Lesson-3-Class Notes and Review points

Several Terms can use for behaviors, operations, methods, functions.

Implement the function, it includes set of instructions(Algorithm)

List ADT – No implementation/ What to do

```
Int size();
Add(item);
```

Implementation Code will tell you how to do.

```
Int arr[];
Int size;
// Size Algorithm
Int size(){

   Return  size;
}
```

Applications of Stack – Call Stack in JavaScript

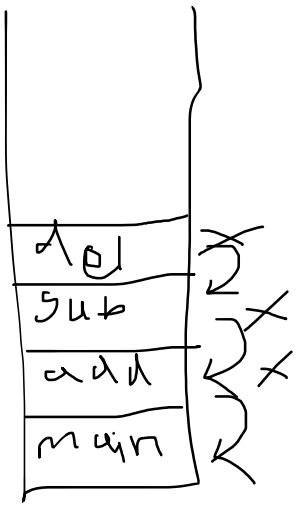
Method Call

```
Add()
{
 Sub()
}
Sub()
{

 Del()
}

Main(){

Add()
}
```



Stack ADT – Operations

- void push(object): inserts an element
- object pop(): removes and returns the last inserted element
- object top(): returns the last inserted element without removing it
- integer size(): returns the number of elements stored
- boolean isEmpty(): indicates whether no elements are stored

Note: pop or top on an empty stack causes an EmptyStackExceptionto be thrown
if stack is full, throw a StackFullException for the push operation

Stack Runtime Analysis

- Let ***n*** be the number of elements in the stack, each operation runs in time ***O***(1).
- The maximum size of the stack must be defined at creation and cannot be changed.
- Linked list implementation of a stack performs O(1) for each operations.

Implementation

- Stack can implement using both array and linked list.
- Since the insertion and deletion operations on a stack are made only at the end of the stack, using an array to implement a stack is more efficient than a linked list. [Straight forward]

Stack Implementation using Single Linked List

The array implementation of a stack has the advantages of simplicity and efficiency. However, one major disadvantage is the need to know what size to declare the array. Some reasonable guess has to be made, but this may turn out to be too small (and the program has to halt) or too big (and storage is wasted).

To overcome this disadvantage, a linked list can be used. Now, we will allocate storage for an element only when it is needed.

The stack is implemented as a linked list with new items added at the head of the list. When we need to pop the stack, the item at the head is removed.

First, we will need to define a Node class that will be used to create nodes for the list. We will use the following declarations:

Visualization: https://www.cs.usfca.edu/~galles/visualization/StackLL.html

```
class Node {
   constructor(d) {
      this.data = d;
      this.next = null;
   }
}
```

Algorithm push(o) p ← new Node(o); p.next ← top; top ← p; Algorithm isEmpty() return top==null	Algorithm pop() if isEmpty() then throw EmptyStackException else hold ← top.data; top ← top.next; return hold; Algorithm peek() if isEmpty() then throw EmptyStackException else hold ← top.data; return hold;

To compute the size, can introduce the count variable, each push() increment the counter and each pop decrement the counter.

Queue ADT – Operations

- void enqueue(object): inserts an element at the end of the queue
- object dequeue(): removes and returns the element at the front of the queue, remove or front on an empty queue throws an EmptyQueueException
- object front(): returns the element at the front without removing it
- integer size(): returns the number of elements stored
- boolean isEmpty(): indicates whether no elements are stored

Implementation

- Queue can implement using both array and linked list.
- Circular array is the best way to implement queue due to memory utilization of deleted elements space.

- Since deletions are made at the beginning of the list and insertions are made at the end of the list, it is more efficient to implement a queue using a linked list than an array list. [Straight forward]

Types of Queue

- Regular Queue (Array/ Linked List) – We focused in this lesson.
- Circular Queue(Efficient Array Implementation – Memory utilization)
- Double ended queue(Both ends can perform Insertion and deletion – Doubly LL)
- Priority Queue (Array/Linked List)

Review Questions for Lesson-3

1. Able to write the pseudo codes for stack operations using array and single LL.
2. Performance of stack operations.
3. Applications of Stack and Queue.
4. Queue implementation using Circular queue and DLL. Analyze its performances.