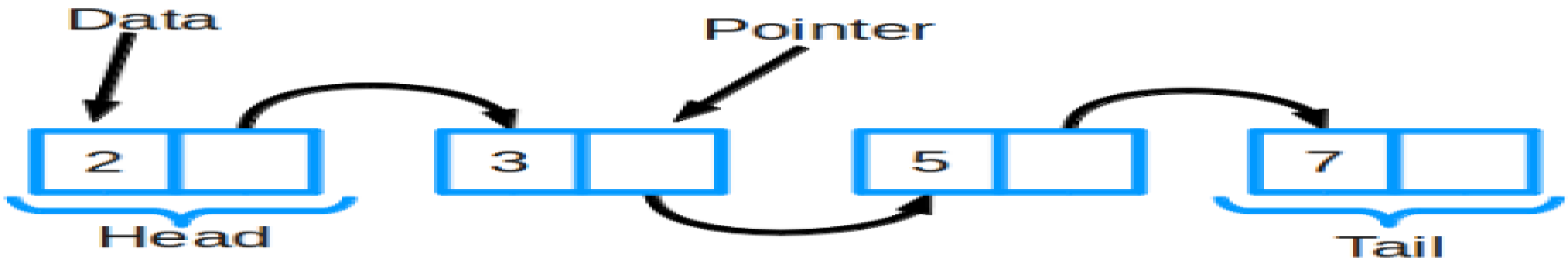
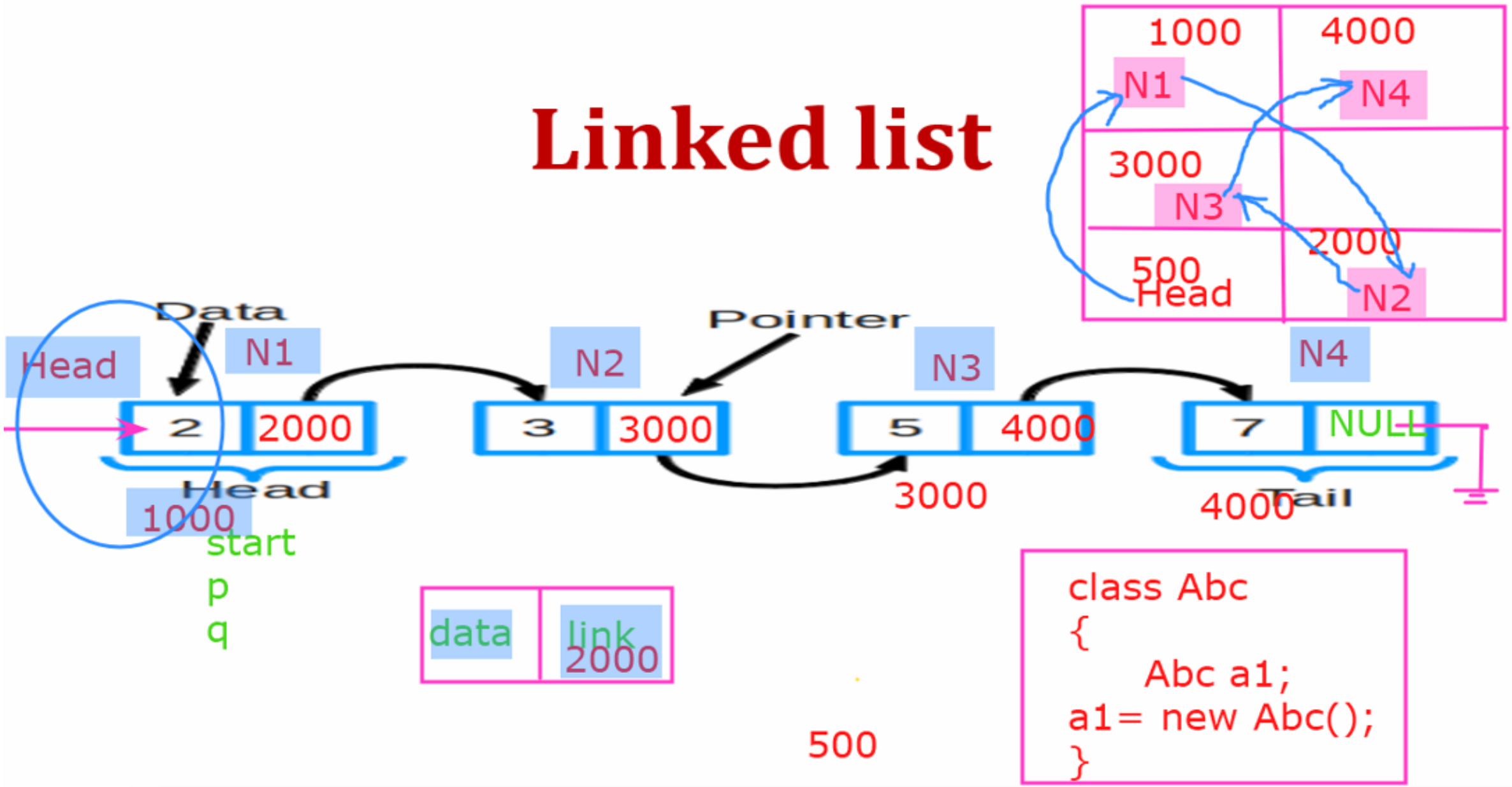
Algorithms & Data Structure

