Made by: Kevin Boudreaux

UTD ID: KCB180002

Class: CS 3345.006

ArrayStack:

My solution for arraystack was to have a new array every time a push or pop was called. This new array would be one size bigger or smaller than the past version to make room or remove the spot where the inserted/removed note was/is. This causes the upper bound complexity of my push and pop functions to be O(n). My isEmpty function is O(1) since it returns a true/false statement and has no loops inside the function. My peek function is also O(1) since it returns the first element in the array.

LinkedStack:

My solution for the linked list stack was to insert a new node(link) at the front of the list every time a push was called. and to remove the front node of the list when a pop was called. My isEmpty function's upper bound complexity is O(1) since it returns a true/false statement. My push function is also O(1) since it does not need to loop through to move everything over like the array's push function does; instead, my push function inserts a node at the very front of the list. The same thing happens with my pop function so it will also be O(1). My peek function returns the head node if it is not null but if there is no head node it will throw an empty stack exception making it O(1). Overall, for the given problem linked list is the more efficient way to handle the data.

After testing my code and finishing it I found a command that copies arrays (System.arraycopy) and after reading how it works and testing to see how fast it is versus what I had before I found it saved around 14 seconds on my machine. The test cases for each trial were only one so there could be a lot of factors on why one was faster against the other, however it did condense the amount of lines needed from two (one for the loop one for the command) to just one line, so it could be argued that it is more compact than the previous version.

For Linked Lists I made some overloaded constructors so my code is cleaner and could be deemed more efficient in that effect but as with arrays I did not use anything extremely fancy because I believed it would be as efficient or less efficient than what was written.