What is a Queue?

Queue is a data structure that stores items in a First-In/First-Out manner.

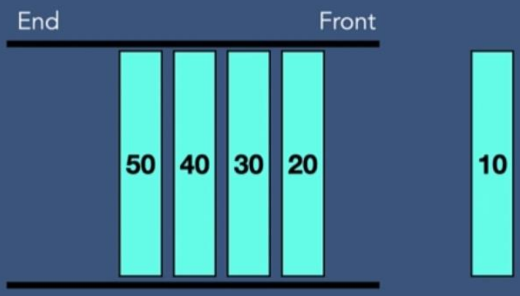
A new addition to this queue happens at the end of the queue.

First person in the queue will be served first

FIFO method - First in First Out

What is a Queue?

Queue is a data structure that stores items in a First-In/First-Out manner.

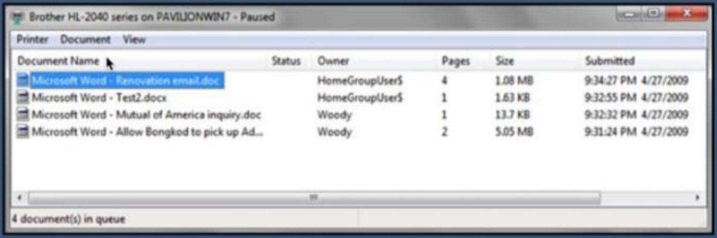


Why do we need Queue?

Point sale system of a restaurant

Why do we need Queue?

Printer queue



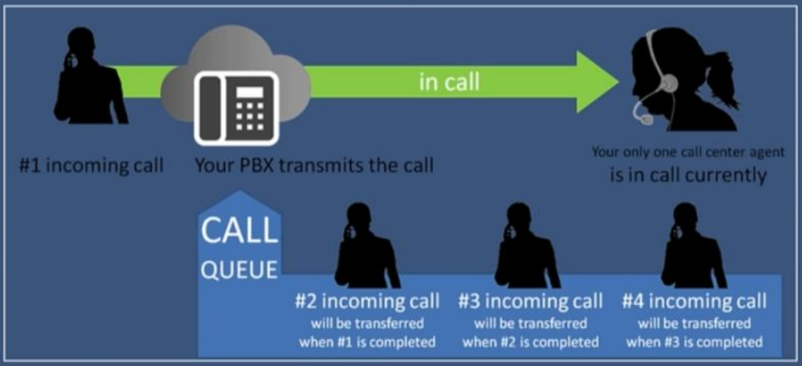
Brother HL-2040 series on PAVILIONWIN7 - Paused

Document Name	Status	Owner	Pages	Size	Submitted
Microsoft Word - Renovation email.doc		HomeGroupUser$	4	1.08 MB	9:34:27 PM 4/27/2009
Microsoft Word - Test2.docx		HomeGroupUser$	1	1.63 KB	9:32:55 PM 4/27/2009
Microsoft Word - Mutual of America inquiry.doc		Woody	1	13.7 KB	9:32:32 PM 4/27/2009
Microsoft Word - Allow Bongkod to pick up Ad...		Woody	2	5.05 MB	9:31:24 PM 4/27/2009

4 document(s) in queue

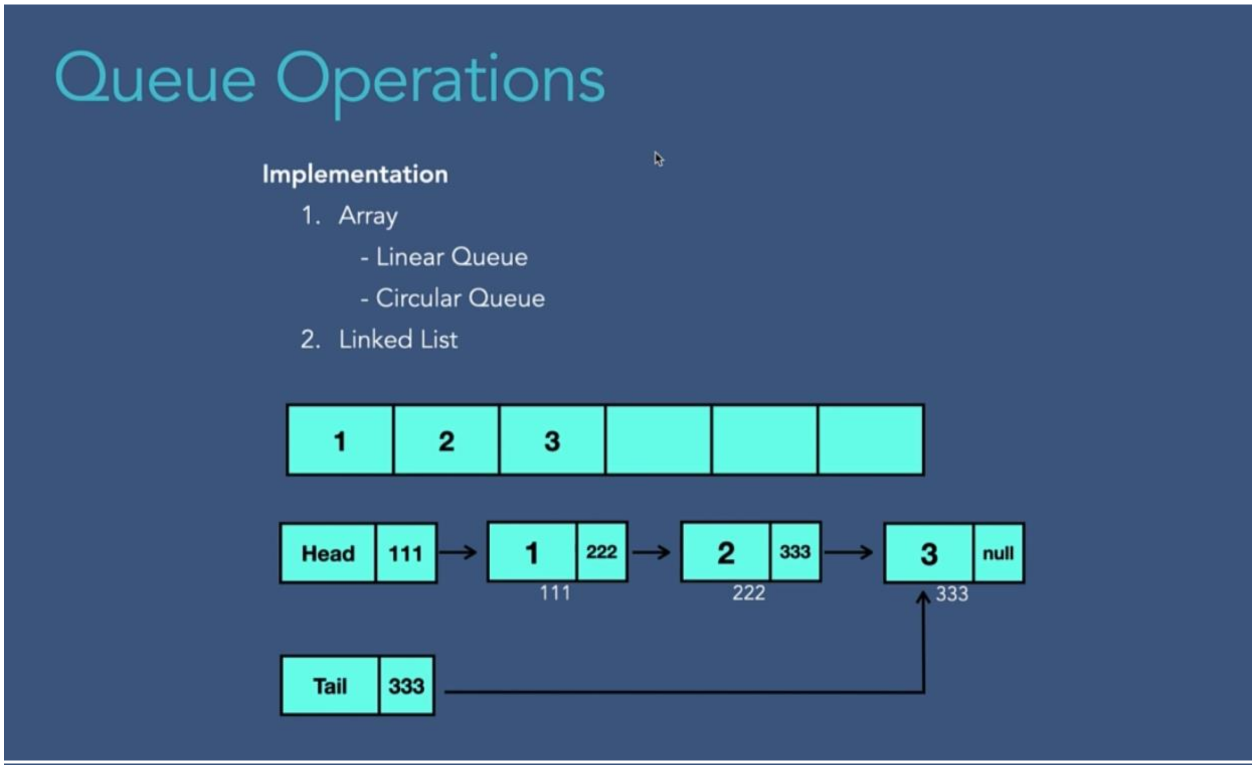
Why do we need Queue?

Call center phone systems



Queue Operations

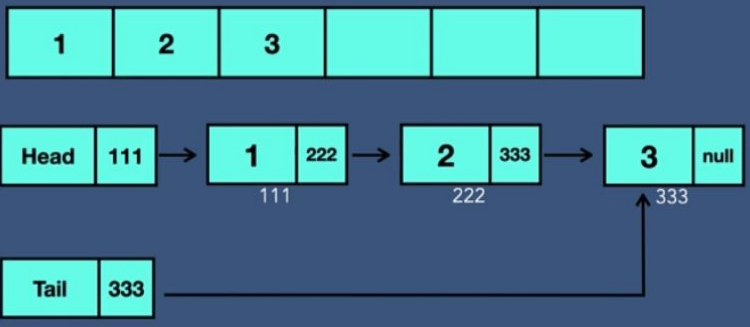
- Create Queue
- Enqueue
- Dequeue
- Peek
- isEmpty
- isFull
- deleteQueue

