notes

Doubly linked list

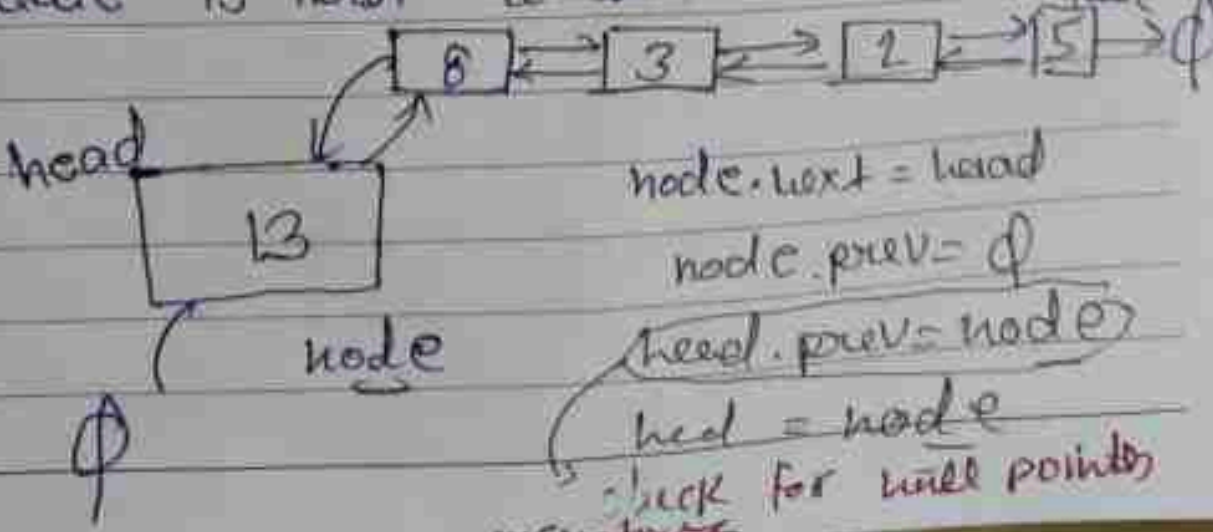
Every single node has a next and a previous

Every single node every single box will have the same structure

```
class Node {
    int val;
    Node next;
    Node prev;
}
```

Create a node:-
when I add 13 at the front of linked list

Create 13 first created a new node:



code :-

```
public class DLL {
    private Node head;

    private class Node {
        int val;
        Node next;
        Node prev;

        public Node(int val) {
            this.val = val;
        }

        public Node (int val, Node next, Node prev) {
            this.val = val;
            this.next = next;
            this.prev = prev;
        }
    }
}
```

notes

insert at first

```
public void insertFirst (int val) {
    Node node = new Node(val);
    node.next = head;
    node.prev = null;
    if (head != null) {
        head.prev = node;
    }
    head = node;
}
```

