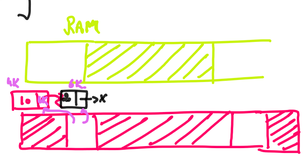
Lecture 15

Sunday, 23 February 2020 2:31 PM

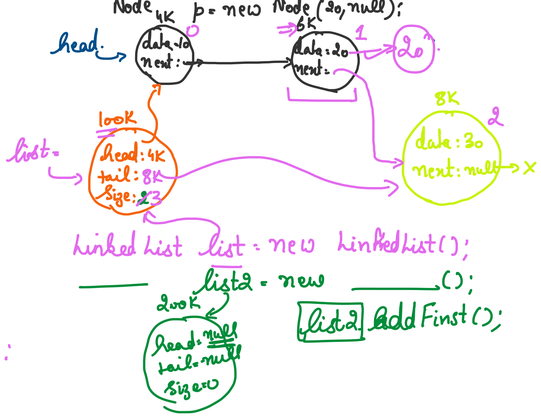
Linked List → why? } Arrays, Stacks, Queues, L.L } → Linear data structures.

Diff b/w Array & L.L ?




```
class L.L {
private class Node {
        int data;
        Node next;
        Node(int data, Node next) {
            this.data = data;
            this.next = next;
        }
    }
    Node head;
    Node tail;
    int size;
```

↳ `Node n = new Node(10, null);`
`Node p = new Node(20, null);`

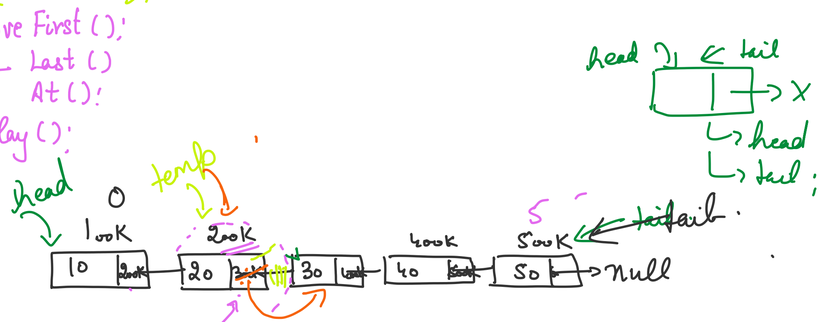


`LinkedList list = new LinkedList();`

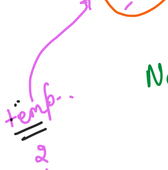
`_____ list2 = new _____();`

`list2.addFirst();`

`size();`
`isEmpty();`
`addFirst(data);`
`addLast(data);`
`addAt(int index, int data);`
`getNodeAt(int index);`
`getFirst();`
`getLast();`
`getAt(int index);`
`removeFirst();`
`removeLast();`
`removeAt();`
`display();`



addFirst →
Node →
↳ head.
size++




```
getNodeAt(int index) {
    int counter = 0;
    Node temp = head;
    while (counter < index) {
        temp = temp.next;
        counter++;
    }
    return temp;
}
```

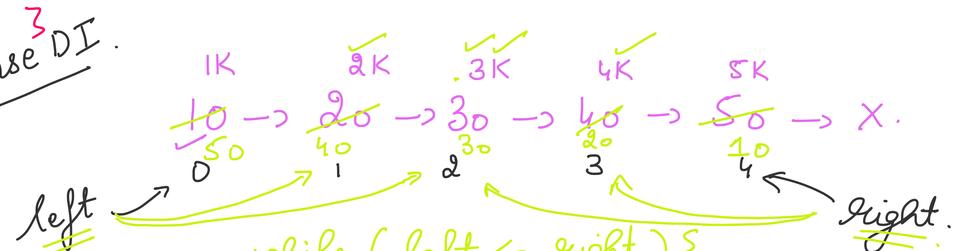
`Node n = new Node(data, this.head);` (this.head = null)
`this.head = n;`
`if (this.isEmpty()) {`
`    this.tail = n;`
`    this.head = n;`
`} else { tail.next = n;`
`    this.tail = n; }`
`size++;`

