOfBoundsException during large inserts. Size() doesn't return zero for empty ArrayLists.