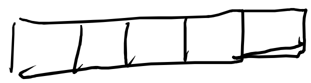
2.12 - Linked List - 1 } → Data Structure

Sunday, November 2, 2025 4:53 PM

#

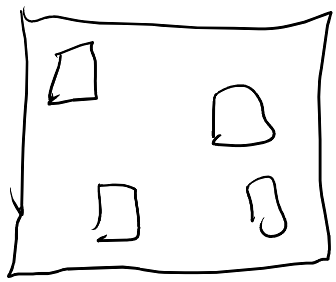
int arr[] = new int[10]

4

Array list →  Dynamic size

continuous array.

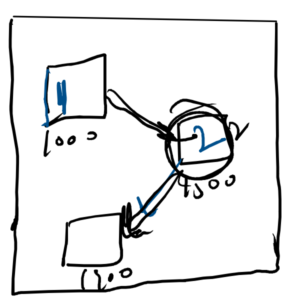
10 ⇒ 40 Bytes



[40 Bytes]

→ Linked list solves the problem of fragmented memory.

3x4=12



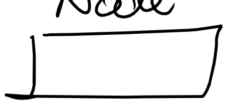
[1, 2, 3]

linked list

Node

⇒ Node: Basic Building block of a linked list

Node



class{
int data;

int, Str, P, 2

int data;
[Node next];

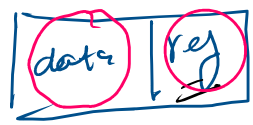
Person p = new Person()

ref obj

}

class Node {
int data;
String
[Node next]

}

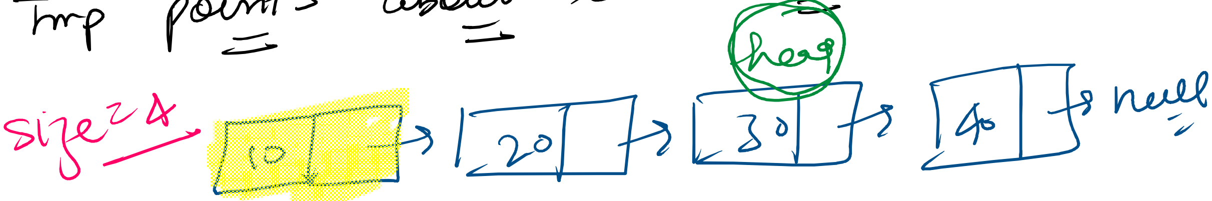


Node

Person p = new Person()

reference obj

Imp points about linked list

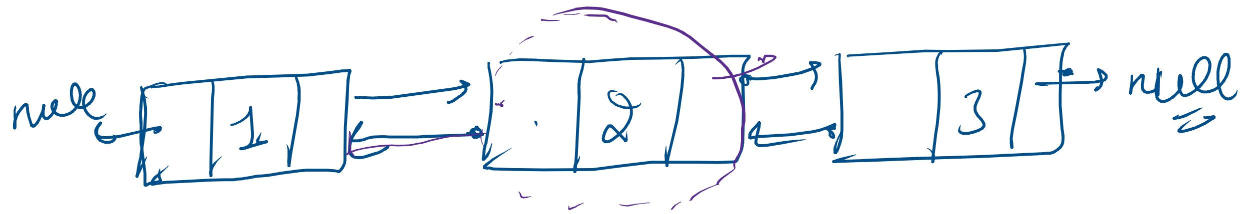


- size of linked list = no. of Nodes in the LL.

- first Node is called head of the LL.

- We can only access a linked list via the head node.

- A LL in which we can only move in a single direction is called a singly LL.



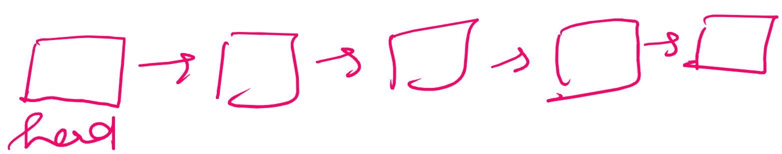
→ A linked list in which we can move in both the directions is called a doubly LL

```
class Node {
    data
    Node next;
    Node prev;

}
```

Strategy to solve linked list questions:

1) Always draw a generally LL diagram of 4-5 nodes.



2) Dry run the movement of the head pointer.

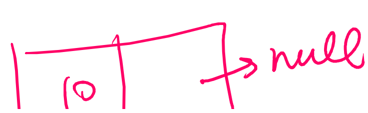
3) Think of two edge cases

→ if list is empty

```
if ( head == null );
```

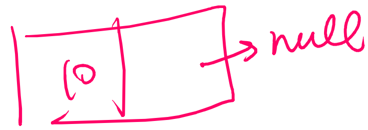
→ if list has only one node

```
if ( head.next == null ) {
```

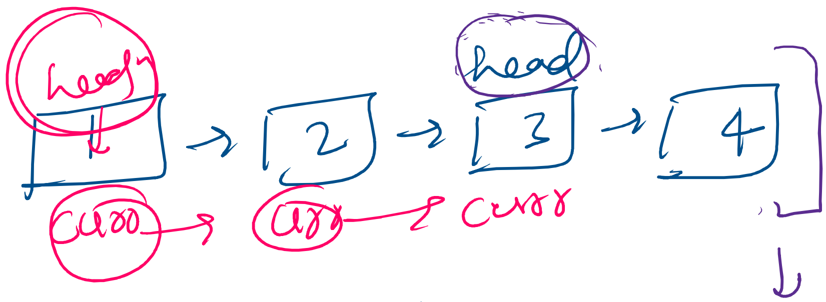


→ If [illegible]

```
if (head.next == null) {
    //
}
```



① V. Imp



→ Generally we don't prefer to move head because it is the only way to access our LL.

→ We make a copy head 'curr' and move it.