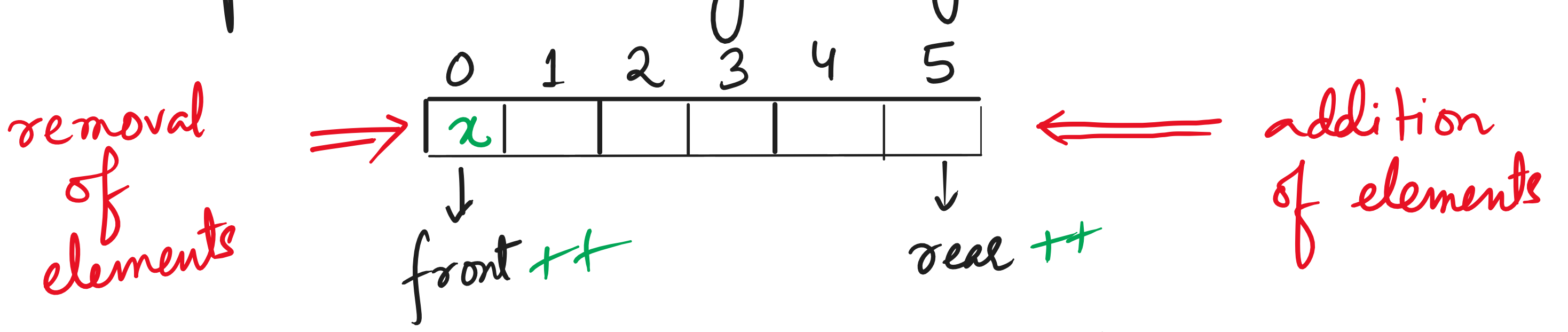
The Queue Data Structure : {First In First Out}

Implementation using array :→ Two indexes → front & rear



* Can take a var (MAX_SIZE) to control capacity.

Cases : → Empty Queue → front = rear = −1.

Single Element → front = rear = 0.