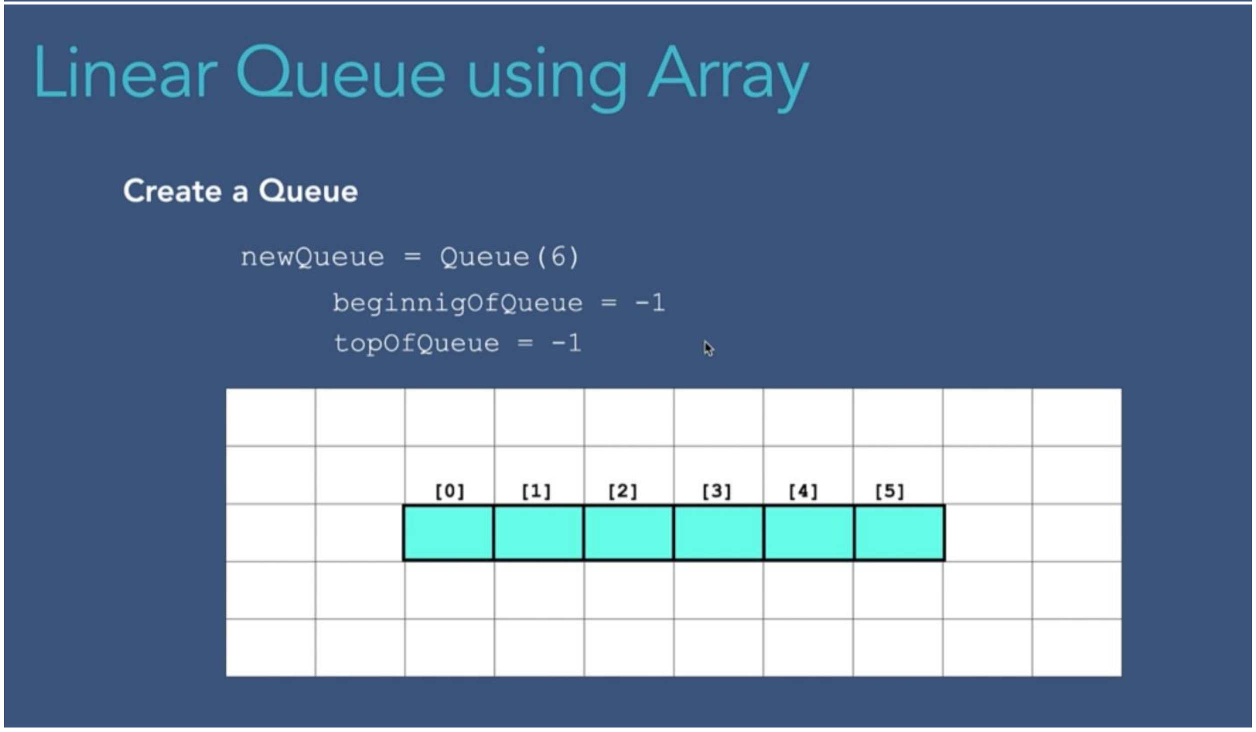


Queue Operations

Implementation

1. Array
   - Linear Queue
   - Circular Queue
2. Linked List

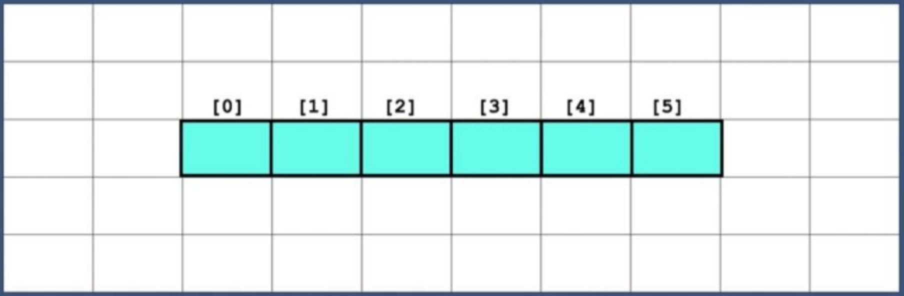


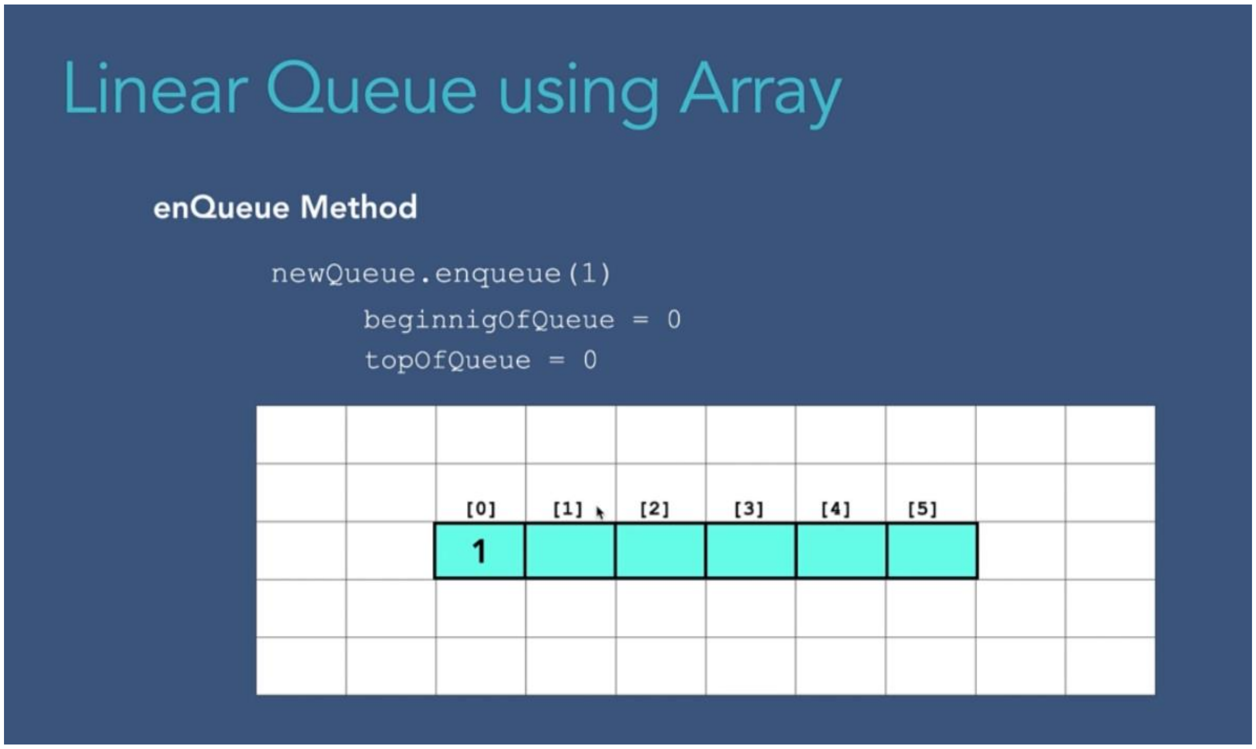


Linear Queue using Array

Create a Queue

```
newQueue = Queue(6)
      beginnigOfQueue = -1
      topOfQueue = -1
```

