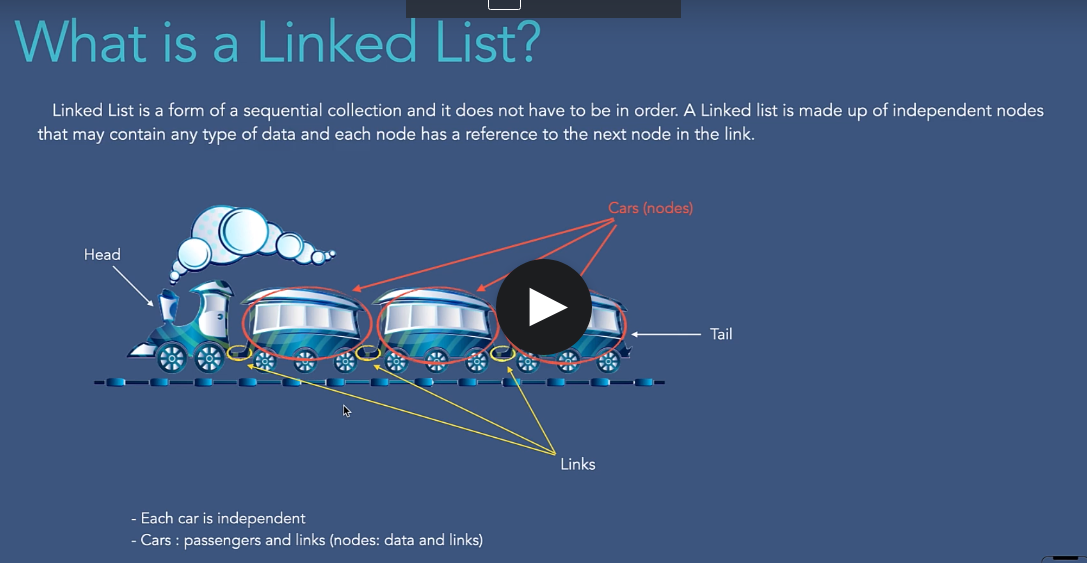
**Linked List**



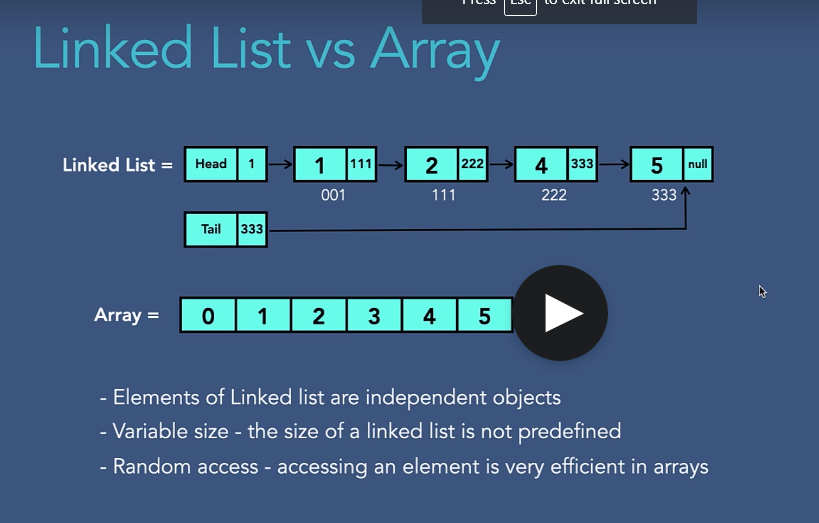
Why we need tail?

by knowing the last elements address, we can increase the efficiency of the inserting an element at the end of linkedList.

Tail is not mandatory.

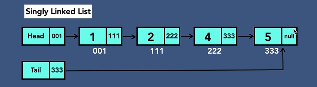
**Linked List vs Array**

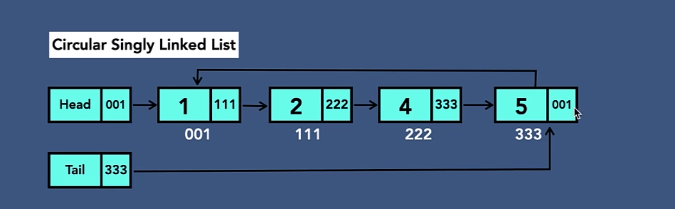
So given these arrays are usually preferred for either small dataset, and for all the data sets that are not shifted around so often.