→ dequeue

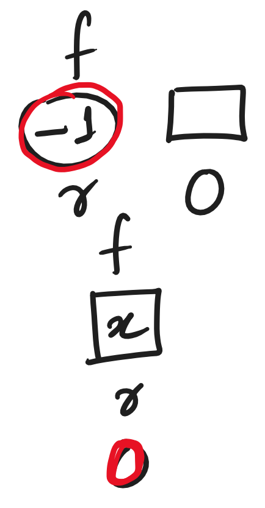
→ enqueue

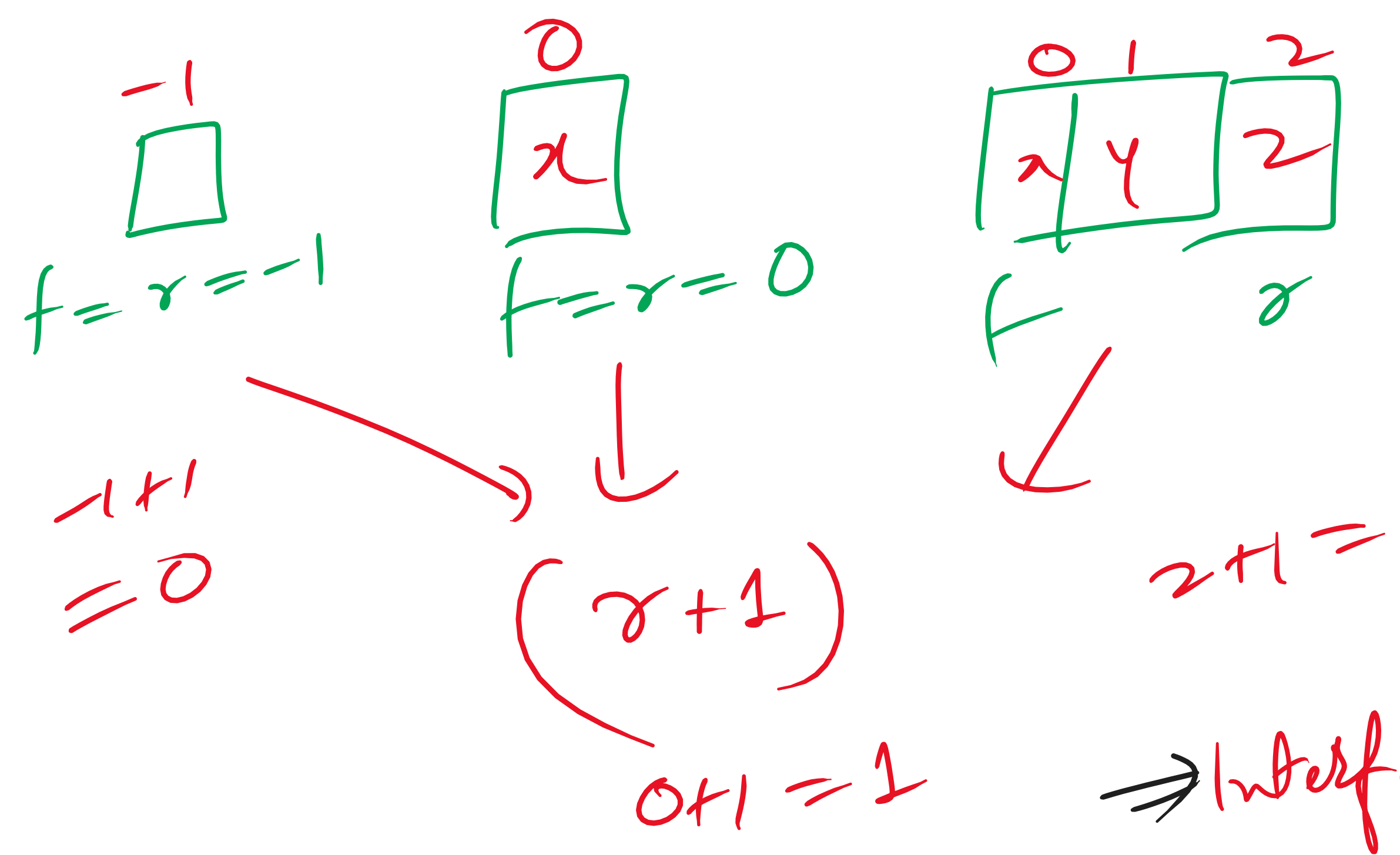
Remove → front++

Add → rear++

arr[front] → FE

arr[rear] → RE





Can we create object of an interface?

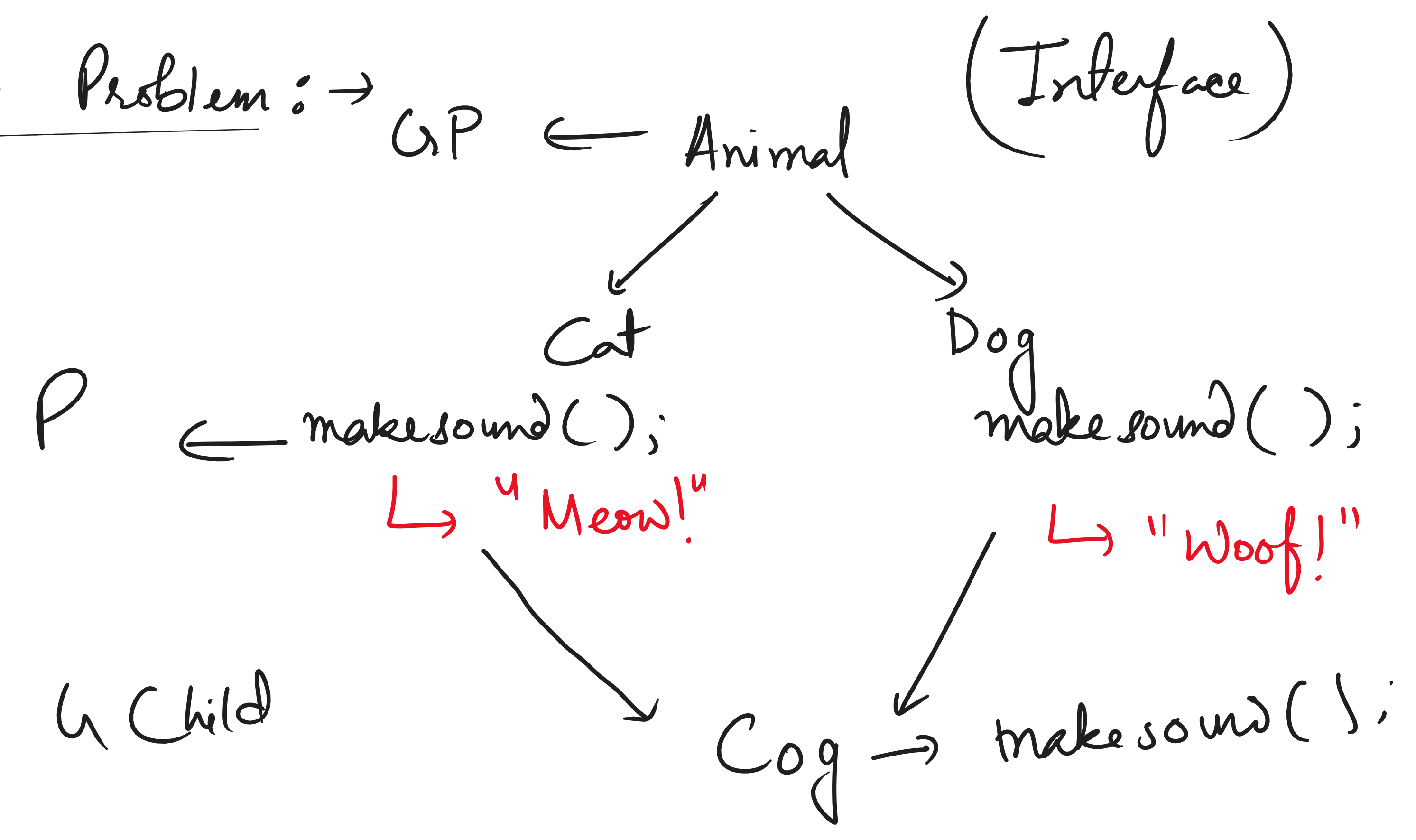
NO :