Kiran Waghmare

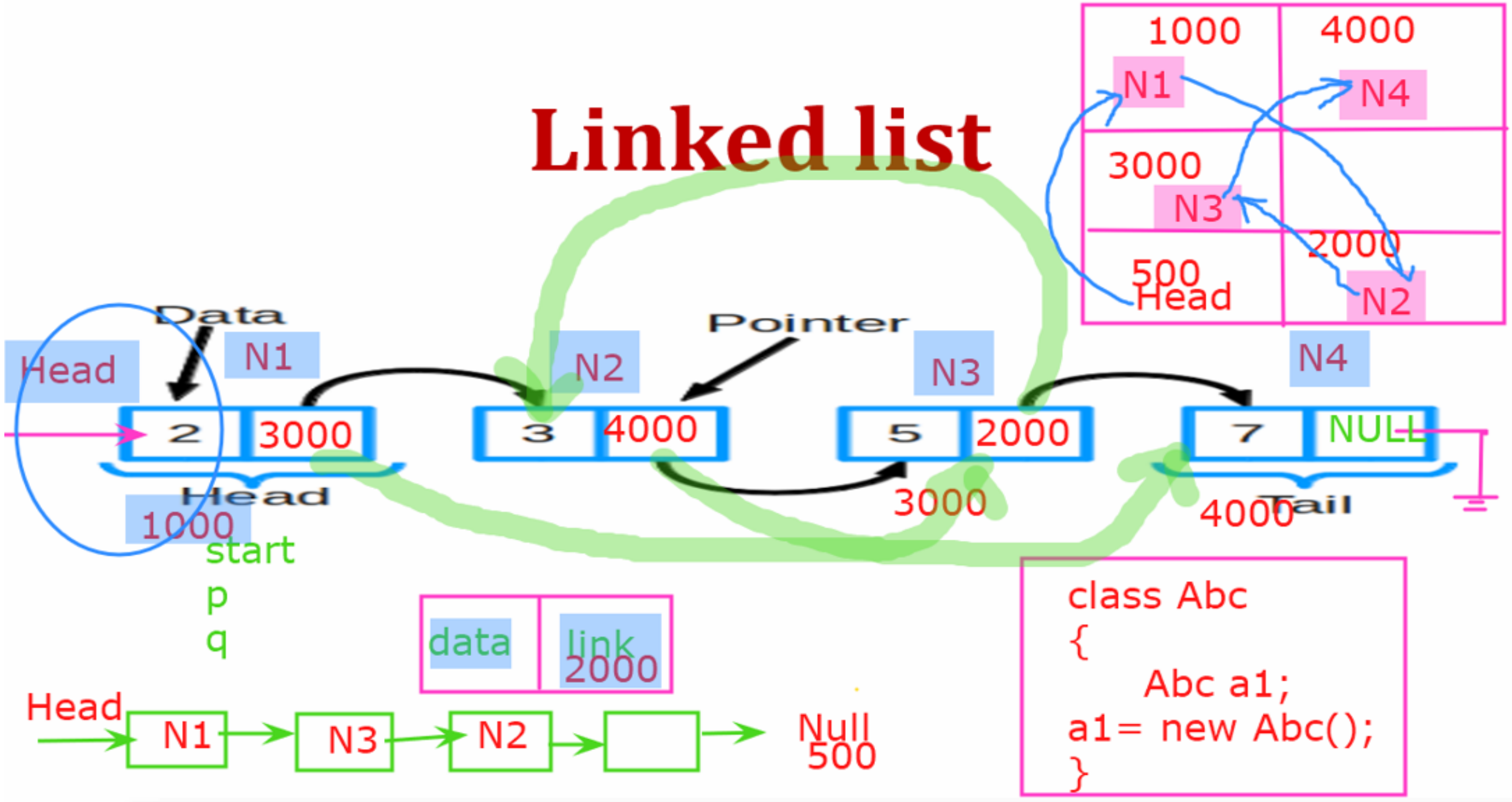
Linked list



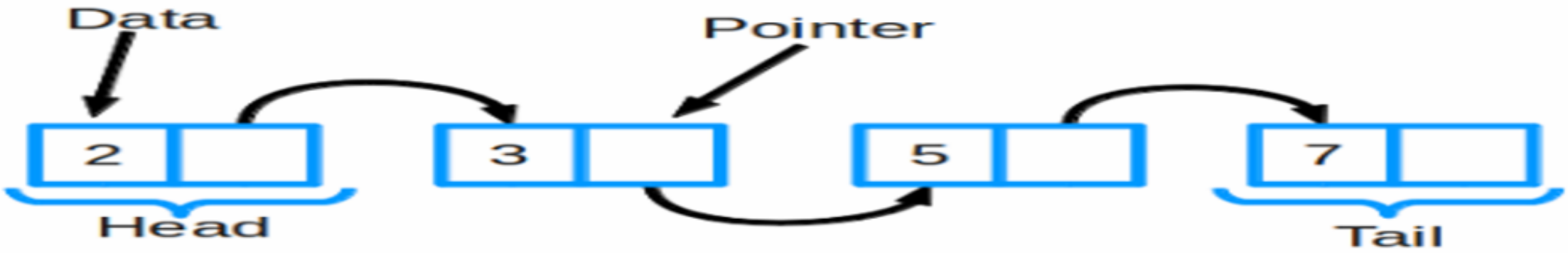
Linked list


1000
4000
N1
N4
3000
N3
2000
500
Head
N2
Data
Pointer
Head
N1
N2
N3
N4
2
2000
3
3000
5
4000
7
NULL
Head
1000
3000
4000
Tail
start
p
q
data
link
2000
500
class Abc
{
Abc a1;
a1= new Abc();
}

Linked list


1000
4000
N1
N4
3000
N3
2000
500
Head
N2
Data
Pointer
Head
N1
N2
N3
N4
2
3000
3
4000
5
2000
7
NULL
Head
1000
3000
4000
Tail
start
p
q
data
link
2000
Head
N1
N3
N2
Null
500
class Abc
{
Abc a1;
a1= new Abc();
}

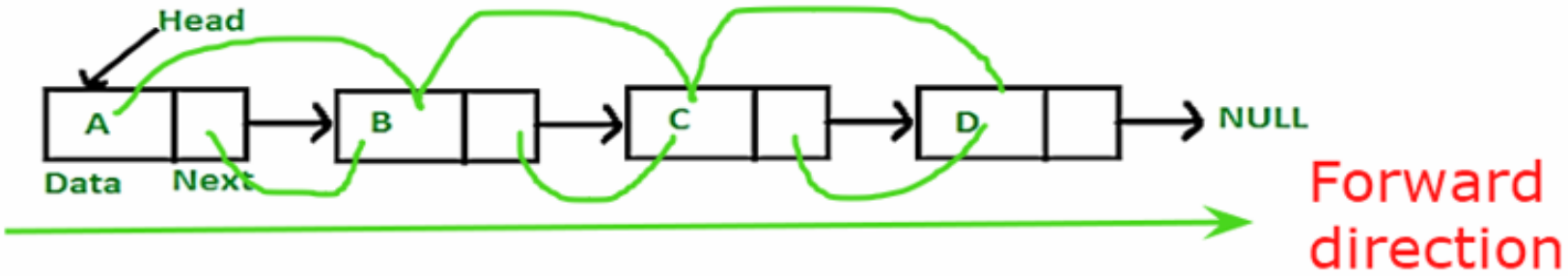
Linked list



-sequential data structure, which is connected via links
-sequence of links
-consist of 2 parts: data, link(reference of next node)
-set a pointer to indicate the start of list
-Last node link is always = NULL

Linked List Representation

- Linked list can be visualized as a chain of nodes, where every node points to the next node.