`getAt(int index) {`
`    getNodeAt(index - 1);`
`    Node _ .`

`getFirst();`
`getLast();`
`getAt();`

`removeFirst();`
`removeLast();`
`removeAt();`

Reverse DI. (3)

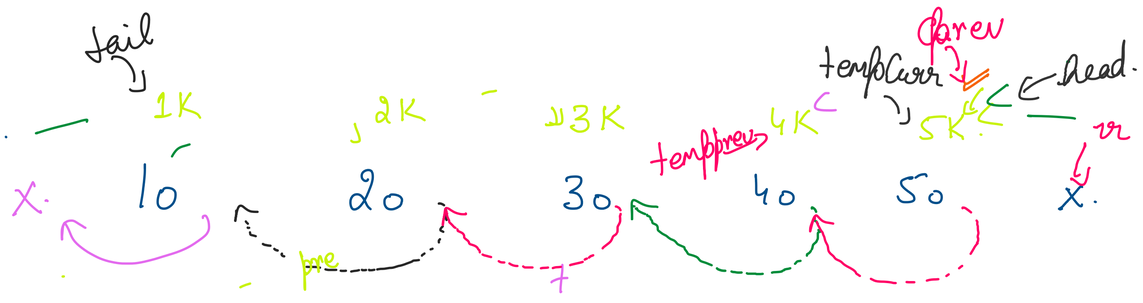



```
while (left <= right) {
    Node l = this.getNodeAt(left);
    Node r = _____(right);
    int temp = l.data;
    l.data = r.data
    r.data = temp;
    left++;
    right--;
}
```

n

3

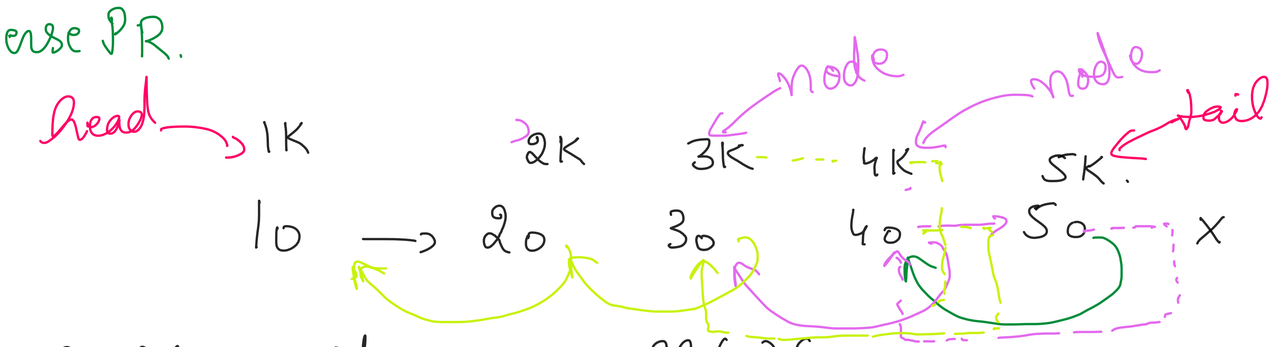
Reverse PI.




```
Node prev = this.head;
Node curr = prev.next;
while (curr != null) {
    Node tempPrev = prev;
    Node tempCurr = curr;
    prev = curr;
    curr = curr.next;
    tempCurr.next = tempPrev;
}
Node temp = this.head;
this.head = this.tail;
```

```
this.tail = temp;
this.tail.next = null;
```

Reverse PR.




```
public void reversePR(){
    this.reversePR(this.head);
=> { swap head and tail
    tail.next=null;
}
```

```
public void reversePR(Node node){
    {if(node == this.tail){
        return;
    }
    reversePR(node.next);
    node.next.next = node;
}
```

3K ↑ 2K ↑ 1K

node.

Q=> ReverseDR