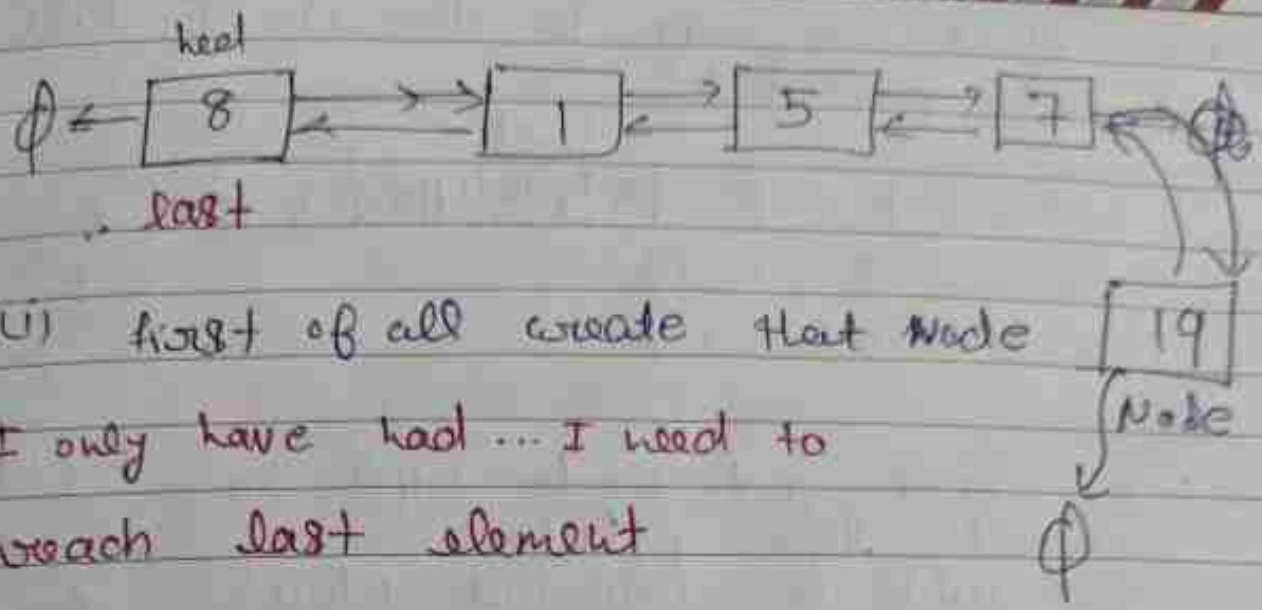
To display the linked list:-

```
public void display () {
    Node node = head;
    while (node != null) {
        system.out.print (node.val + " -> ");
        node = node.next;
    }
    System.out.println ("END");
}
```

notes

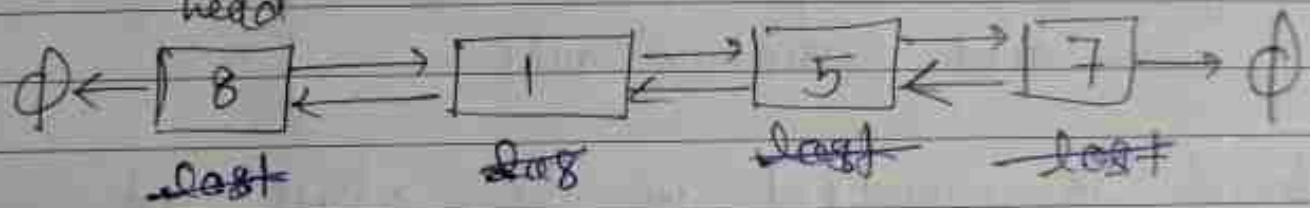
display in reverse :-

```
public void display () {
        Node node = head;
        Node last = null;

    while ( node != null) {
System.out.print ( node.val + " -> ");
        last = node;
        node = nod.next;

}
  Sout ("END");

Sout (" Print in Reverse");
   while ( last != null) {

   System.out.print ( last.val + " -> ");
    last = last.prev;
}
   Sout ("START");
}
```



(i) first of all create that Node I only have head ... I need to reach last element

last element is the one whose next element is the null.



So it will checked the Node you are currently right Now is the next element null... No if move forward.