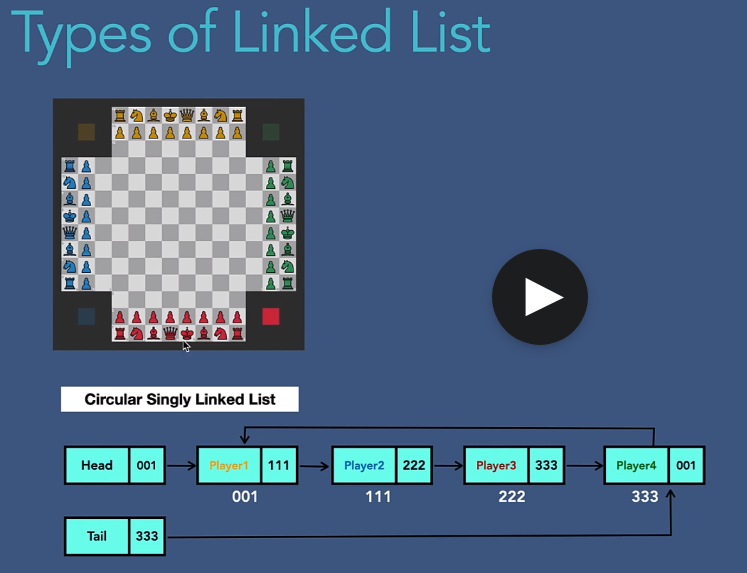


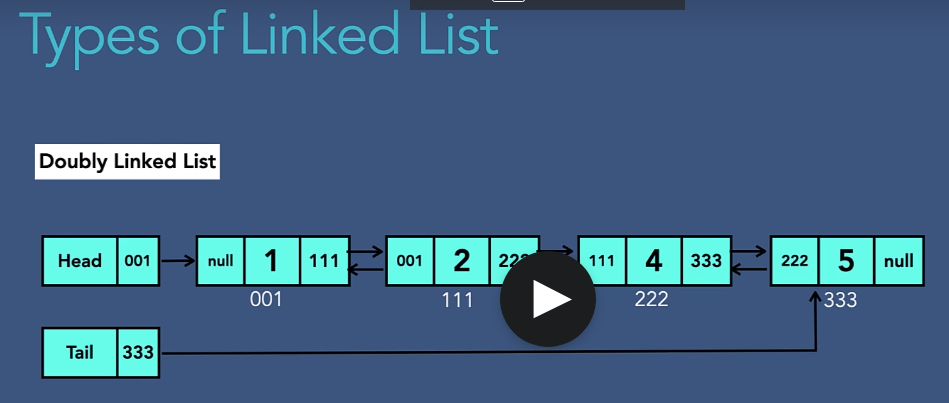
**Types of Linked List**

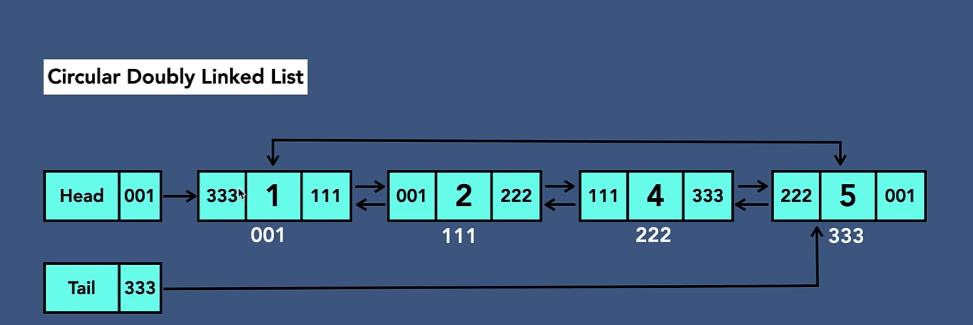












So this gives us flexibility of traversing from first node to the last node and from last node to the first.

