Q2-B Report

The Aims for Question 2 A was too complete enqueue and dequeue method within the class ‘MyArrayQueue’. Queue is a FIFO (First in First Out) Data structure.

Enqueue is the method which adds an element to the end of the list, it is important to note that in this instance we are using circular queues, so are tail value can be Infront of the first value if that is where there is space. In my enqueue method the queue is first checked if it is full, if full then we can create a temporary array double the size of are queue where the values from the queue starting from the front which value is held using variable ‘front’ are added to the temporary queue in order, along with the element which initially was passed as an argument in the enqueue method, this temporary array is then turned into are queue with the front becoming zero, and the number of elements increasing by 1, meaning now we have a larger queue which we can add more elements too. If the queue isn’t full, we can simple locate the next null value in the queue and then add that element to its location. We find the next null value through addition of the front marker with the (number of elements – 1) if this equates too or exceeds the highest index value then it loops around and adds the elements Infront the front marker, hence circular queue. If It doesn’t exceed then it is just added to the next ‘null’ location. Once this is completed the number of elements is increased, this will allow us to calculate the next null location when enqueue is called again.

The ‘dequeue’ method in comparison was simple it would initially check if it was empty, if it was it would throw an error telling the user the “Queue is empty” hence there are no elements to remove from the queue. If it wasn’t empty the front value would be assigned to a variable ‘frontVal’ that would be returned, the index location of the front value would then be swapped with a null value, meaning that if another element was to be enqueued it could take its place. front value is then increased by 1, if it is greater than or equal to queue length then it will be returned to 0 as it means it would have exceeded the index range of the array, and finally the number of elements is decreased by 1, allowing us to find the location of the next null element in the array when trying to enqueue a new element.

```
PS C:\Users\fiach_4diktta\Code\Cardiff_University\Year_1\CM1210 OOP\OOP_Coursework_2>
PS C:\Users\fiach_4diktta\Code\Cardiff_University\Year_1\CM1210 OOP\OOP_Coursework_2>  c:; cd 'c:\Users\fiach_4di
ktta\Code\Cardiff_University\Year_1\CM1210 OOP\OOP_Coursework_2'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '
-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\fiach_4diktta\AppData\Roaming\Code\User\workspaceStorage
\96a5dbf571ab56e7eb982b5de952da75\redhat.java\jdt_ws\OOP_Coursework_2_6ef04ef7\bin' 'MyArrayQueue'
[null, null, null]
ENQUEUE
[a, null, null]
ENQUEUE
[a, b, null]
ENQUEUE
[a, b, c]
DEQUEUE
a
[null, b, c]
ENQUEUE
[d, b, c]
ENQUEUE
[b, c, d, e, null, null]
ENQUEUE
[b, c, d, e, f, null]
ENQUEUE
[b, c, d, e, f, g]
DEQUEUE
b
[null, c, d, e, f, g]
DEQUEUE
c
ENQUEUE
[h, null, d, e, f, g]
DEQUEUE
d
[h, null, null, e, f, g]
DEQUEUE
e
[h, null, null, null, f, g]
DEQUEUE
f
[h, null, null, null, null, g]
DEQUEUE
g
[h, null, null, null, null, null]
DEQUEUE
h
[null, null, null, null, null, null]
DEQUEUE
Queue is empty
[null, null, null, null, null, null]
PS C:\Users\fiach_4diktta\Code\Cardiff_University\Year_1\CM1210 OOP\OOP_Coursework_2>
```