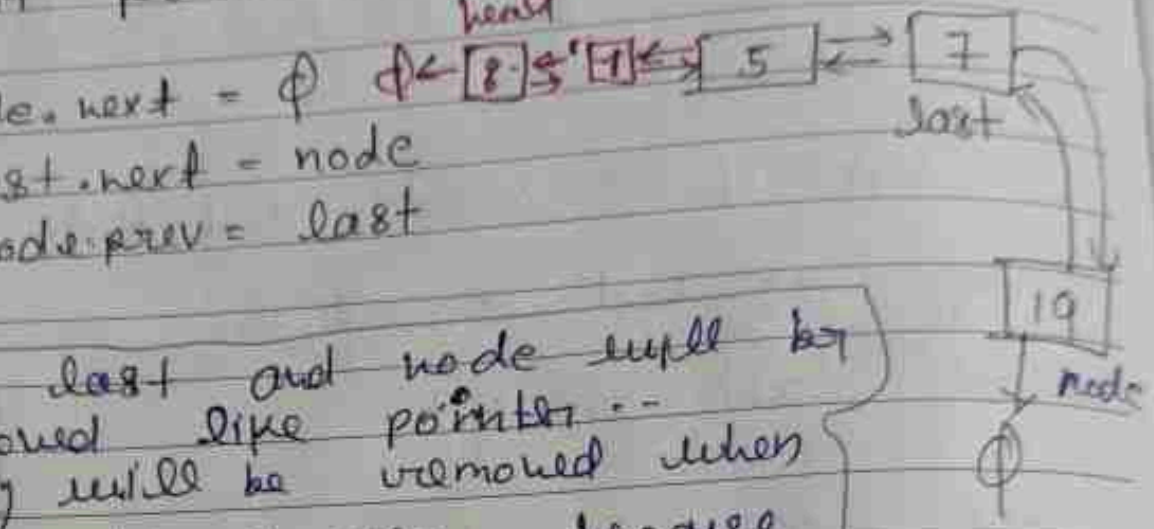
last will become (1)

checked again.. So. on

when it will check at (7) so it will stop ... you are now at the last node.

notes

Now it is the way to do it if tail is not provided.



node.next = ϕ
last.next = node
node.prev = last

So here, last and node will be removed like pointers.. they will be removed when the function is over... because they are created inside the function.. scope will be over and function over.

* head is created in the class it will stay always.. it's a part of the structure of the linked list.

Edge case

```
if (head = ϕ)
    node.prev = ϕ
    head = node.
```

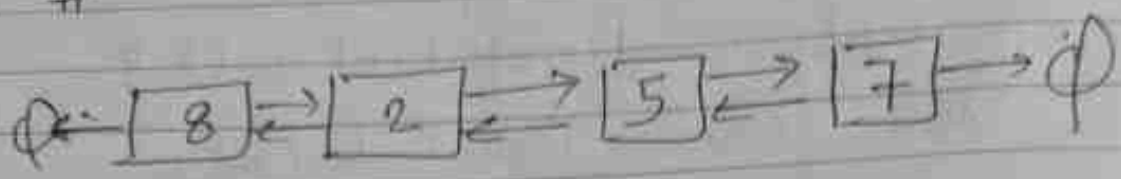
notes

code :-

```
public void insertLast (int val) {
    Node node = new node (val);
    Node last = head;

    while( last.next != null ){
        last = last.next;
    node.next = null;

if( head == null ) {
    node prev = null;
    head = node;
    return;
}
while ( last.next != null) {
    last = last.next;
}
last.next = node;
node.prev = last;
}
```