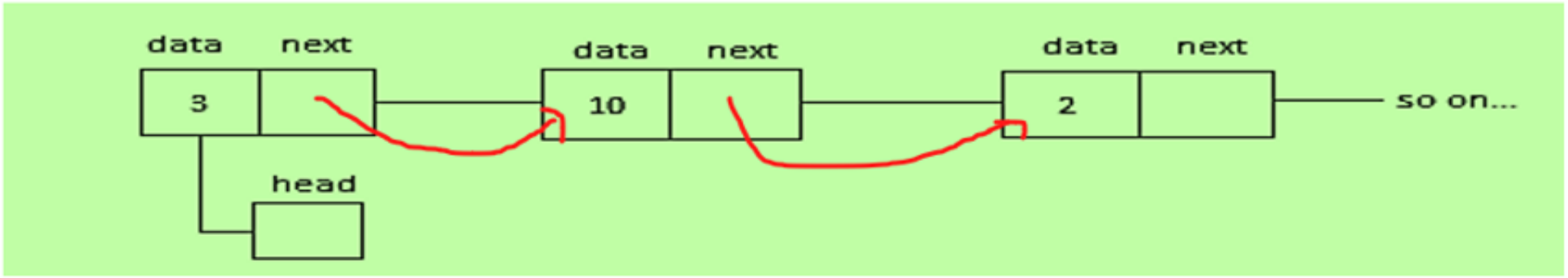
- As per the above illustration, following are the important points to be considered.

1. Linked List contains a **link element** called **first.**
2. Each link carries a **data field(s)** and a **link field** called **next.**
3. Each link is **linked with its next link** using its **next link.**
4. **Last link carries a link as null** to mark the end of the list.

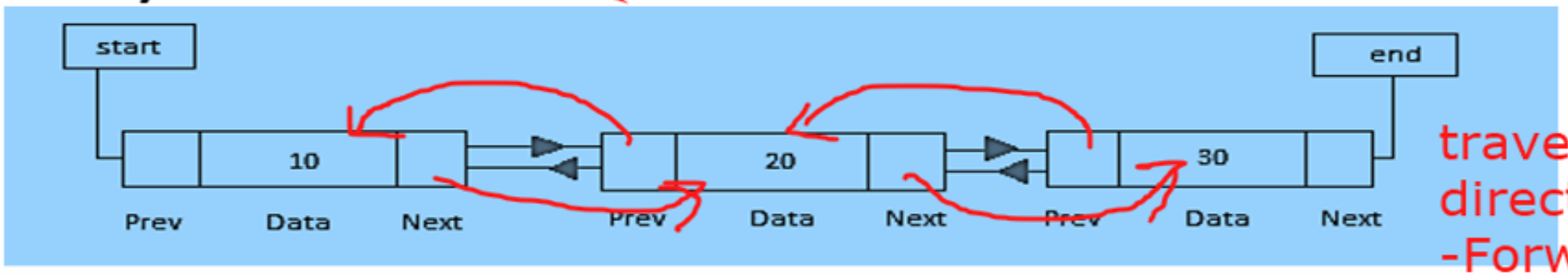
Types of Linked List

- **Following are the various types of linked list.**
  1. **Simple Linked List** – Item navigation is forward only.
  2. **Doubly Linked List** – Items can be navigated forward and backward.
  3. **Circular Linked List** – Last item contains link of the first element as next and the first element has a link to the last element as previous.