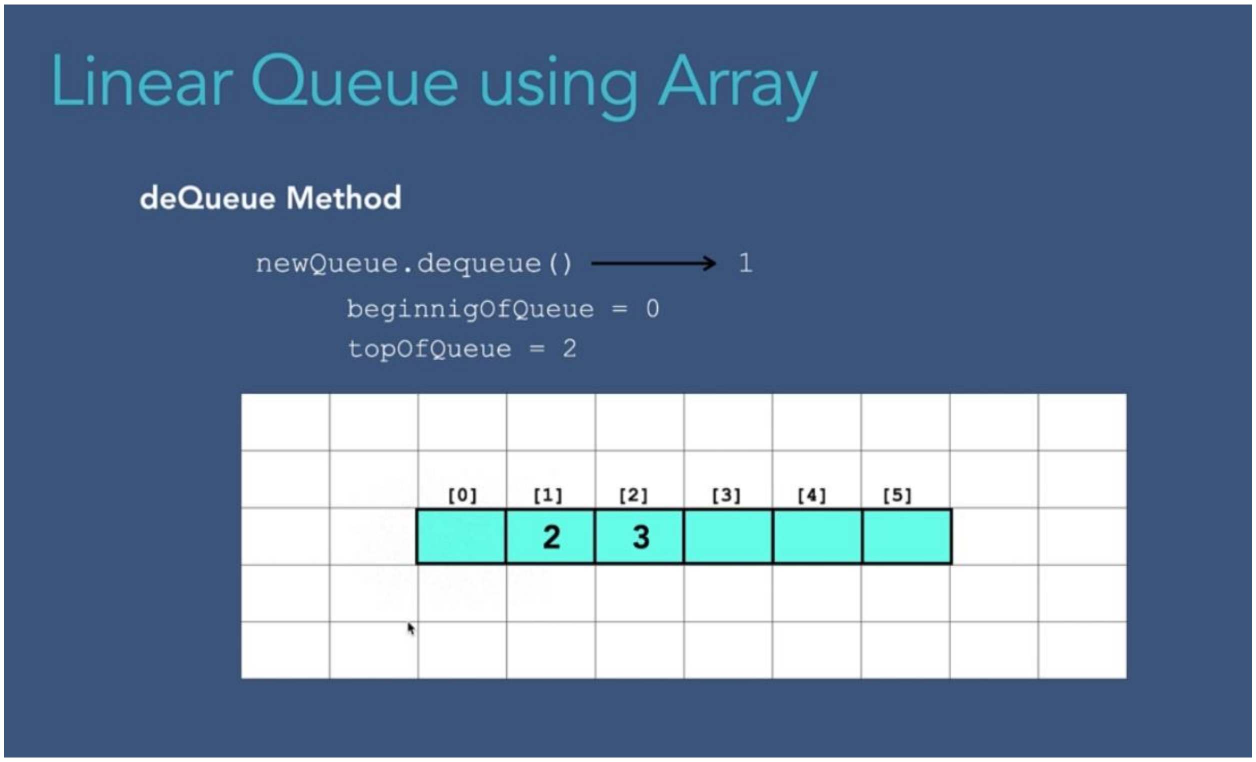




Linear Queue using Array

enQueue Method

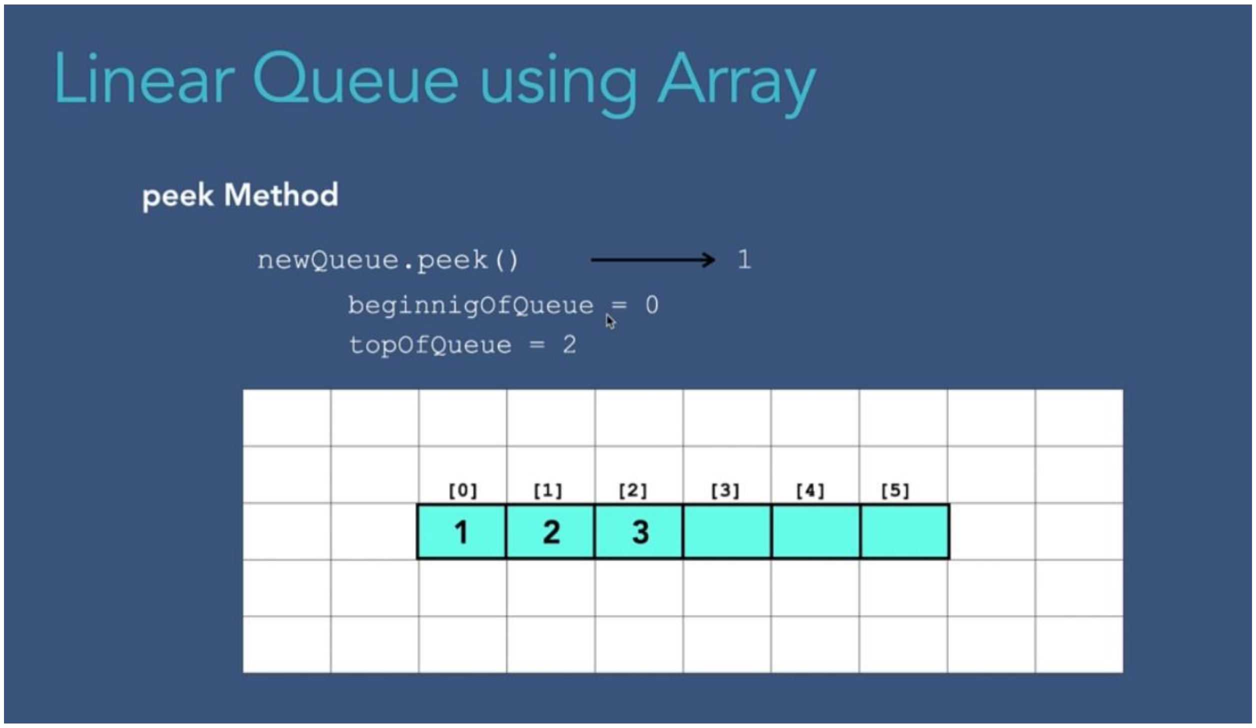
```
newQueue.enqueue(1)
      beginnigOfQueue = 0
      topOfQueue = 0
```



Linear Queue using Array

deQueue Method

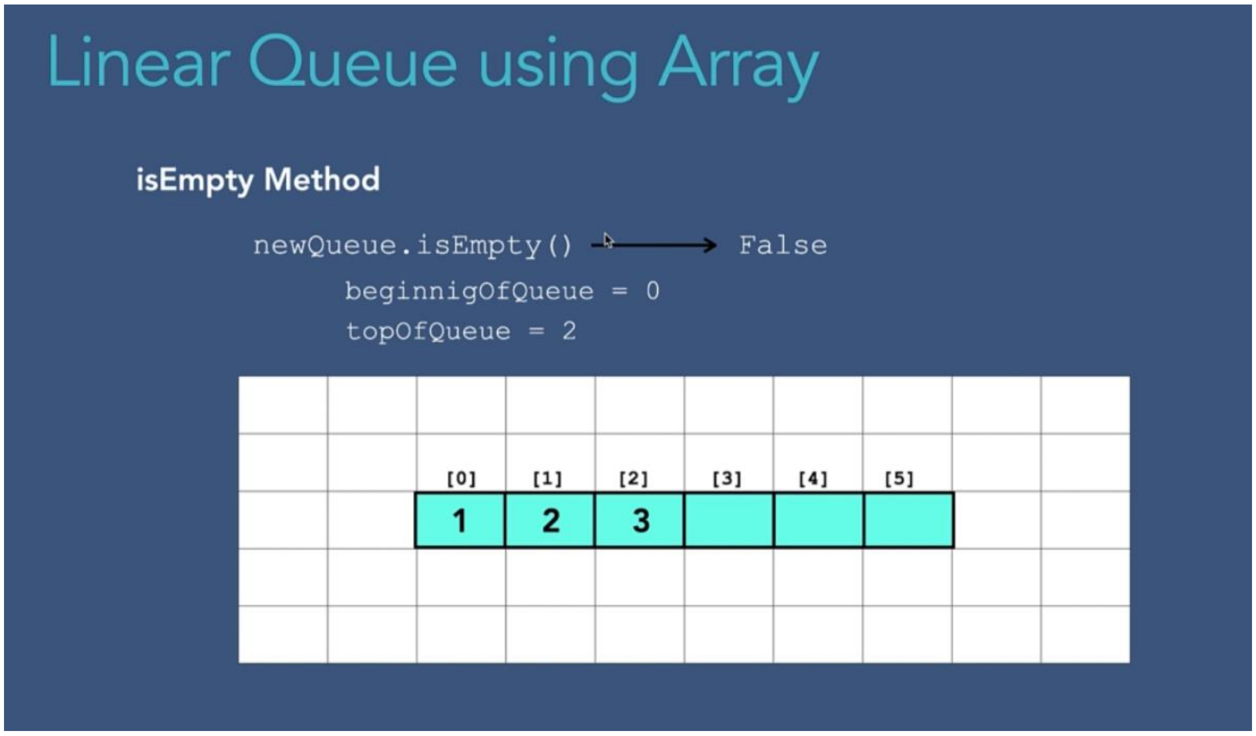
```
newQueue.dequeue() ⟶ 1
      beginnigOfQueue = 0
      topOfQueue = 2
```



Linear Queue using Array

peek Method

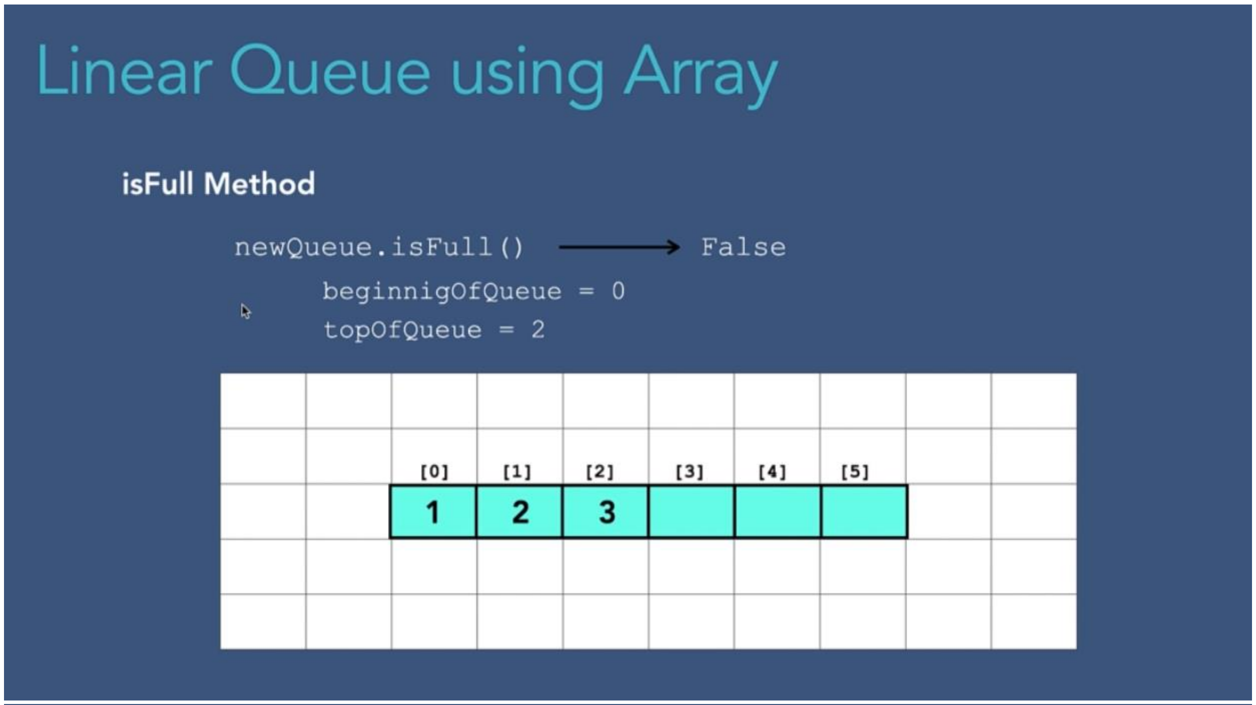
```
newQueue.peek()  ───→  1
      beginnigOfQueue = 0
      topOfQueue = 2
```