**Linked list in memory**

we have just remembered that in case of array elements are located continuously in memory, so which means that they are next to each other.

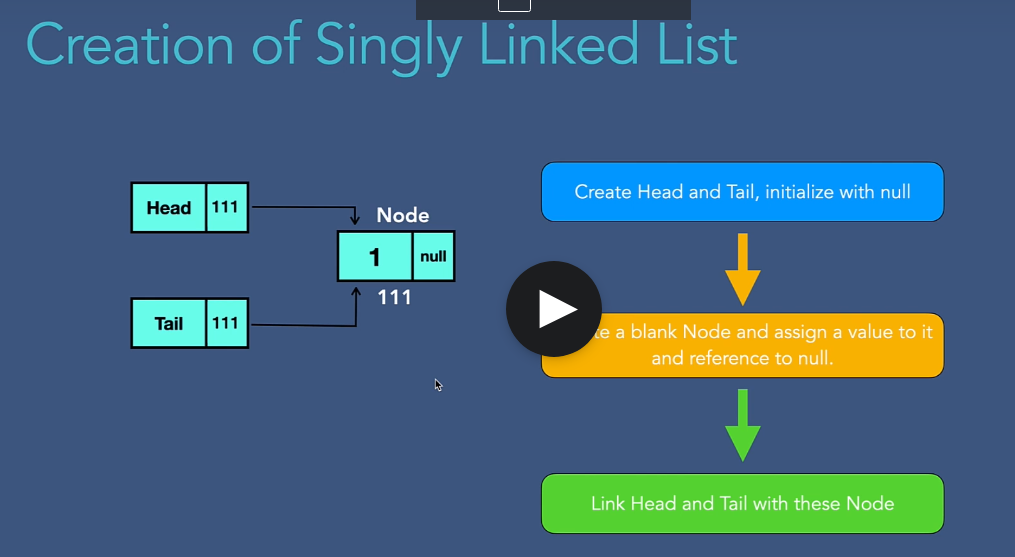
**Creating singly linked list**

Creation, traversal, insertion, deletion, searching – operations

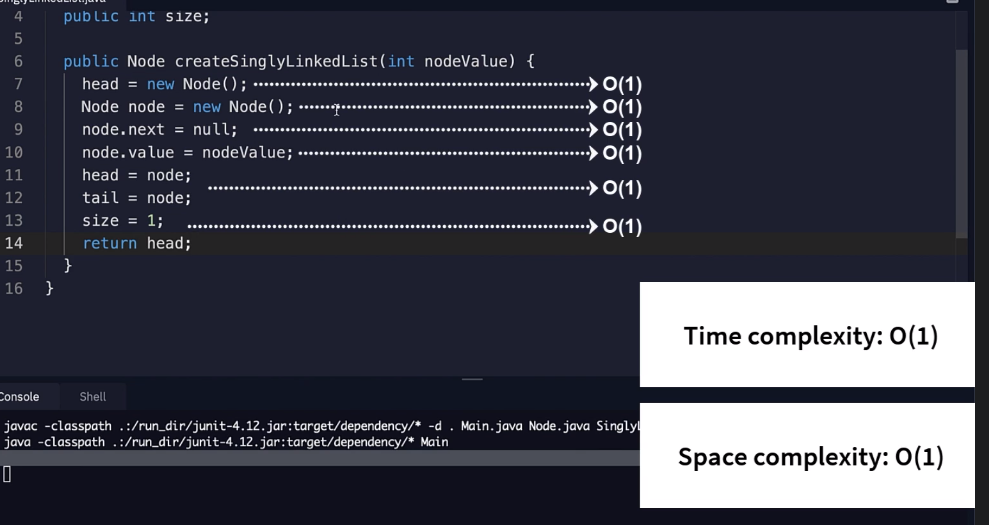
Step 1- create head, tail and initialize the reference to null.

Step 2- create a node with data and reference=null

Step 3- head and tail points to the newly created node.