- **Simple Linked List**

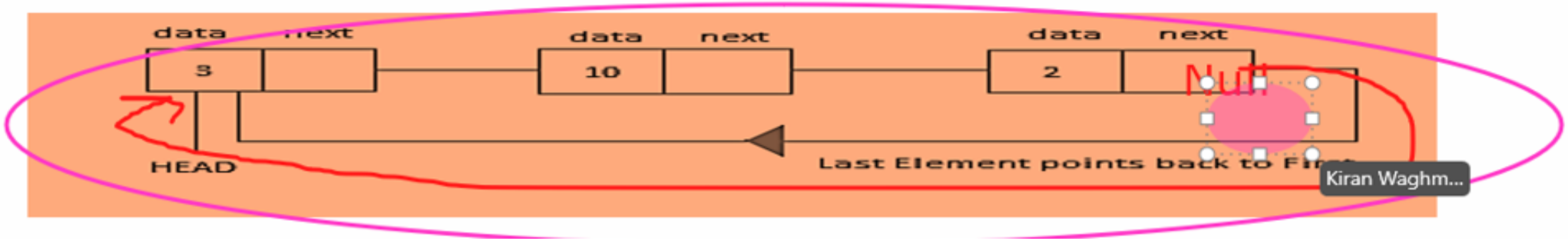


- **Doubly Linked List**



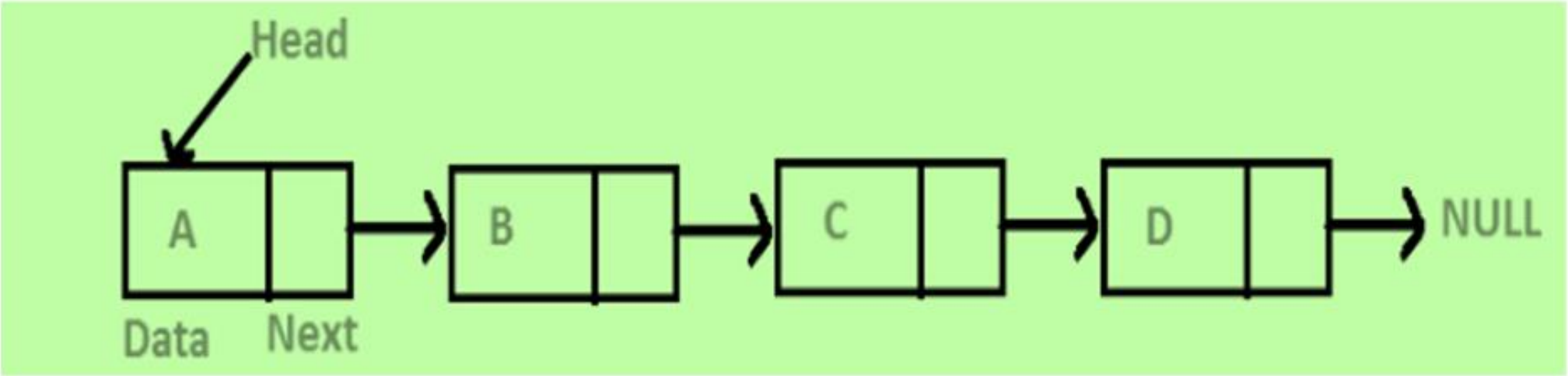
traverse in 2 direction
-Forward: next
-Backward:prev

- **Circular Linked List**



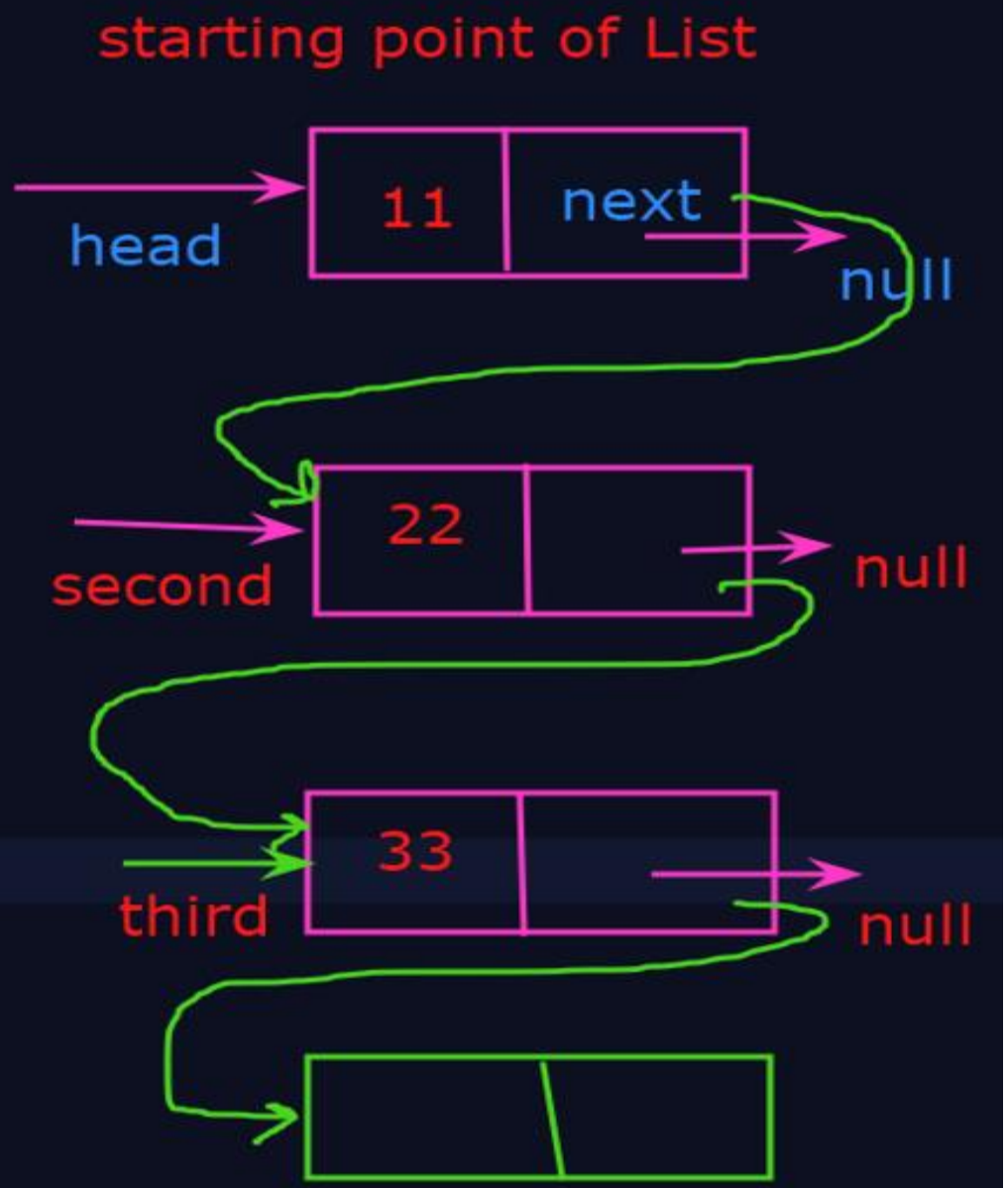
Singly Linked List

- **Singly Linked Operations: Insert, Delete, Traverse, search, Sort, Merge**



```
class Node
{
        int data;
        Node next;
        Node(int d)
        {
            data = d;
            next = null;
        }
}
```

