Assignment4

Name and Surname: Muhammed Enes Yavuz

Identity Number: 21989712

Problem Definition

In this assignment we are working with stack and queue data structures. We must implement our stack and queue. We have some data in stack.txt and we push this data to our stack. Also we have a queue.txt and we enqueue this data to our queue. After than with the help of the temporary stacks and queues we are doing some operation in this stack and queue. These operations must be according to our command.txt.

Algorithm

While implementing stack and queue I use node implementation. In this implementation we have a data and a reference to the next data. In the stack implementation we must implement our code according to last in first out principle so while implementing MyStacks methods I use only top element .For the queue implementation we must implement our code according to first in first out principle so while implementing MyQueue methods I use two nodes and this names are front and rear.

Solution

After the implement MyStack and MyQueue I store data of command.txt with the help of the ReadFromFile class in a array and send this to Operations class. In the Operation class I also store data's of stack.txt and queue.txt in different arrays. Then for the stack I push all this data in a stack. And for the queue I enqueue all data in in a queue. I have two more classes named StackOperations and QueueOperations. In this classes I write all methods from expect to us with the help of the temporary stacks and also queues. While I write this methods I use push, pop, peek and empty methods in the MyStack class and enqueue, dequeue and isEmpty methods in the MyQueue class. One of the methods from expect to us for stack is removeGreater method. In this method we eliminate elements in our stack if they are bigger then given integer. In this first I control stack is empty or not. While it is not empty I remove top element from stack and if it is greater than given integer I don't add them into my temporary stack else I add them into my temporary stack. After this I have a temporary stack and it have values inside smaller or equal to the given integer. But this stack is reverse of my first stack so with a second while loop I removed values one by one and add them to my first stack. The other method of stack is calculateDistance method. In this method first I create an integer named total. After then in a while loop I remove top element of my stack and store in a integer named element. I add this element to a temporary stack named thirdStack. I remove an element again and call it temp. I increase total to absolute value of element minus value. I

add the temp in a temporary stack named secondStack and add them again my firstStack. So after this operation my firstStack has one less element. With help of the this while loops I can compare to all element with the others. Also in this while loops I add the top element to thirdStack and with help of another while loop I add elements from the thirdStack to firstStack. In the addOrRemove method if given integer is less than zero I removed absolute value of given integer element from my stack. If given integer is bigger than zero I call random number and add them to my stack given integer time. In the reverse method first I calculate how many element in my stack after that while my stack is not empty if given integer bigger than element count I change value of given integer to size and transfer firstsStack elements one by one to secondStack, after I transfer elements of secondStack to thirdStack one by one at the end I transfer elements of third stack one by one to my firstStack. So after this operation firstStacks first given integer times element has been reversed. For the sorting method I removed top element from my stack and called this temp and then add this to secondStack. With help of another while loop if top of firstStack is smaller than secondStack top element swap them else just add top element to the second stack. This operation is give me sorted stack but from big to small so I should reverse my stack for this I remove elements from secondStack and I add them to the firstStack. Last method of stack is distinctElements. In this method first I copy my firstStack with the help of the fourthStack and fifthStack. So after this my firstStack values stored in fifthStack. After this I select top element of my stack store them in thirdStack and compare it with all other elements in the stack if they are equal I did nothing so I lose them else I store them in the secondStack. I transfer all values of the secondStack to the firstStack and select the next element in the stack.So I do same operation while my firstStack is not empty. Then I count elements in my ThirdStack all of them are different. At the end I want my firstStack don't change so transfer all values of fifthStack to the firstStack. For the queue my removeGreater, calculateDistance , addOrRemove and distinctElements methods similar to the same name methods. In the reverse method first I count how many elements in my queue while my stacks is not empty if count of elements is smaller than given integer change value of given integer to size. I add them to the a new stack. But not all of them only count as given integer so probably still I have elements in my queue. After that first add all values of my stack to the secondQueue and then all values of firstQueue to the secondQueue. At the end of this method I transfer all elements to the firstQueue. For the sorting method of queue first I pick front element of queue and compare to all of other elements when I find smaller element than first element I choose this element smallest so I can find smallest element in the queue. In a second loop I remove smallest element or maybe smallest elements from my queue and add them a thirdQueue so in the finish my thirdQueue is sorted. I should refill my firstQueue so I remove elements from my thirdQueue and add them to the firstQueue while doing this operation nothing change in my queue it is still sorted.

removeGreater for Stack

While I am writing removeGreater method I used empty, pop and push methods from MyStack class. All of this methods have constant complexity. I have two while loops inside the removeGreater method. If we have n element inside the stack this while loops executed n+1 times. So this loops have O(n) complexity. And at the and I can say my removeGreater method has O(n) complexity.