Linear Queue using Array

isEmpty Method

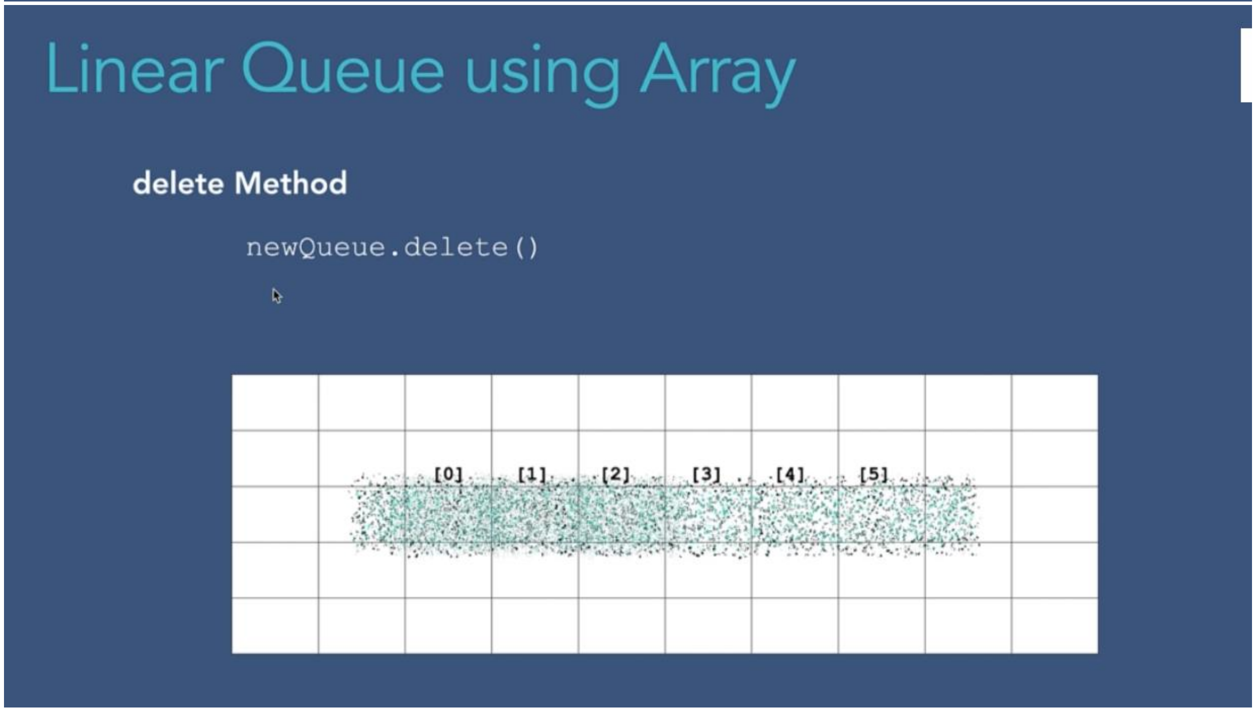
```
newQueue.isEmpty()  ───→  False
      beginnigOfQueue = 0
      topOfQueue = 2
```



Linear Queue using Array

isFull Method

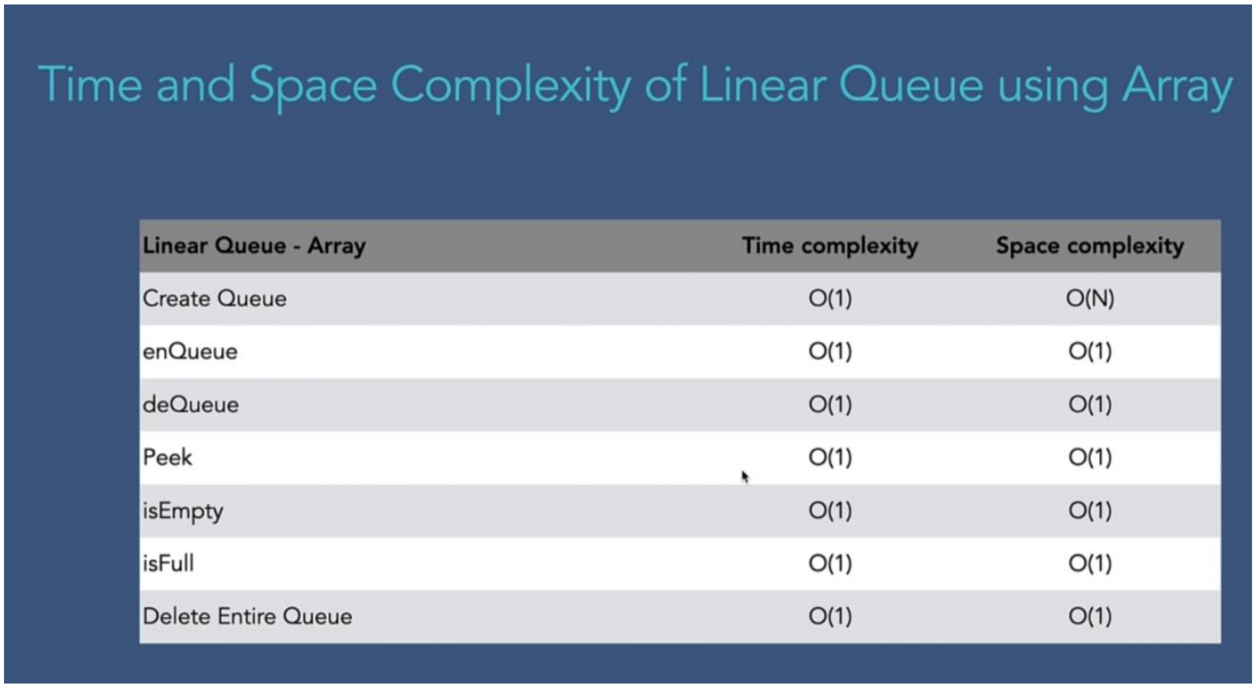
```
newQueue.isFull()  ⟶  False
      beginnigOfQueue = 0
      topOfQueue = 2
```



Linear Queue using Array

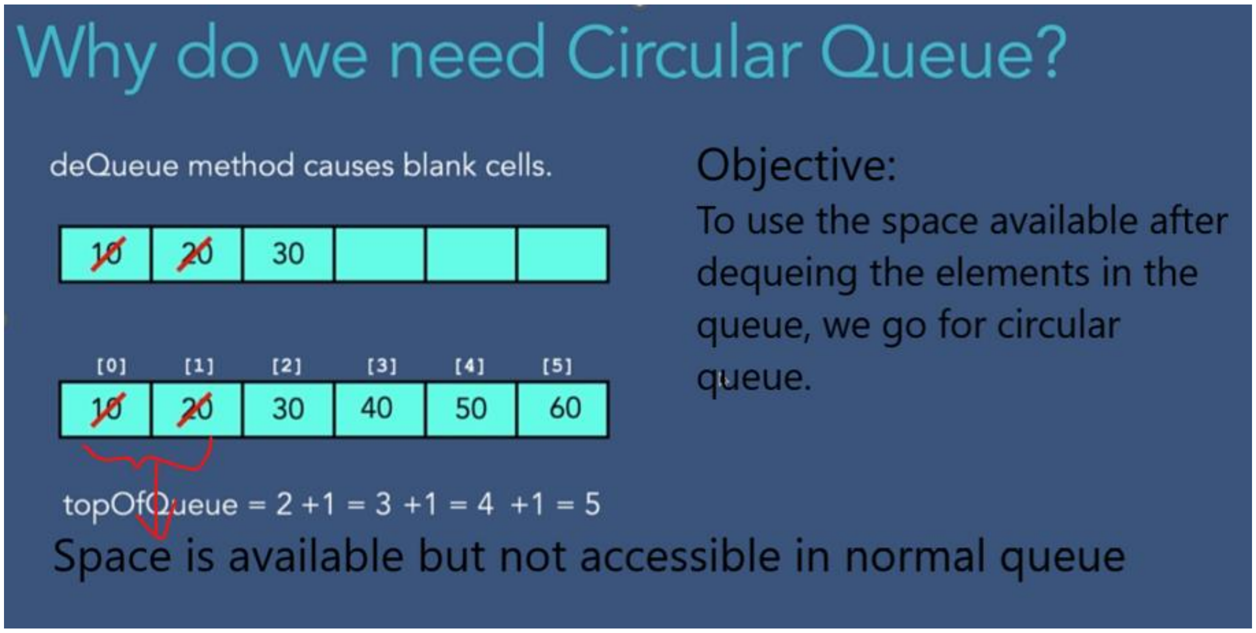
delete Method

```
newQueue.delete()
```