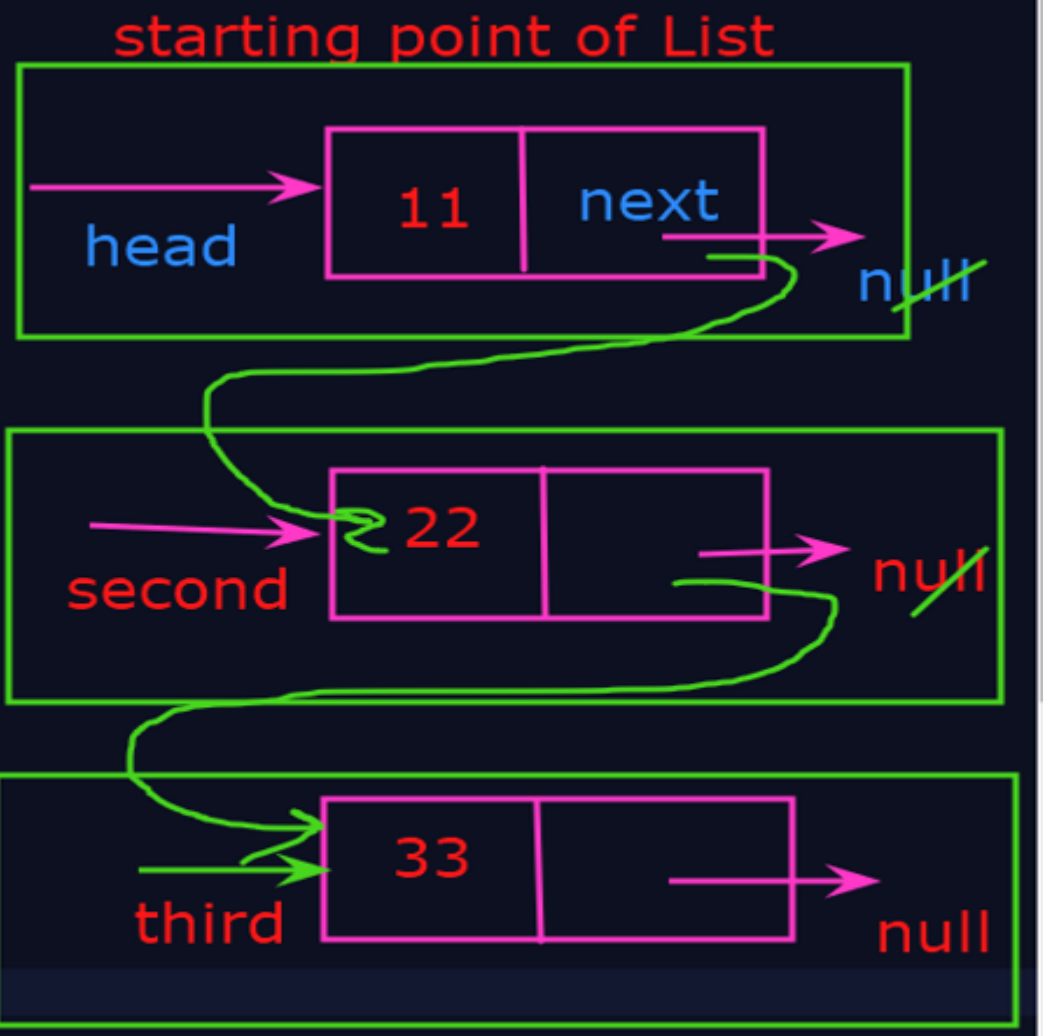
```
        Node next;
        Node(int d)
        {
            data = d;
            next = null;
        }
}

public static void main(String args[])
{
    List1 l1 = new List1();

    l1.head = new Node(11);
    Node second = new Node(22);
    Node third = new Node(33);


}
}
```



```
class List1
{
Node head;
class Node
{
        int data;
        Node next;
        Node(int d)
        {
            data = d;
            next = null;
        }
}
public static void main(String args[])
{
    List1 l1 = new List1();

    l1.head = new Node(11);
    Node second = new Node(22);
    Node third = new Node(33);

    l1.head.next = second;
    second.next = third;
```

=>start of Node

=> create a new Node



```
        Node(int d)
        {
            data = d;
            next = null;
        }
}
```

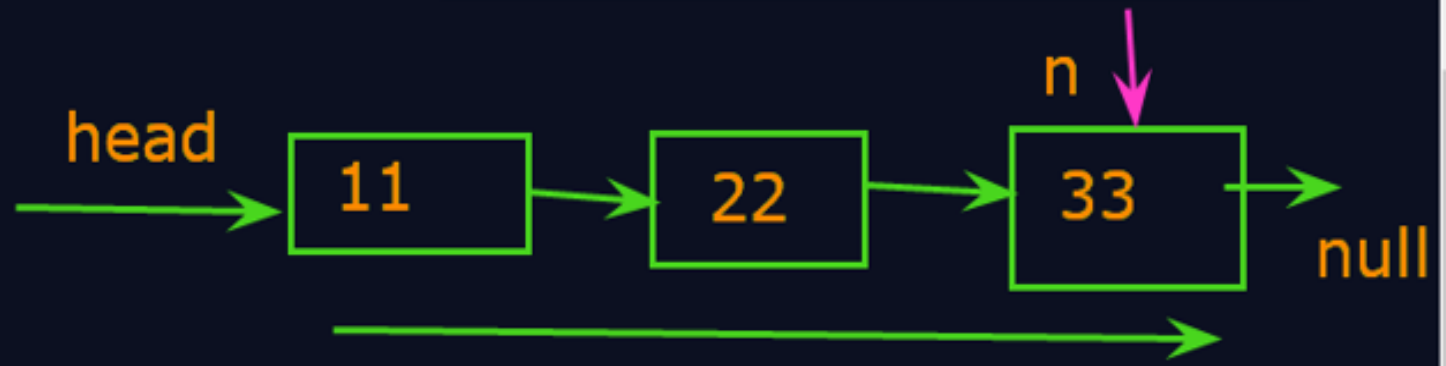

```
public void disply()
{
    Node n = head;
}

public static void main(String args[])
{
    List2 l1 = new List2();

    l1.head = new Node(11);
    Node second = new Node(22);
    Node third = new Node(33);

    l1.head.next = second;
```

n
head
head.data=11
head.next=connextion with 22
n.data=33
n.next=null

n=n.next

head=head.next
head=head.next.next

```
        Node(int d)
        {
            data = d;
            next = null;
        }
}

public void disply()
{
    Node n = head;
    while(n != null)
    {
        System.out.print(n.data);
        n=n.next;
    }
}
```



11 22 33

```
public static void main(String args[])
{
    List2 l1 = new List2();

    l1.head = new Node(11);
    Node second = new Node(22);
    Node third = new Node(33);

    l1.head.next = second;
    second.next = third;

    l1.display();


}
```

Basic Operations

- **Following are the basic operations supported by a list.**
  1. **Insertion** − Adds an element at the beginning of the list.
  2. **Deletion** − Deletes an element at the beginning of the list.
  3. **Display** − Displays the complete list.
  4. **Search** − Searches an element using the given key.
  5. **Delete** − Deletes an element using the given key.

