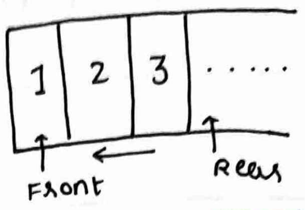
Queue

→ Queue is a linear Data structure base on FIFO (First in First out).

→ It has front and rear pointer to track first and last element.

→ Deque is double ended Queue.



★ In Java, Queue is an interface and not a class so we don't directly implement Queue. (Same like List)

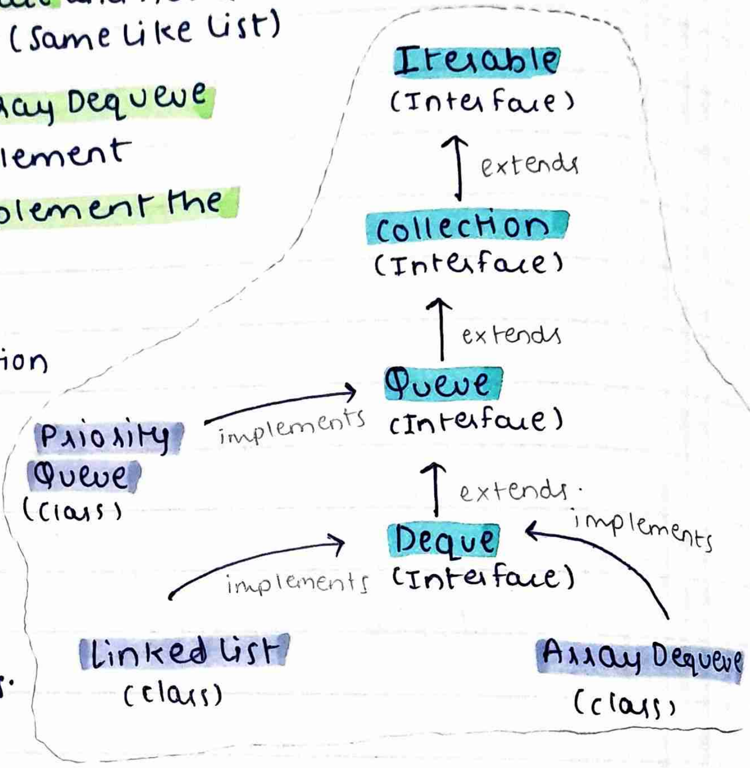
★ We use Linked List and Array Dequeue (these are classes) to implement Queue. These classes implement the Queue interface.

★ Choose LinkedList for implementation when:-

1. we need dynamic size
2. Efficient front and rear operations.

★ Choose ArrayDequeue when:-

1. Queue has fixed size.
2. We need alot of random access.
3. Lower memory usage.



Queue Syntax:-

```
Queue <Datatype> queue = new LinkedList<>();
Queue <Datatype> queue = new ArrayDequeue<>();
```

Operations performed on Queue.

Enqueue	O(1)	Adds element to rear of the Queue.
Dequeue	O(1)	Removes element from the front.
Peek	O(1)	Returns the front element without removing it.
Empty	O(1)	checks if queue is empty
Full	O(1)	Checks if Queue is Full

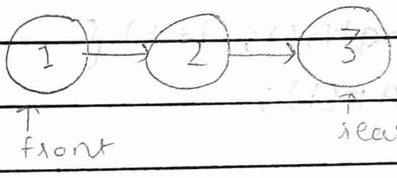
PAGE No.
DATE / /

Implementation of Queue:- (codes for implementation are done)

1. Using Array:-
→ fixed size. "n"
→ remove O(n)
→ we can implement Circular Queue ✓
↳ Circular Queue.

add O(1) (Rear + 1 = front) → full
peek O(1) Rear = -1
remove O(1) front = -1

2. Using Linked List :-
→ efficient addition and removal



★ Points to remember:-
→ The collection framework functions of Queue depend on both, what do they do and what is their return type.

→ Priority Queue and Linkedlist are not thread safe implementations
Priority Blocking Queue is one alternative implementation if thread safe implementation is needed.