| | |
|---|---|
| **Name** | Lahiru Hasaranga Weliwitiya |

# Part A

a) *(code provided in folder labelled "A - a" )*

b) In order to not use any array methods for enqueue() and dequeue(), a new empty array will need to be created, and loops will be needed to incrementally make changes to it, excluding/including the item that needs to be added/removed using separate variables to indicate their index positions.

   - The enqeue method of Stack involves adding an item to the "top" of the Stack, so a new array can be created, copying all the items from the previous array with the new item added to the end.

   - The dequeue method of Stack involves "popping" off an item of the top of the Stack, so a new array can be created, copying all the items from the previous array, with the new item removed from the end.

   - The enqueue method of Queue involves adding an item to the start of the accompanying array & shifting the rest of the items one index position to the right. Accordingly, a new array can be created using a loop, placing the new item in the first index position, copying all the items from the previous array.

   - The dequeue method of Queue involves removing the first item of the accompanying array, so a new array can be created using a loop by copying all the items from the previous array, ignoring the first item.

   Alternatively, another base data structure (eg: dictionary/Hashmaps) can be used, which completely avoids using array methods, but is impractical for this case.

c) The changes I made are listed below:

   - The entire code base was changed to support the Generic Type "T", instead of being limited to Strings (and String arrays).

   - The return type of the "enqueue()" method was modified to return void, as it seemed unneccesary to return the array of the Queuable structure each time it's instance is modified. This also decreases the application overhead.

   - An "enqueueList" method was added, to make adding a list of items more convenient, and is dependent on the enqueue method.

   - A "peek" method was added, so that the item that is featured prominently in either data structure (i.e. the top of the Stack & the start of the Queue) can be observed.

   - Auxiliary methods with suitable function names that depend on the existing queue & enqueue methods, "push" & "pop" were added to the Stack class to increase readabilty:

   - toString() method implementations were added, to support printing the contents of the arrays in each data structure.

   *(code provided in folder labelled "A - c")*

## Part B

*(code provided in folder labelled "B - a,b,c" )*

Tech-stack used:

- Frontend - Flutter (Web)
- Backend - Dart
- Database - Firebase (Realtime)