Who can see what you share here? Recording On

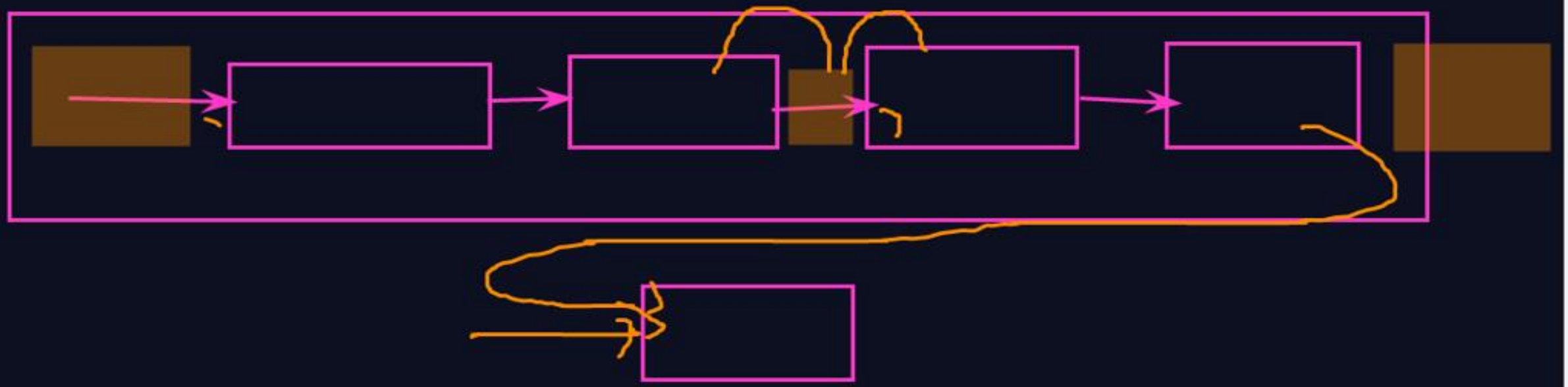
Linked List

Insertion Operation:

1.At begining

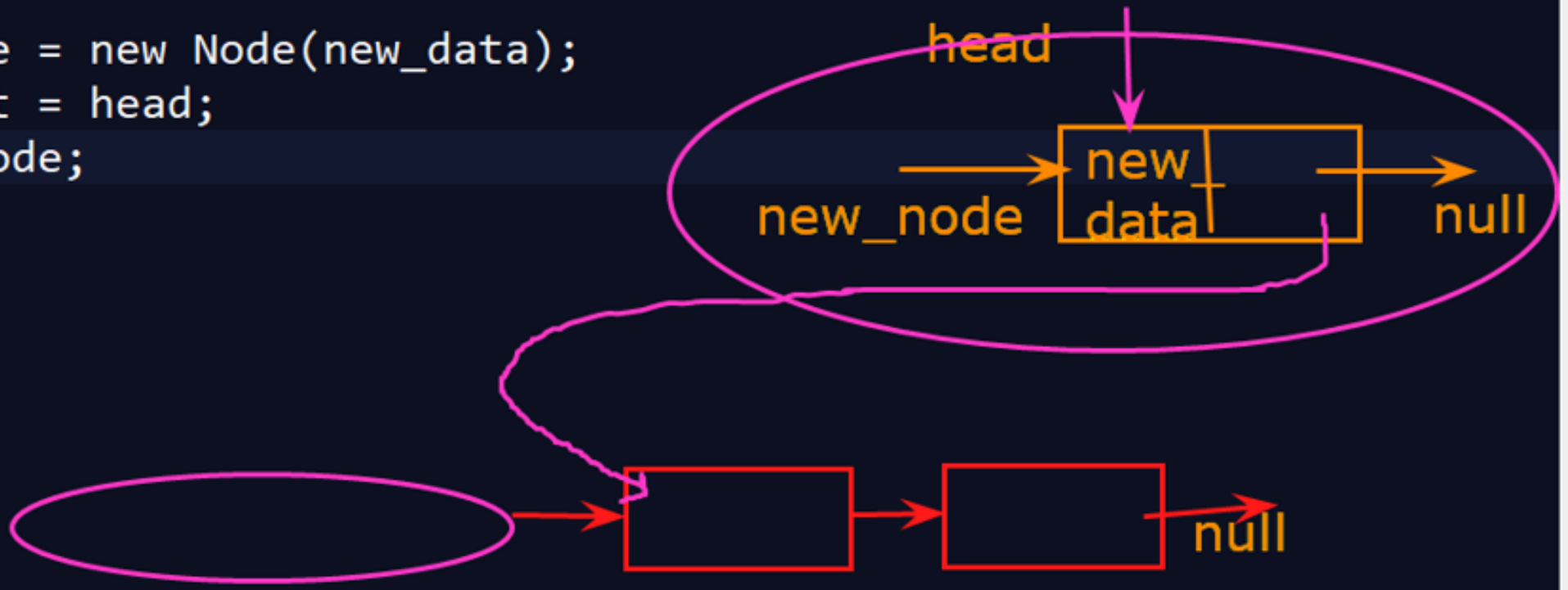
2.In between

3.At end



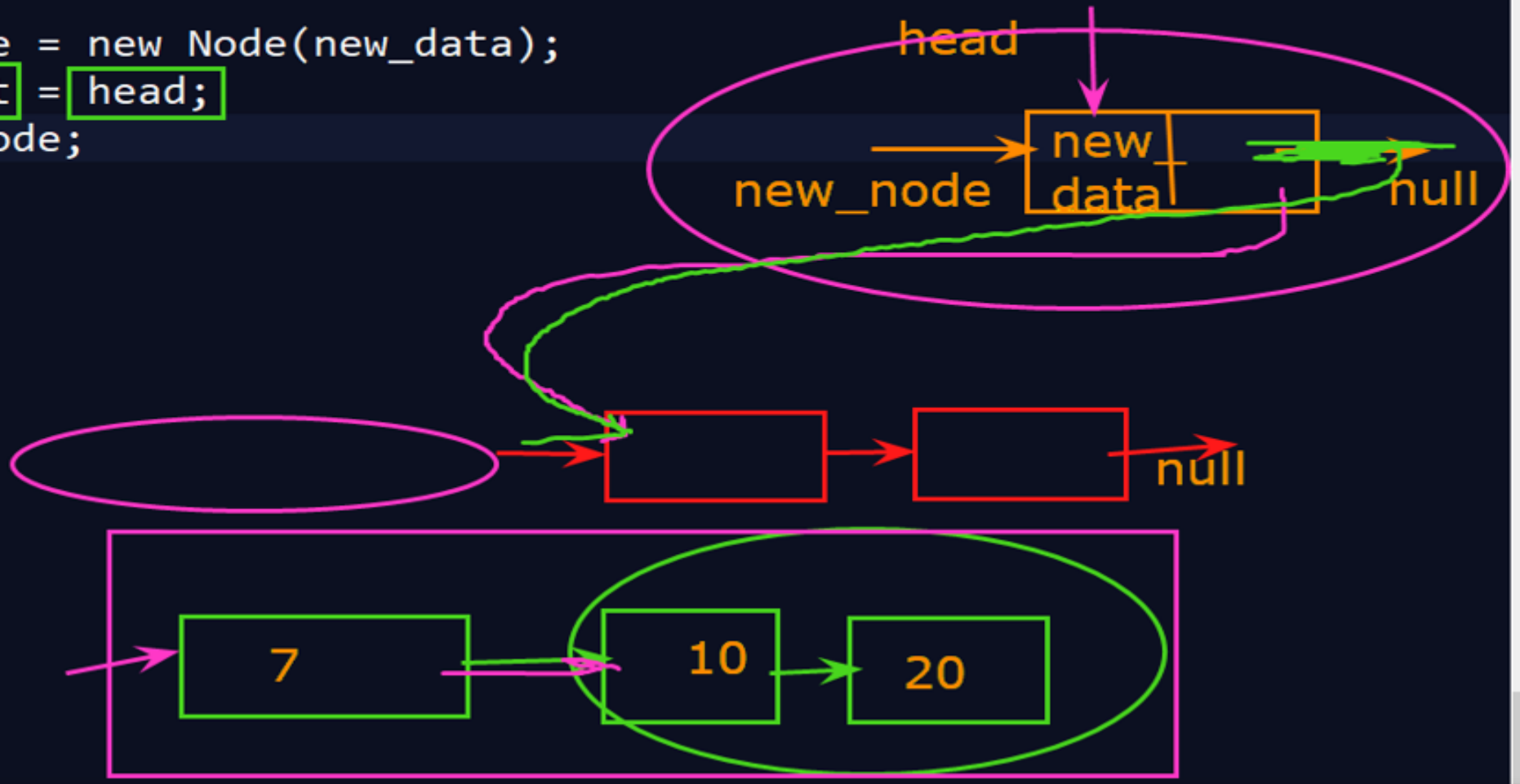
case 1: Insert at begining

```
public void insert(int new_data)
{
    Node new_node = new Node(new_data);
    new_node.next = head;
    head = new_node;

}
```

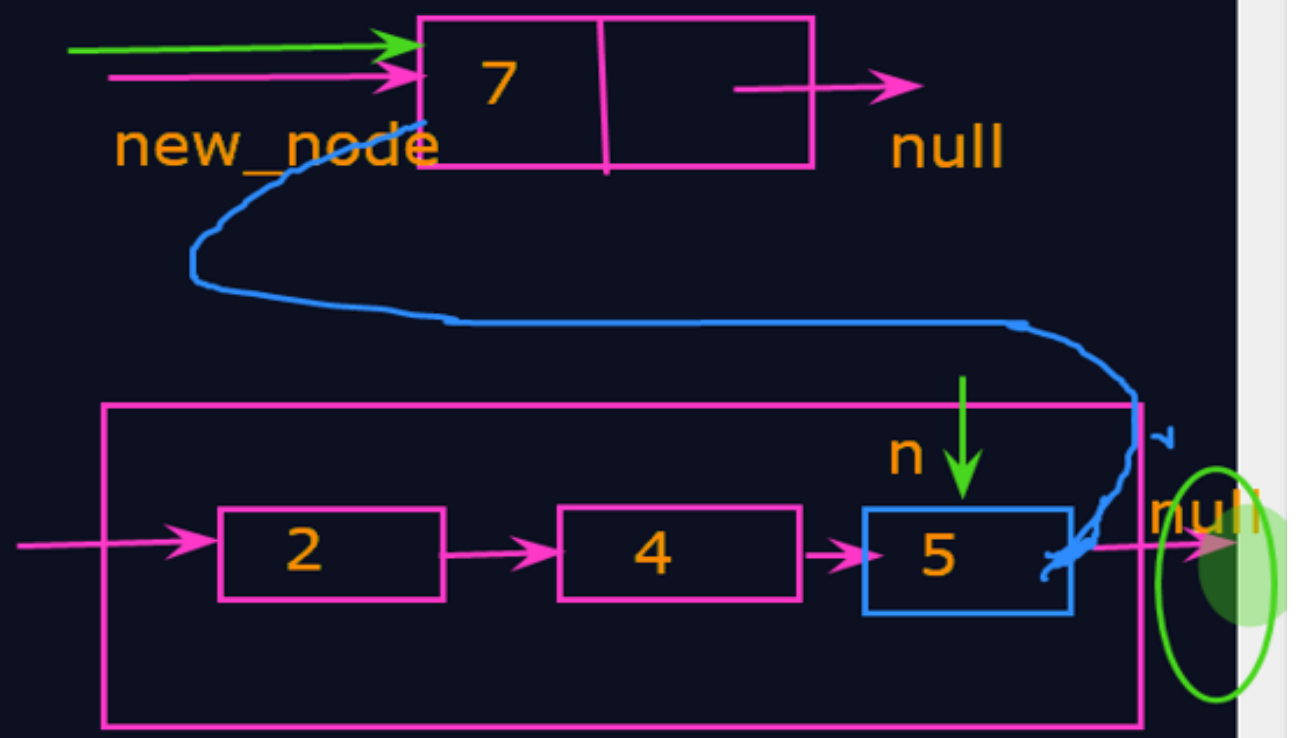


case 1: Insert at begining

```
public void insert(int new_data)
{
    Node new_node = new Node(new_data);
    new_node.next = head;
    head = new_node;

}
```