Time and Space Complexity of Linear Queue using Array

Linear Queue - Array	Time complexity	Space complexity
Create Queue	O(1)	O(N)
enQueue	O(1)	O(1)
deQueue	O(1)	O(1)
Peek	O(1)	O(1)
isEmpty	O(1)	O(1)
isFull	O(1)	O(1)
Delete Entire Queue	O(1)	O(1)



Why do we need Circular Queue?

deQueue method causes blank cells.

topOfQueue = 2 +1 = 3 +1 = 4 +1 = 5

Space is available but not accessible in normal queue

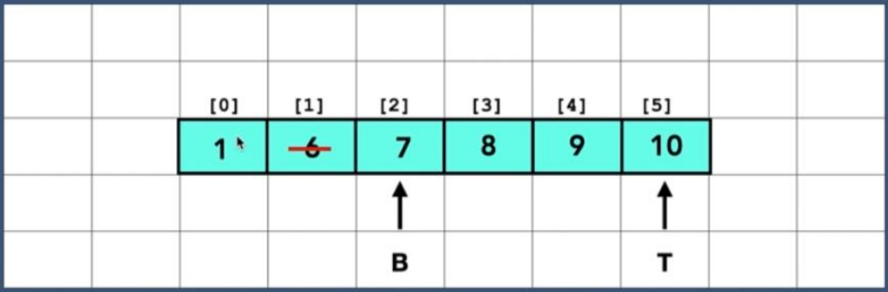
Objective:

To use the space available after dequeing the elements in the queue, we go for circular queue.

Circular Queue using Array

deQueue Method

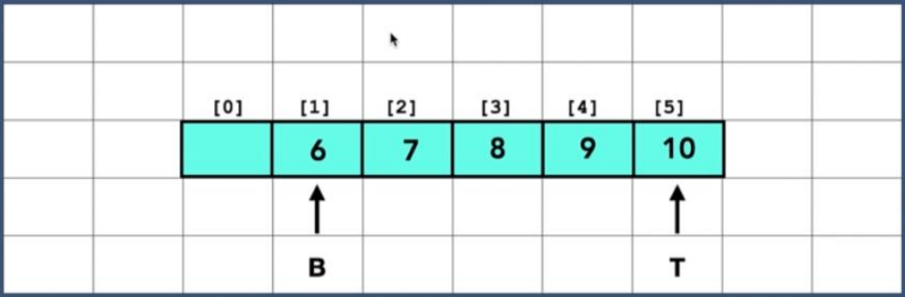
```
newQueue.enQueue(1)
        beginnigOfQueue = 2
        topOfQueue = 5
```



Circular Queue using Array

Peek Method

```
newQueue.peek()      ⟶  6
        beginnigOfQueue = 1
        topOfQueue = 5
```

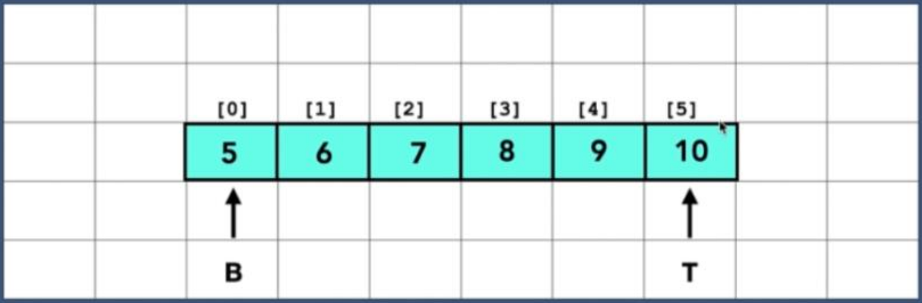


Circular Queue using Array

isFull Method

```
newQueue.isFull()
      beginnigOfQueue = 0
      topOfQueue = 5
```

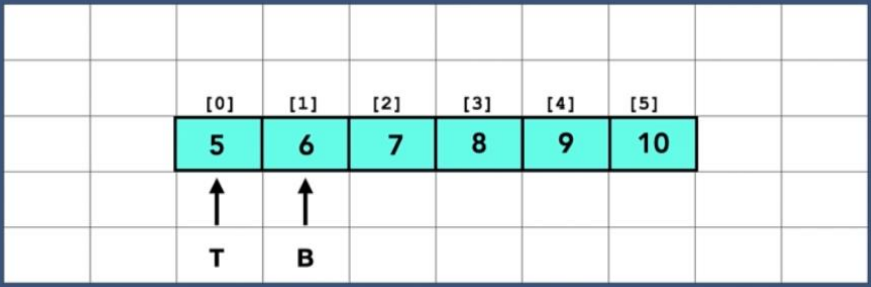
Condition:
The top of the queue should be at the end and the beginning of the queue should be at the start position.

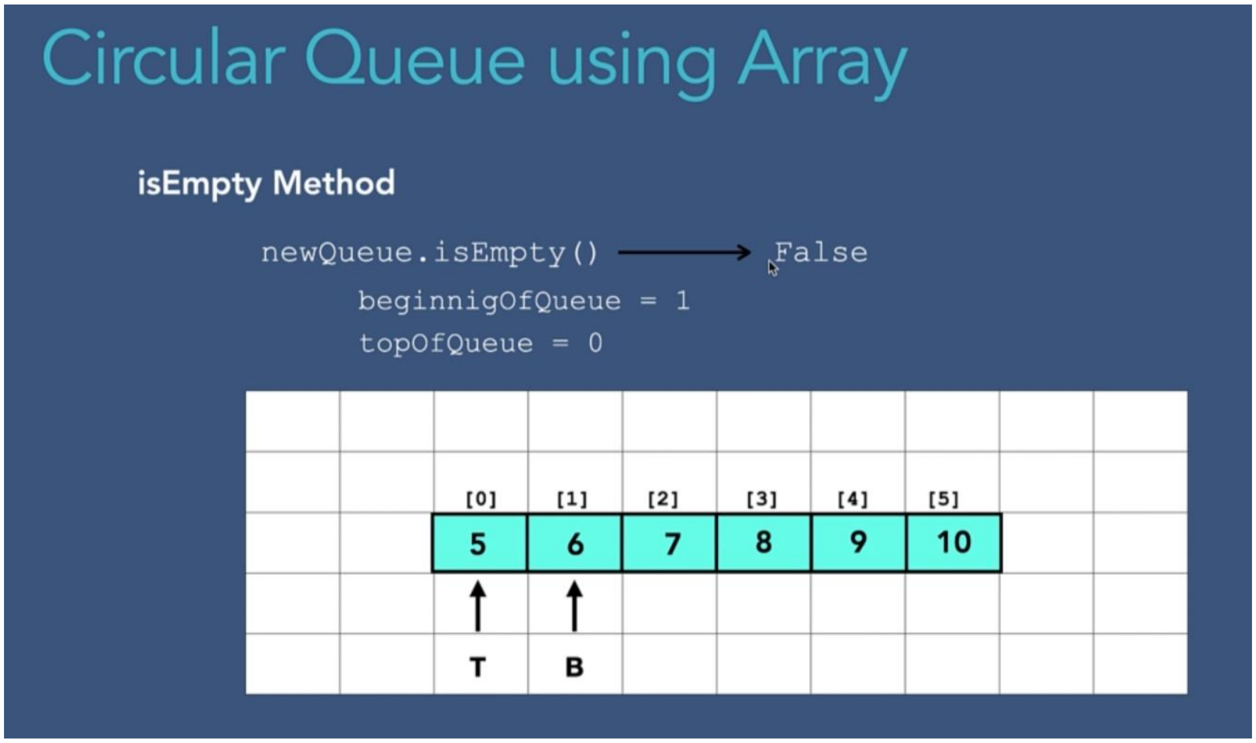


Circular Queue using Array

isFull Method

```
newQueue.isFull()  ————→  True
      beginnigOfQueue = 1
      topOfQueue = 0
```