notes

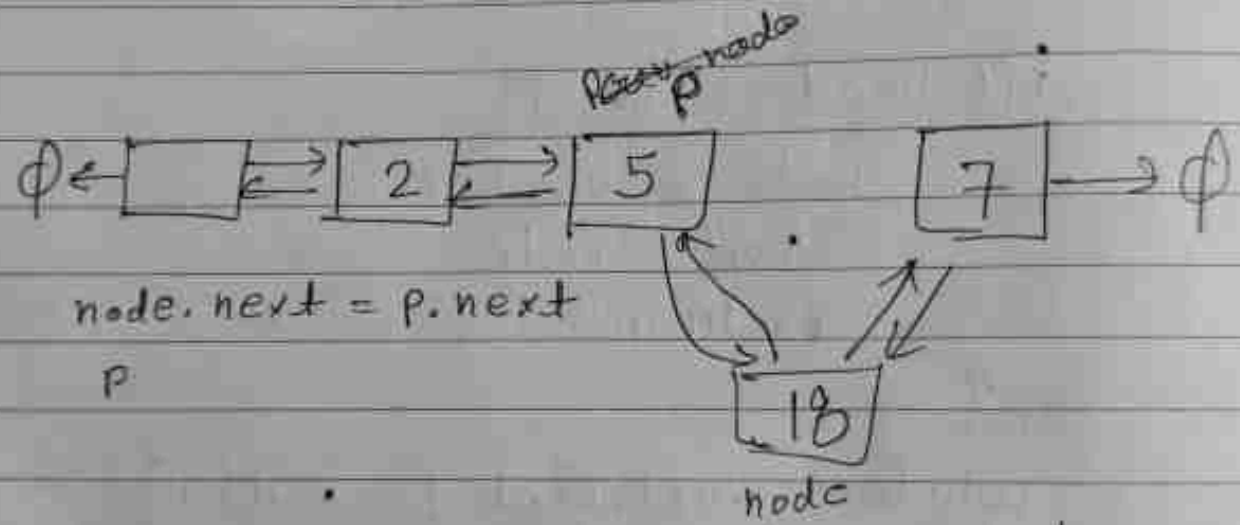
#



you wanna insert after 5.

~~after of add method~~

try to see... it will look like at the end

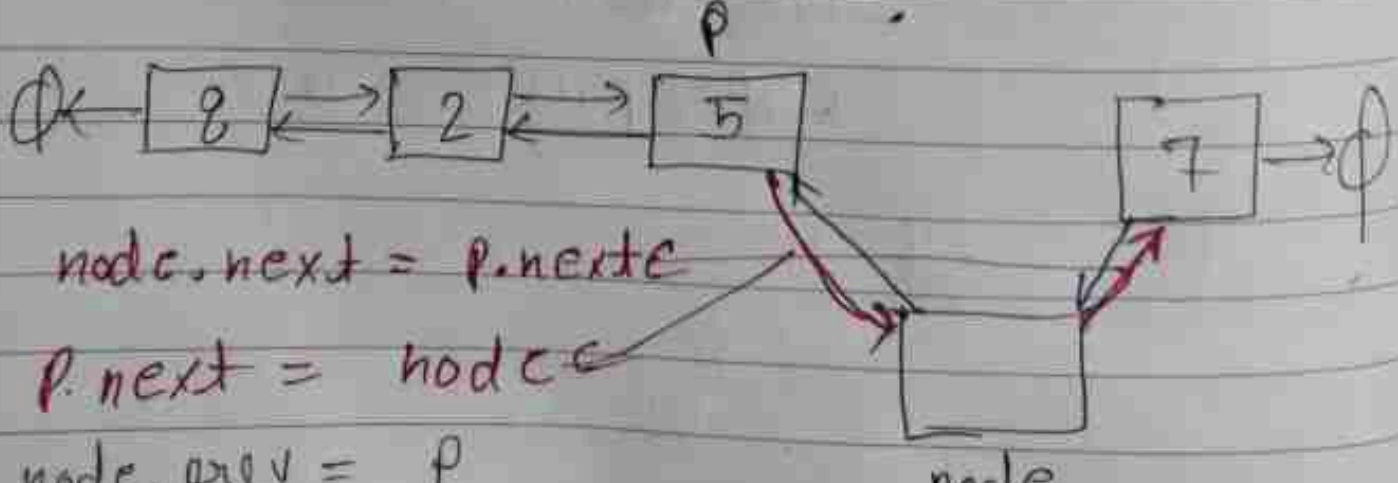


node.next = p.next

p

Now lets see how to do it.

we need the pointer to [5] and [7]



node.next = p.next

p.next = node

node.prev = p

notes

node.next.prev = node

here we need to add a check

check

node.next.prev ... going to be null pointer exception ... Yes it may.

it may be possible that [18] node

last element you are trying to insert.

in that case

node.next this may be null

Time complexity O(n)

Code:

```
public Node find(int value) {
    Node node = head;
    while (node != null) {
        if (node.val = value) {
            return node;
        }
        node = node.next;
    }
    return null;
}
```

notes

```
public void insert(int after, int val){
    Node p = find(after);
    if(p == null) {
    sout(" Does not exist");
        return;
    }
    Node node = new Node(val);
    node.next = p.next;
    p.next = node;
    node.prev = p
    node.next.prev =
    if( node.next != null ){
        node.next.prev = node;
    }
}
```