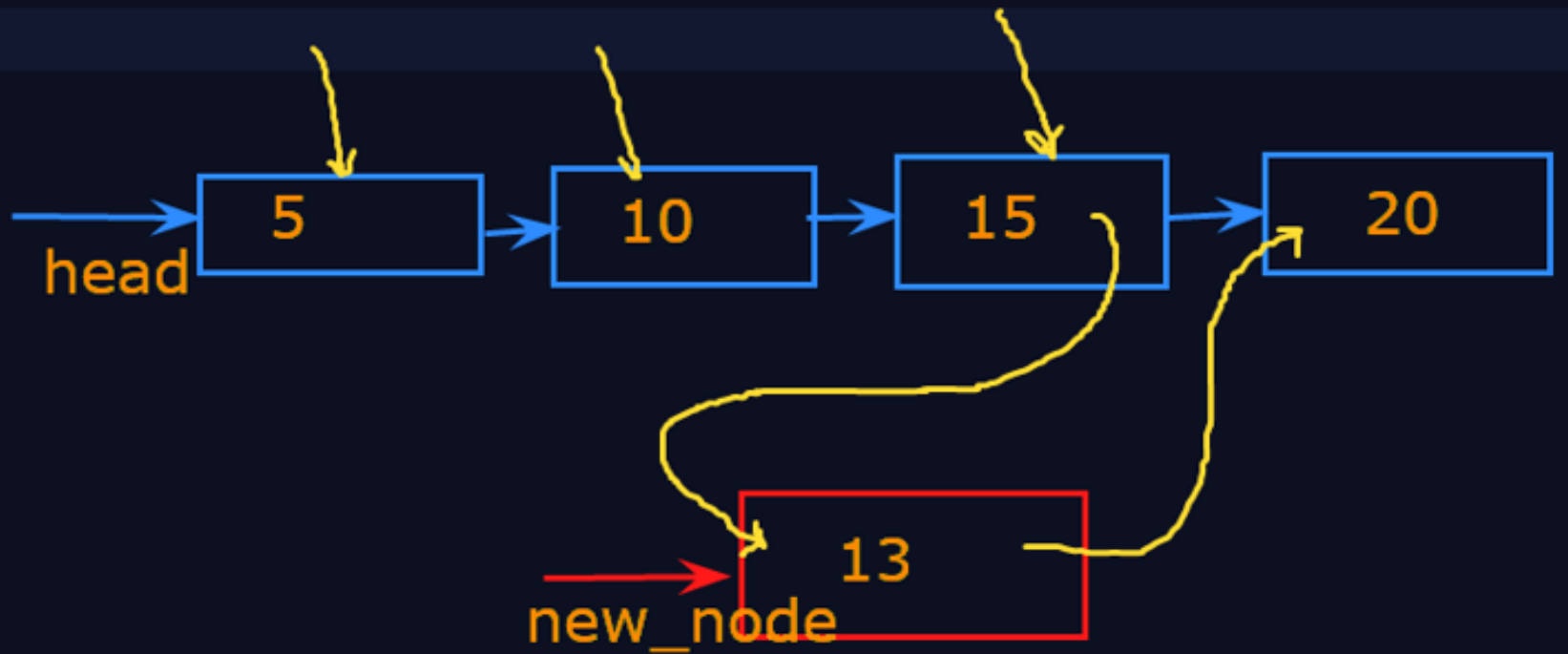
```
    Node new_node = new Node(new_data);
    if(head == null)
    {
        head = new Node(new_data);
        //head = new_node;
        return;
    }

    Node n = head;
    while(n.next != null)
        n=n.next;
    n.next=new_node;
    return;
}
```



null

head == null

Case 3: Insertion of between 2 nodes




```
new_node.next = head.next;
head.next = new_node;
```

```
{
    List3 l1 = new List3();

    l1.head = new Node(11);
    Node second = new Node(22);
    Node third = new Node(33);

    l1.head.next = second;
    second.next = third;

    l1.insert(44);
    l1.insert(55);
    l1.display();


}
```

Who can see what you share here? Recording On

C:\Windows\System32\cmd.exe

```
1 error

C:\Test>javac List3.java

C:\Test>java List3
11---->22---->33---->#
C:\Test>javac List3.java

C:\Test>java List3
55---->44---->11---->22---->33---->#
C:\Test>
```

```
public static void main(String a
{
    List3 l1 = new List3();

    l1.head = new Node(11);
    Node second = new Node(
    Node third = new Node(3

    l1.head.next = second;
    second.next = third;

    l1.insert(44);
    l1.insert(55);
    l1.append(66);
    l1.append(88);
    l1.display();


}
}
```

Who can see what you share here? Recording On

```
C:\Windows\System32\cmd.exe
1 error

C:\Test>javac List3.java

C:\Test>java List3
11---->22---->33---->#
C:\Test>javac List3.java

C:\Test>java List3
55---->44---->11---->22---->33---->#
C:\Test>javac List3.java

C:\Test>java List3
55---->44---->11---->22---->33---->66---->88---->#
C:\Test>
```

```
public static void m
{
    List3 l1 = new L

    l1.head = new No
    Node second = ne
    Node third = new

    l1.head.next = s
    second.next = th

    l1.insert(44);
    l1.insert(55);
    l1.append(66);
    l1.append(88);
    l1.insertAfter(l1.head, 99);
    l1.display();
```

```
C:\Test>javac List3.java

C:\Test>java List3
11---->22---->33---->#
C:\Test>javac List3.java

C:\Test>java List3
55---->44---->11---->22---->33---->#
C:\Test>javac List3.java

C:\Test>java List3
55---->44---->11---->22---->33---->66---->88---->#
C:\Test>javac List3.java

C:\Test>java List3
55---->99---->44---->11---->22---->33---->66---->88---->#
C:\Test>
```

Thanks