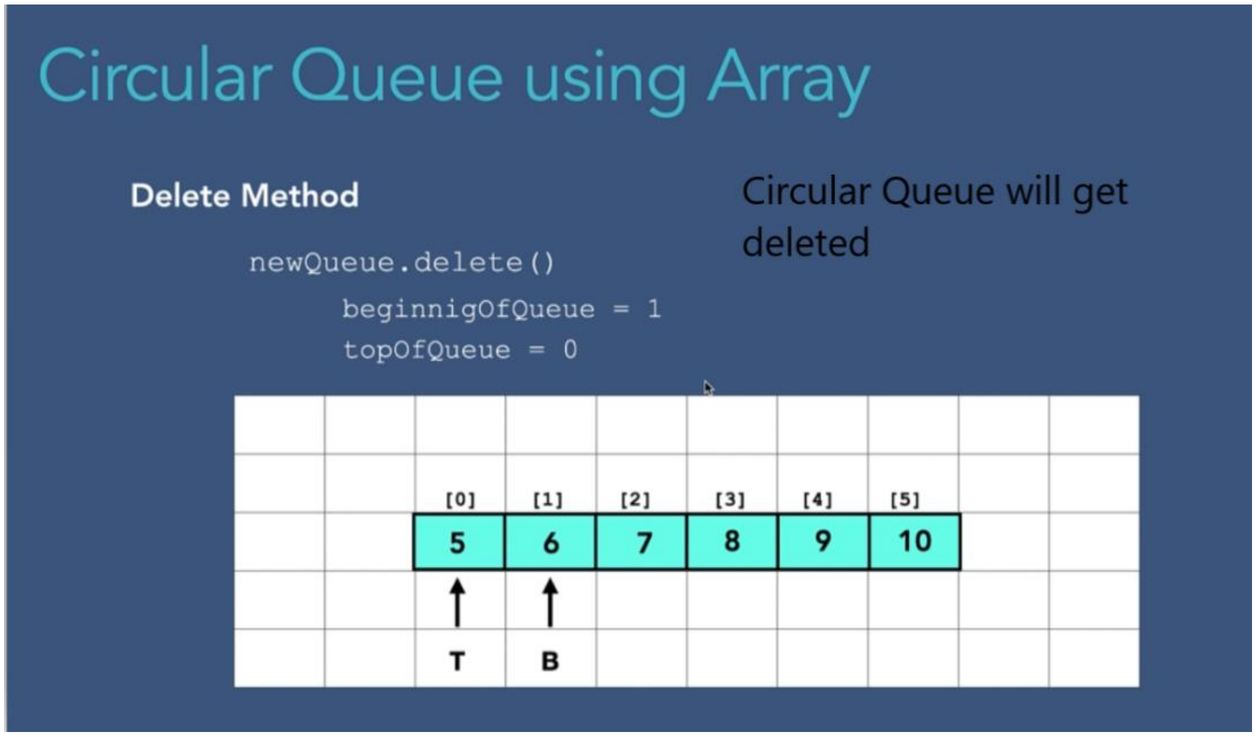




Circular Queue using Array

isEmpty Method

```
newQueue.isEmpty() ⟶ False
      beginnigOfQueue = 1
      topOfQueue = 0
```

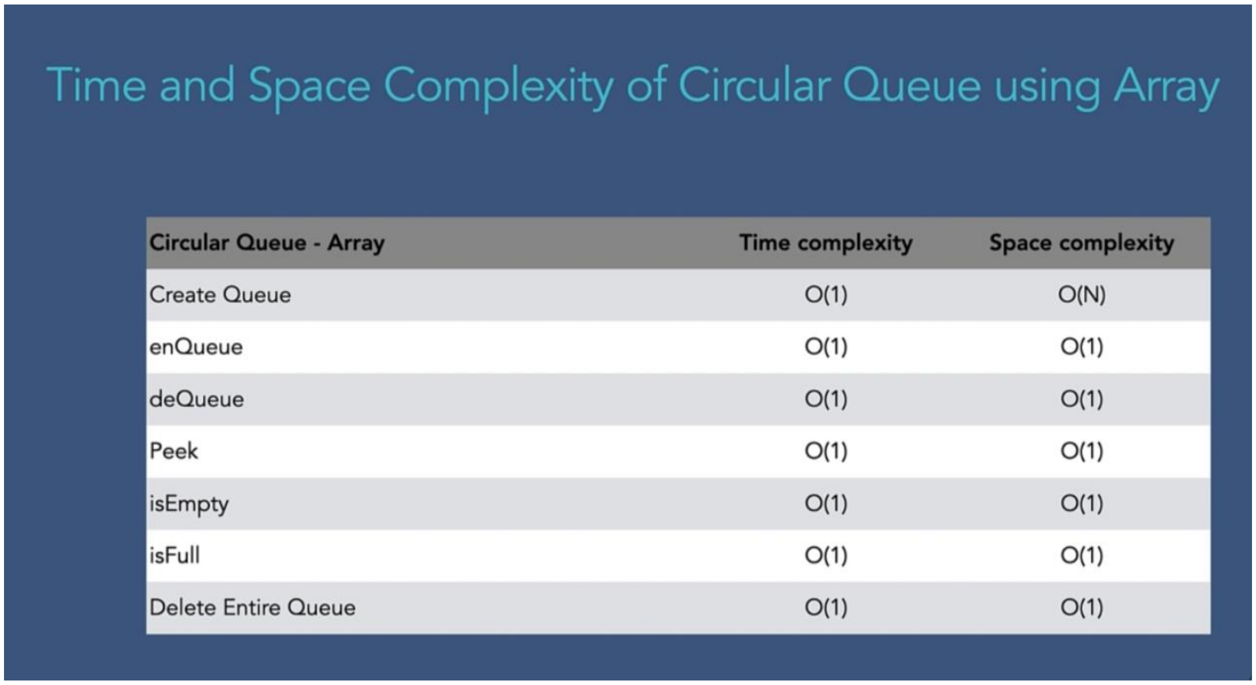


Circular Queue using Array

Delete Method

Circular Queue will get deleted

```
newQueue.delete()
      beginnigOfQueue = 1
      topOfQueue = 0
```

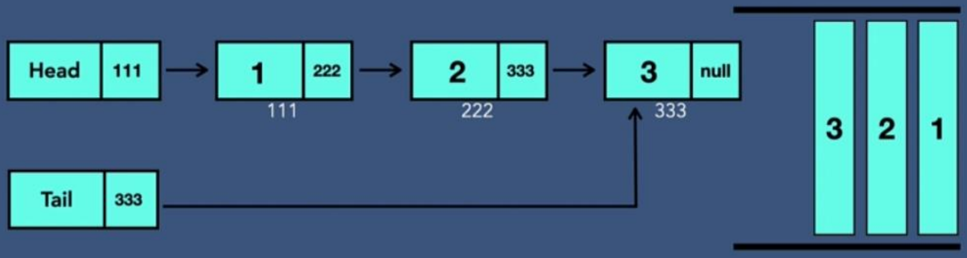


Time and Space Complexity of Circular Queue using Array

Circular Queue - Array	Time complexity	Space complexity
Create Queue	O(1)	O(N)
enQueue	O(1)	O(1)
deQueue	O(1)	O(1)
Peek	O(1)	O(1)
isEmpty	O(1)	O(1)
isFull	O(1)	O(1)
Delete Entire Queue	O(1)	O(1)

