Problem definition

In this experiment, we are expected to implement our own Stack and Queue classes to read and do operations specified in the input.txt.

With the help of this experiment, we will get familiar with data structures such as Stack and Queue.

Method & Algorithm

I created two classes named Stack and Queue and writed main methods such as push, pop, enqueue, dequeue, size etc. Moreover, in order to make change from the orders which is taken from commands.txt, I created InputManager class which splits all lines and calls functions. With done that, I created two different outputStrings (stackOutput, queueOutput) which store operations informations in it. After reading command.txt, stackOutput and queueOutput is written to the two different txt files named stackOut.txt and queueOut.txt.

Analysis of the distinctElements Method

STACK ➡ I thought that it will be difficult if I try to count distinct elements in the stack if I use unsorted stack so my method first creates an empty stack, fills the main elements in this empty stack and sorts all the elements. With done that, another stack which is empty is created by method. Method takes elements from the sorted stack and add them to empty stack if the element is not added before. Thanks to the sorting, method just needs to check last added element. After the operations done, it returns the size of the filled stack. Sorting method's time complexity is $\Theta(n^2)$ *and distinctElements method's time complexity* $\Theta(n)$ *so the time complexity is* $\Theta(n^2)$.

QUEUE ➡ In queue, this method creates an empty queue and starts to add elements from main queue to empty queue and sorts them . Basicly, function keeps two variables (a, b) which are added very recently. Compare them if they are equal . If they are not equal b is added to empty queue and a becomes b. After the transactions, it returns the size of the filled queue . Sorting method's time complexity is $\Theta(n^2)$ *and distinct elements method's time complexity is* $\Theta(n)$ *so the time complexity is* $\Theta(n^2)$.