CS 3310
Dr. Gupta
Mariia Kravtsova
SLC Report
October 18, 2016

Assignment 2 implementation has the following classes:
NewStack - implementation of stack
NewQueue - implementation of queue -
StackQueue - implementation of stack using two queues
QueueStack - implementation of queue using two stacks
MinValueStack Implementation of a stack with current minimum value using two stacks

To save time the nodes of the linked list are created in each stack and queue implementation. In this way there could be more control in case separate modifications are needed, even though it might seem redundant. NewStack has the following methods and complexities:
isEmpty() - O(1)
push() - O(1)
pop() - O(1)
print() - O(n)
space complexity of O(n)

NewQueue has the following methods and complexities:
isEmpty() - O(1)
enqueue() - O(1)
dequeue - O(1)
getSize() -O(1)
print() - O(n)
space complexity of O(n)

StackQueue class requires two stacks to replicate a behavior of a queue. The way I approach the problem is to push item onto stack1, then pop items from stack1 onto stack2, then pop stack2. This way the order of FIFO is replicated. In this case the enqueue remains as O(1), and dequeue is O(n) with space of O(1).

QueueStack class creates two queues which replicate the behavior of a stack. The push method has time and space complexity of O(1), and pop method has the time complexity of O(n) and space complexity of O(1). The push method enqueues the item onto queue1 and the pop method has element from queue1 dequeued and enqueued onto queue2, but last element in q2 is kept as top. Then remove last element from queue1, and swap elements back to queue1. Even though the swap is not necessary, I thought it was a cleaner way to keep track of things.

MinValueStack class does not use generics. I had a very hard time understanding and implementing this with IComparable in Java. One of the emails and testing files pointed that this should be tested with Integer, so I assumed so with my implementation. My solution requires two stacks, one to hold the current minimum value, and another to have all the values on the stack. While doing so we have to maintain the same size for the both stacks. So then poping the values does not loose the minimum value. The complexity of this is O(1), even though we are using extra space for the minimum stack.

Below is the output sample from my program on the txt files provided:

```
Queue from Stacks enqueue:
IcomefromWMU!
Queue from Stacks dequeue:
Queue is empty
Stack from Queues push:
IcomefromWMU!
Stack from Queues pop:Stack is empty.

Queue from Stacks enqueue:
IcomefromWMU!
Queue from Stacks dequeue:
!
Stack from Queues push:
IcomefromWMU!
Stack from Queues pop:
I

MinValue:
push:
67 4 53 43 7 4 8 348 4 98 4 23 78 1 2
pop:
1
```