Module 3 Assignment: Array-Backed Queue

Marcar esta página

Coding Assignment fecha límite Aug 26, 2023 05:50 -05

Array-Backed Queue

For this assignment, you will be implementing a *Queue* back by an array. Recall that a Queue is a first-in, first-out (FIFO) data structure; the first item inserted is the first item to be removed. Your array-backed Queue should follow the requirements stated in the javadocs of each method you are to implement.

IMPORTANT:

- **You will be given 5 attempts on this assignment, with a 30 minute cooldown between submissions.**
- **Please run your code before each submission to ensure that there are no formatting errors! If there are formatting errors in your code, your code will not be graded and a submission attempt will be logged. For more information, please review the Vocareum overview below.**

Capacity

The starting capacity of the *ArrayQueue* should be the constant `INITIAL_CAPACITY` defined in the file. Reference this constant as-is. Do **not** simply copy the value of this constant. Do **not** change the constant. If, while adding an element, the ArrayQueue does not have enough space, you should regrow the backing array to **twice** its old capacity.

This means that if the initial capacity of the backing array is 9, then after one resize the capacity will be 18, and after two resizes the capacity will be 36. Do **not** resize the backing array when removing elements.

Circular Arrays

The backing array in your *ArrayQueue* implementation must behave circularly. This means the front variable might wraparound to the beginning when you remove, to take advantage of empty space while maintaining $O(1)$ efficiency for all operations (adding will be amortized $O(1)$).

Dequeue

For this assignment, the front variable in ArrayQueue should represent the index that holds the next element to dequeue. After dequeuing that element, you should simply treat the next index in the array as the new front, though you will have to account for the circular behavior yourself. **Do not shift any elements during a removal**.

When elements are deleted from the backing array, the index that the element was removed from should be set to `null`. All unused positions in the backing array must be set to `null`.

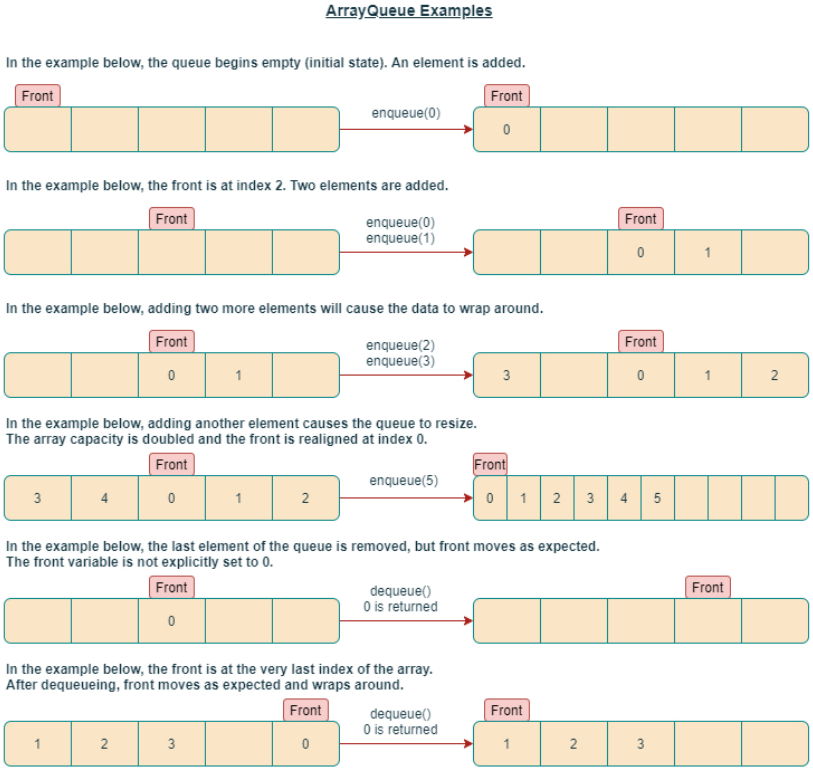
Additionally, after removing the last element in the queue, move the front variable like you normally would. Do **not** explicitly reset it to 0. This effectively means that going from size 1 to size 0 should not be a special case for your code.

Enqueue

Enqueue

For enqueuing, add to the back of the queue. To access the back of the queue, you can add the size to the front variable to get the next index to add to, though you will have to account for the circular behavior yourself. If there are empty spaces at the front of the array, the back of the queue should wrap around to the front of the array and make use of those spaces.

When resizing the backing array, "unwrap" the data. Realign the Queue with the front of the new array during the transfer. The front variable of the queue is once again at index 0. The resize case is the only case in which the data should be realigned.



General Tips

- To add to the back of the queue while accounting for the circular behavior, you can locate the correct index with `(front + size) % backingArray.length`. Of course, you will have to perform resizes whenever necessary. Can this technique also be used to help you realign your elements during a resize?
- We highly recommend copying the starter code and working in your preferred IDE in order to have access to features such as code completion, auto-formatting, and much more!

Here are general assignment guidelines that should be followed.

- Do not include any package declarations in your classes.
- Do not change any existing class headers, constructors, instance/global variables, or method signatures. For example, do not add throws to the method headers since they are not necessary. Instead, exceptions should be thrown as follows: `throw new InsertExceptionHere("Error: some exception was thrown");`
- All helper methods you choose to write should be made private. Recall the idea of Encapsulation in Object-Oriented Programming!
- Do not use anything that would trivialize the assignment. (e.g. Don't import/use java.util.ArrayList for an ArrayList assignment.)
- Always be very conscious of efficiency. Even if your method is to be $O(n)$, traversing the structure multiple times is considered inefficient unless that is absolutely required (and that case is extremely rare).
- If applicable, use the generic type of the class; do not use the raw type of the class. For example, use `new LinkedList<Integer>()` instead of `new LinkedList()`.

Use of the following statements should be avoided at all times.

package	System.arraycopy()	clone()
assert()	Arrays class	Array class
Thread class	Collections class	Collection.toArray()

Reflection APIs	Inner or nested classes	Lambda Expressions

The Vocareum (code editor) interface has six main components:

- The **Drop-Down** in the top left. This lets you choose from multiple available files. Note that this drop-down will only be visible in assignments that require multiple files.
- The **Run** button. This will compile your code and run a file scan. Running your code will not count towards your total allowed submission attempts, therefore you are free to run as many times as needed.
- The **Submit** button. This will compile your code, run a file scan, grade your assignment, and output results to console. Note that for most assignments in this class, you will only be allowed a limited number of submissions. A submission is counted when the submit button is clicked, regardless of whether or not your code can compile or if there are any file issues. Therefore, we **highly recommend** that you run your code before submitting to ensure that there are no issues that will prevent your code from being graded and that every submission attempt will generate meaningful results.
- The **Reset** button. This will revert all your changes and reset your code to the default code template.
- The **Code Window**. This is where you will write your code. For large coding assignments, we highly recommend copying the starter code and working in your preferred IDE to have access to features such as code completion, auto-formatting, and much more!
- The **Output Window**. This window will appear whenever you run or submit your code and will display the output for you to view.

For additional help, please visit the Vocareum information page located in the course information module!

‹ Anterior Siguiente ›