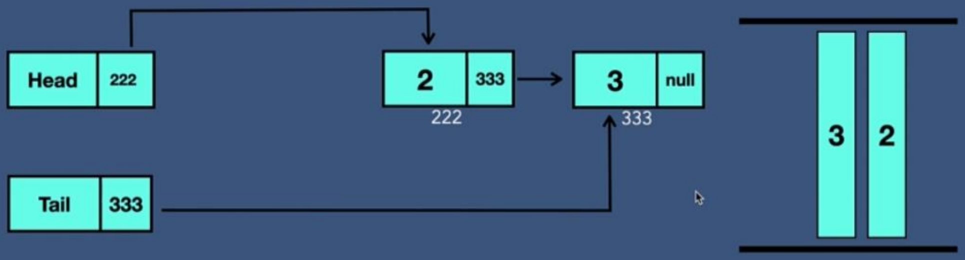
Queue using Linked List

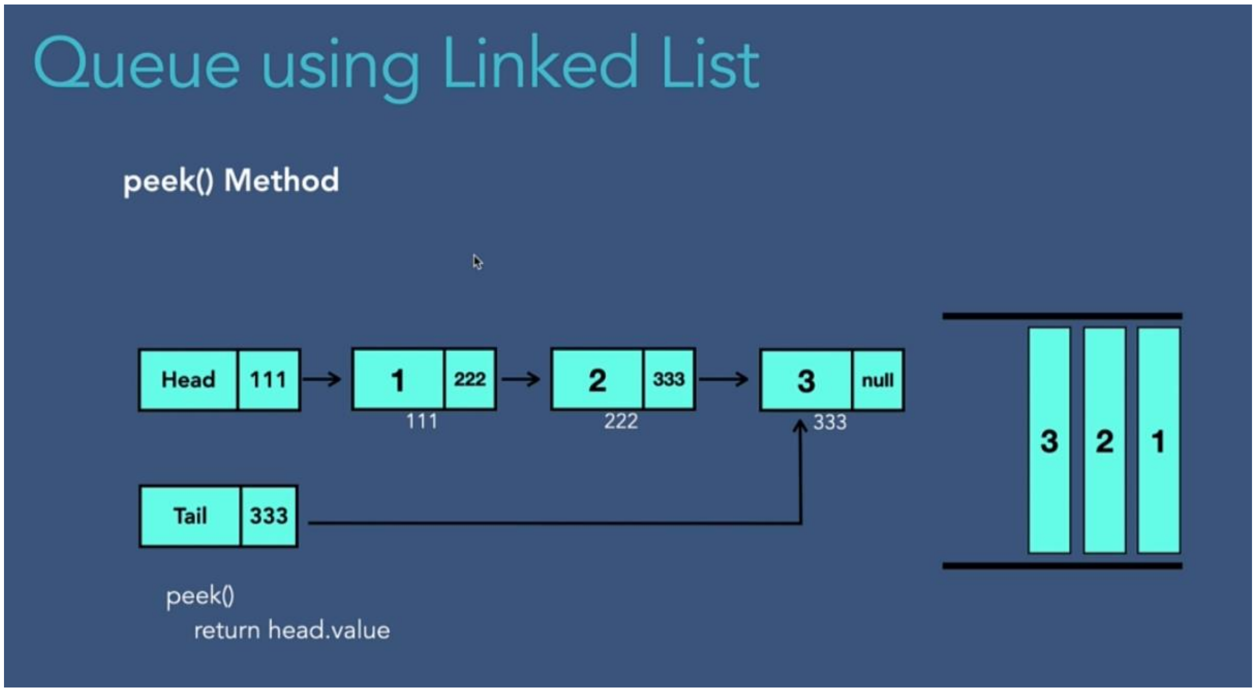
enQueue() Method



Queue using Linked List

deQueue() Method

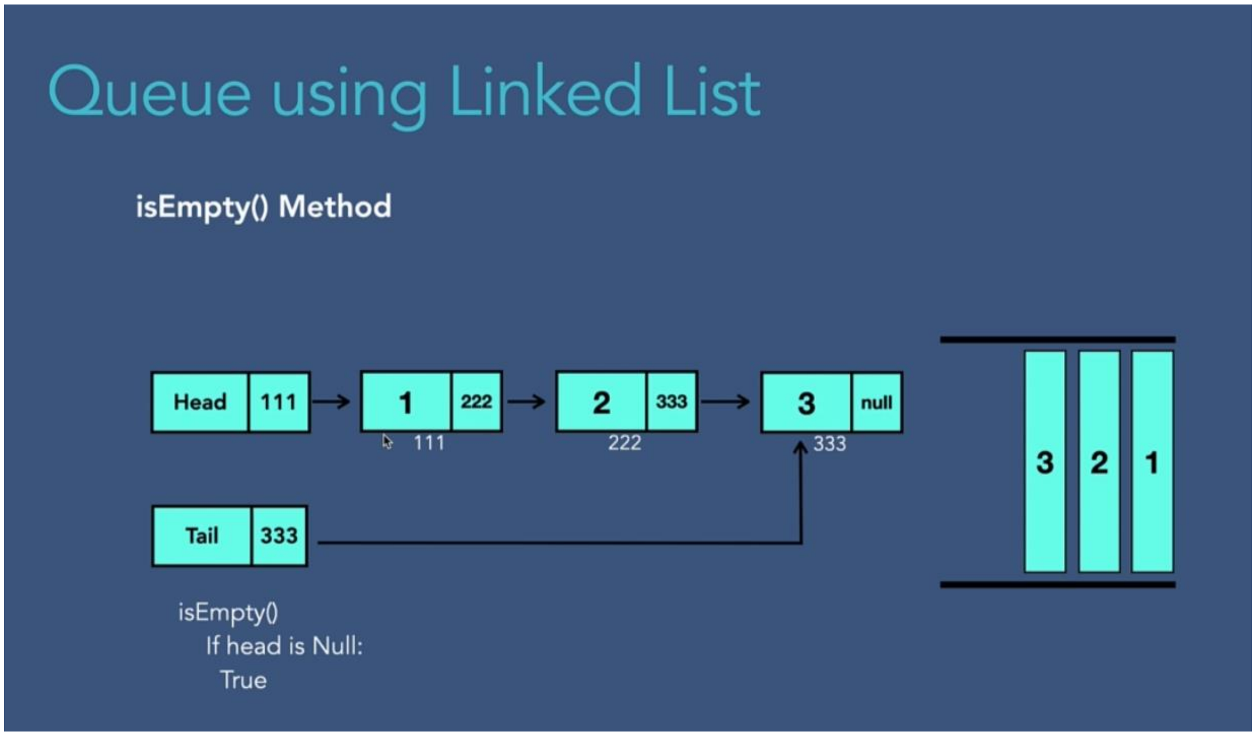




Queue using Linked List

peek() Method

```
peek()
    return head.value
```

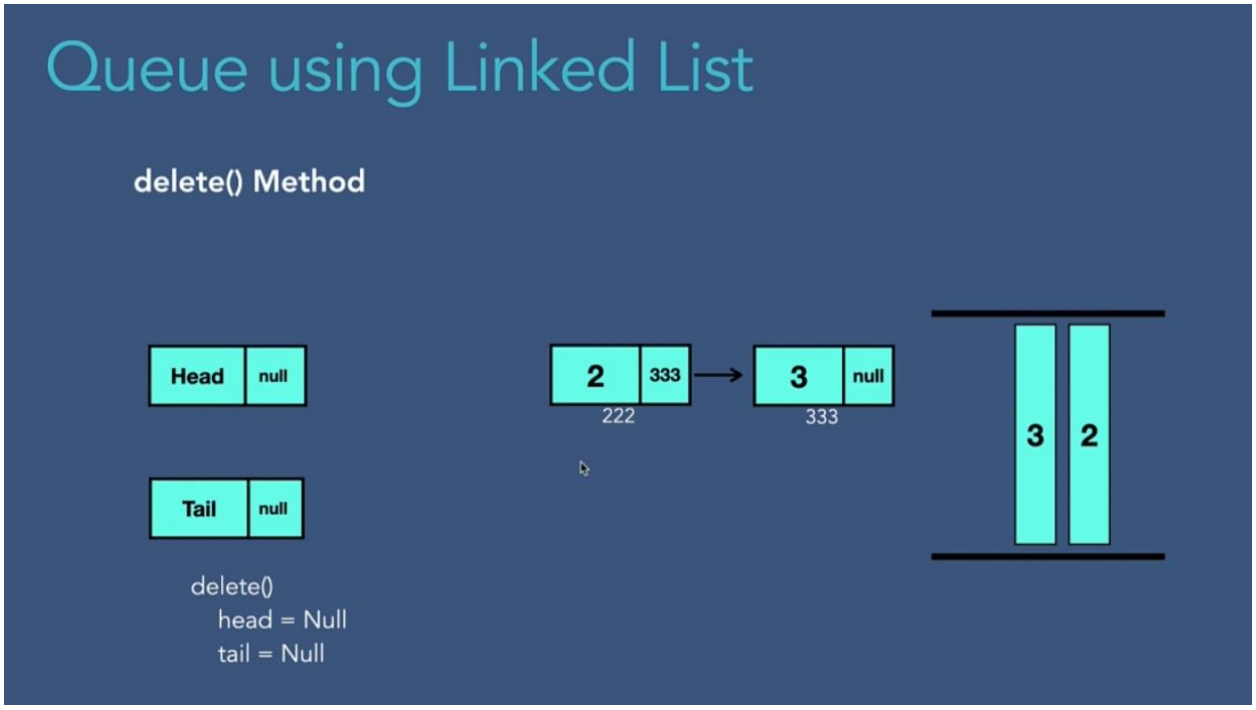


Queue using Linked List

isEmpty() Method

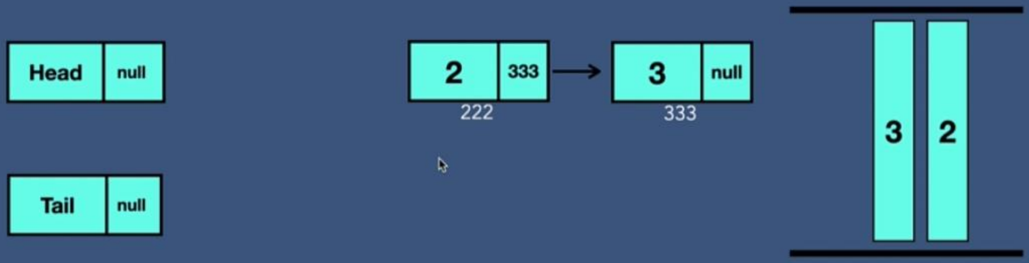
```
isEmpty()
    If head is Null:
        True
```

IsFull() Method is not available in Queue using Linked List as Linked List is dynamic in nature and memory is allocated for new nodes inserted.