⇒ Interface 100% abstraction

Class (0 − 100%)

Can we extend multiple classes in Java ? (NO)

Diamond Problem :→



* Generate Binary from 1 to N using a Queue.

(N = 5) → 1, 10, 11, 100, 101

* Constraints : The data need not be integer only.
But, the output should be in this given format.

"1", "10", "11", "100" "101"

stop

Input N = 5

N=3

Queue → ["1"]

While (N-- > 0)

Step 1 : Pop "1" Print → Push "10", "11" → Queue [~~"10"~~, "11"]

Step 2 : Pop "10" Print → Push "100", "101" → Queue [~~"11"~~, "100", "101"]

Step 3 : Pop "11" Print → Push "110", "111" →

Step 4 : Pop "100" Print Push 1000, 1001 [~~100~~, 101, 110, 111]

Step 5 : Pop "101" Print Push 1010, 1011

front

Collections Framework In Java :

Package : java.util

[Built-in Data Structures in Java]

① List (Interface)
- ↳ ArrayList
- ↳ LinkedList
- ↳ Vector → Don't use
- ↳ Stack (extends Vector)

② Set (Interface) (enum)
- ↳ HashSet → (Hashing)
- ↳ LinkedHashSet
- ↳ TreeSet

③ Queue (Interface) / (DeQue)
- ✓ ↳ PriorityQueue (Heap)
- ✓ ↳ LinkedList
- ✓ ↳ ArrayDeque (implements Deque)

(Binary Tree, BST, Heaps)
Trie

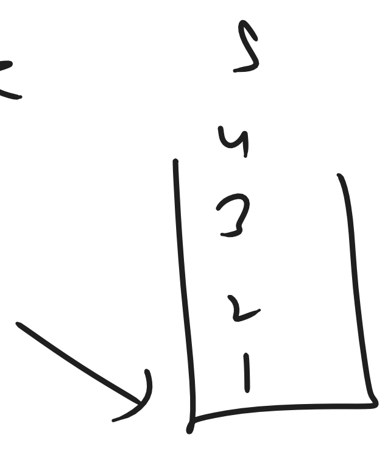
④ Map <K,V> (Interface)
- ↳ HashMap
- ↳ LinkedHashMap
- ↳ TreeMap
- ✓ ↳ HashTable ****
- ↳ Properties (extend HT)

* Generic Queue<T>

* Queue → reversal → Stack

Q 1,2,3,4,5

O/p 5,4,3,2,1



Feedback : 14107

url → [bigoficttraining.com]