Project #4
CS 2210 – Fall 2022
Christopher LaFave

I. Requirements: Create an array-based queue that supports generics using the implementation provided on the project instruction sheet and test it thoroughly.

II. Design: I created 4 private variables, front, rear, capacity, and queue. The queue holds all the information, front points to the next value that gets pushed out, the rear points to the spot after the last element, and capacity is the space currently allocated for the array. These variables are all I needed to implement all of the methods.

III. Security Analysis: Because this ADT uses generics, in theory somebody could create an object type that can mess with things it shouldn't be able to. Other than that, I think the exceptions cover all the other input attacks.

IV. Implementation: I made two constructors, one that takes capacity, and one that defaults capacity to 64.
The Enqueue method takes in the element and pushes rear to the next spot. It checks if it needs to send rear back to the front of the array and also checks if it needs to increase the capacity.
Dequeue simply returns the next value and moves front to the next value. It has a check to see if it needs to loop back to the front.
Front is the same as Dequeue but it doesn't move the front pointer.
Size first checks if the queue is empty, then checks if it is full by asking if front and rear are pointing to the same thing. Because it checked if it was empty first, this always means that the queue is full. Last, it uses the equation showed in class to calculate the size.

V. Testing: I thoroughly tested every method, every exception, every constructor, if the looping works right, if the capacity extender works right, and if the generics work right.

VI. Summary/Conclusion: The program appears to function properly. It gives the correct answer for all valid inputs and throws the correct exceptions when tested.

VII. Lessons Learned: I learned much more about creating an ADT. I also learned how to implement generics. I became much more comfortable with JUnit, and learned the inner-workings of queues.