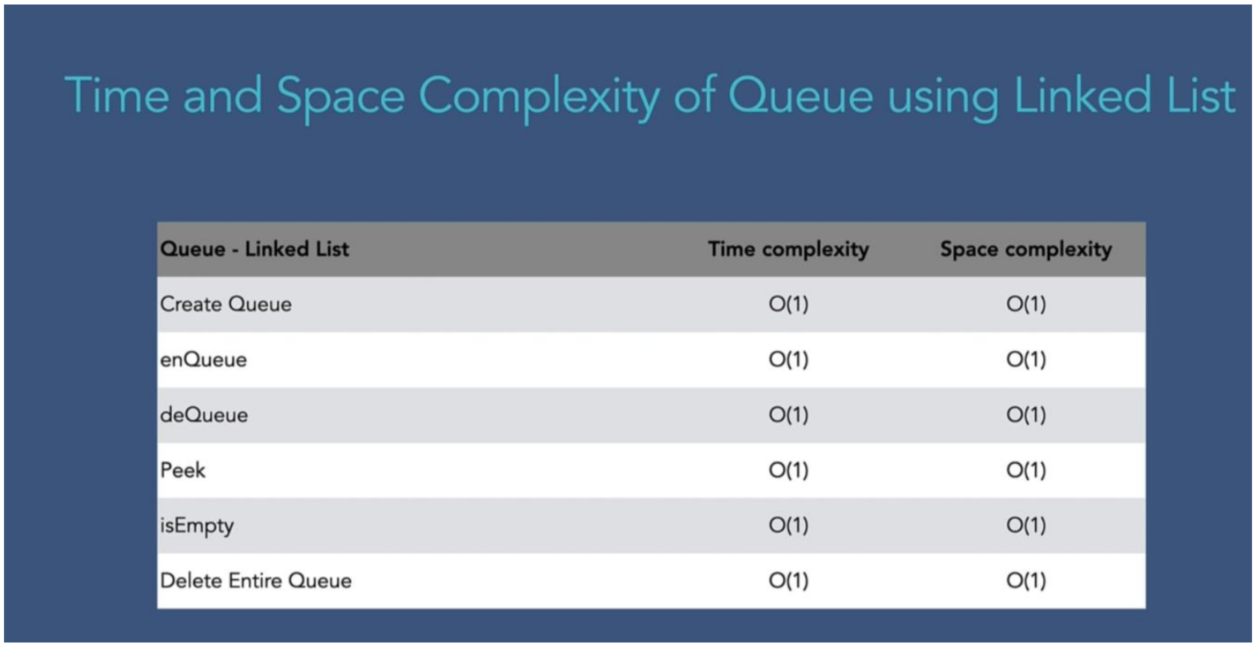


Queue using Linked List

delete() Method

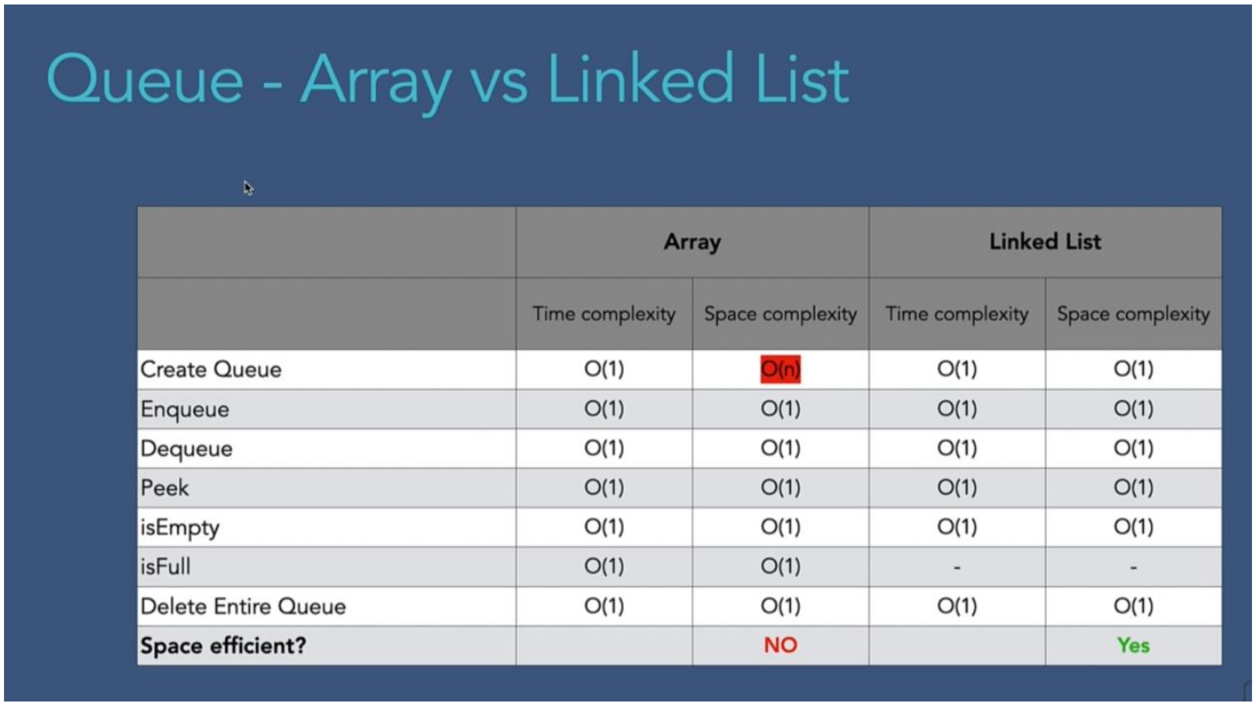


```
delete()
    head = Null
    tail = Null
```



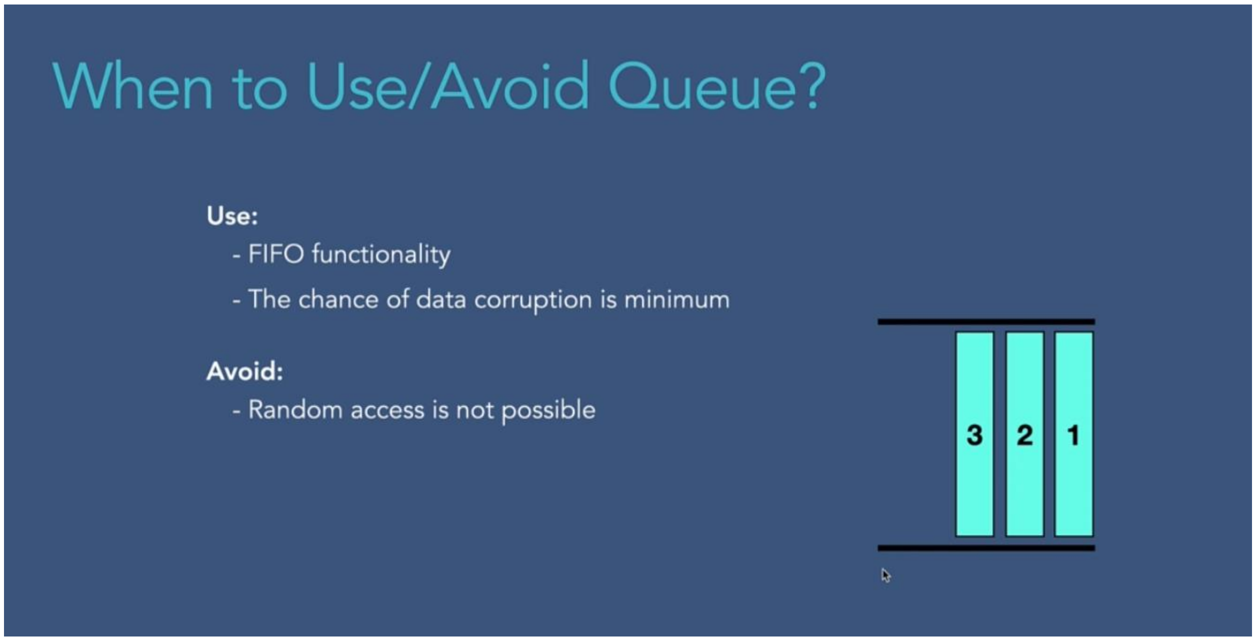
Time and Space Complexity of Queue using Linked List

Queue - Linked List	Time complexity	Space complexity
Create Queue	O(1)	O(1)
enQueue	O(1)	O(1)
deQueue	O(1)	O(1)
Peek	O(1)	O(1)
isEmpty	O(1)	O(1)
Delete Entire Queue	O(1)	O(1)



Queue - Array vs Linked List

	Array		Linked List	
	Time complexity	Space complexity	Time complexity	Space complexity
Create Queue	O(1)	O(n)	O(1)	O(1)
Enqueue	O(1)	O(1)	O(1)	O(1)
Dequeue	O(1)	O(1)	O(1)	O(1)
Peek	O(1)	O(1)	O(1)	O(1)
isEmpty	O(1)	O(1)	O(1)	O(1)
isFull	O(1)	O(1)	-	-
Delete Entire Queue	O(1)	O(1)	O(1)	O(1)
Space efficient?		NO		Yes



When to Use/Avoid Queue?

Use:

- FIFO functionality
- The chance of data corruption is minimum

Avoid:

- Random access is not possible