Introduction to Linear Data Structures :→ (Reverse)

* Arrays → 1D, 2D, MD Arrays
* Stacks → Last In First Out or First In Last Out
* Queues → First In First Out or Last In Last Out
* Linked Lists → Singly, Doubly, Circular Lists

Stacks :→ (LIFO) (FILO) ↓ top--

Important methods:
* empty( ) ✓
* full( ) ✓
* push( ) ✓
* pop( ) ✓
* size( ) ✓
* display( ) ✓
* peek( )

[array]

top top→ top++ top-- MS

x				y	x
0	1	2	3	4	5

-1 top (top) top++

arr[top] = elem;

→ MAX_SIZE = 100
↳ Stack overflow

top

2
8
9

Stack

→ 2 pop()
→ top

top = -1

empty

x	y	z
0	1	2

top 3

size = top + 1
= -1 + 1
= 0

3 + 1
= 4

return top + 1;

templates → C++ <T>

[Generics ?]

<T>

* Create a stack that can accept any data type. (int, char, boolean, float)
  * data type doesn't matter.

1 → 2 → 2 → 1 → null Palindrome

t1 → 1 → 2 → 2 → 1 → true

1 → 2 → 3 → 4 → null Reverse

t1 → → → →

head empty
insert head
4 → 3 → 2 → 1 → null

Stack

4
3
2
1

top

1
2
2
1

* Reverse

Queue Data Structure :→ First In First Out
Last In Last Out

* Trees & Graphs
* (BFS Traversal)
* (Level Order Traversal)

Same ←

additions

f f
-1 | z |
0 0

MAX_SIZE = 100

front -1 f

x	y	z	a	
0	1	2	3	4
	r front		rear	

rear -1
empty

0 ✗

1st case empty
front = rear = -1

2nd case single element
front = rear = 0

enqueue or push

Adding: rear++
arr[rear] = element;

removals

dequeue or pop

Removal: front++ (0th element is removed)

java.util.Stack → class → object
java.util.Queue → Interface → object ✗

LinkedList ArrayList

Important interview question for Cognizant | IBM | TCS | Wipro :→

* Given an integer value N, generate binary numbers from 1 to N in the given format:
  Say N = 5 O/P: 1, 10, 11, 100, 101

Constraints: (i) Output should be exactly as shown.
* (ii) Use the Queue Data Structure.

Initial queue<string> = ["1"]

Step1: Pop "1" Print "1" Push "10" & "11" ["10", "11"] (front)

Step2: Pop "10" Print "10" Push "100" & "101" ["11", "100", "101"]

Step3: Pop "11" Print "11" Push "110" & "111" ["100", "101", "110", "111"]

Step4: Pop "100" Print "100" Push "1000" & "1001" ["101", "110", "111", "1000", "1001"]

Step5: Pop "101" Print "101" Push "1010" & "1011" ["110", "111", "1000", "1001", "1010", "1011"]

(N=5)