Code:



**package** LinkedList;

**public** **class** Main {

**public** **static** **void** main(String args[]) {

SinglyLinkedList sll = **new** SinglyLinkedList();

sll.createSinglyLinkedList(5);

System.***out***.println(sll.head.value);

}

}

**package** LinkedList;

**public** **class** Node {

**public** **int** value;

**public** Node next;

}

**package** LinkedList;

**public** **class** SinglyLinkedList {

**public** Node head;

**public** Node tail;

**public** **int** size;

**public** Node createSinglyLinkedList(**int** nodeValue) {

head=**new** Node();

Node node=**new** Node();

node.value=nodeValue;

node.next=**null**;

head=node;

tail=node;

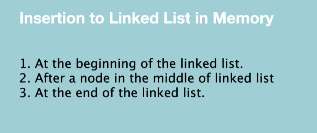
size=1;

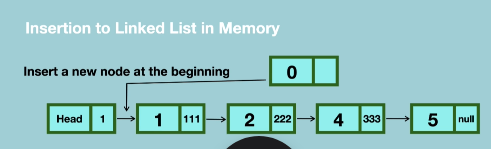
**return** head;

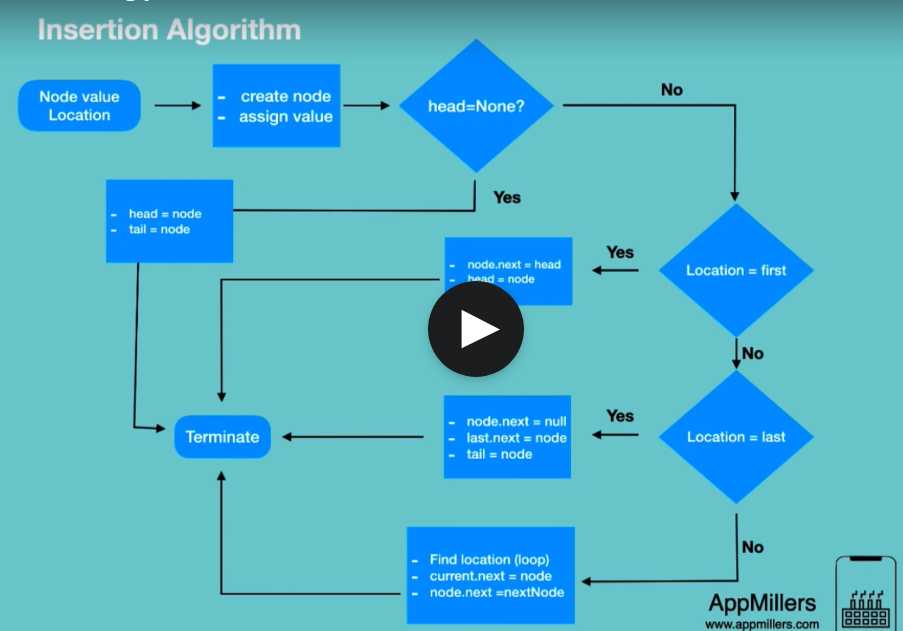
}

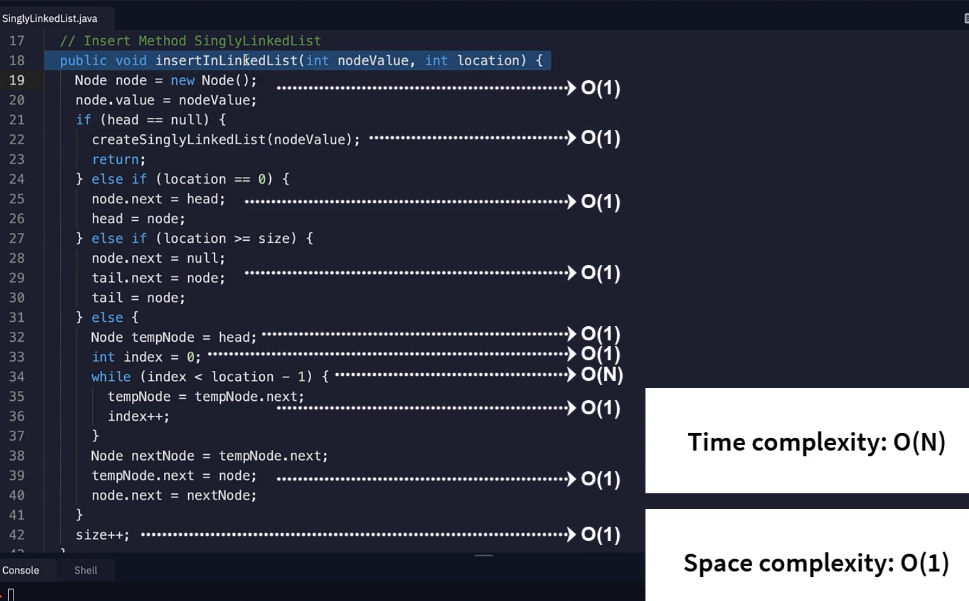
}

Insertion



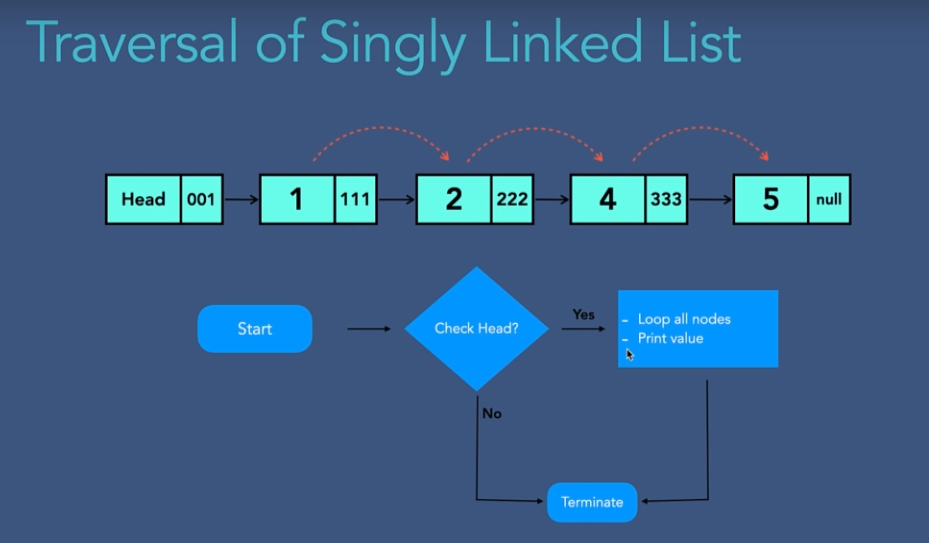


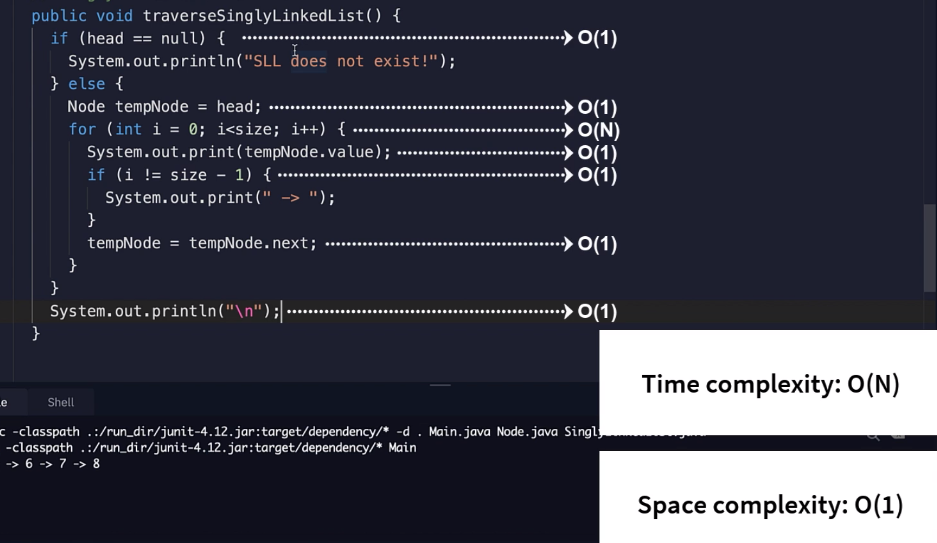




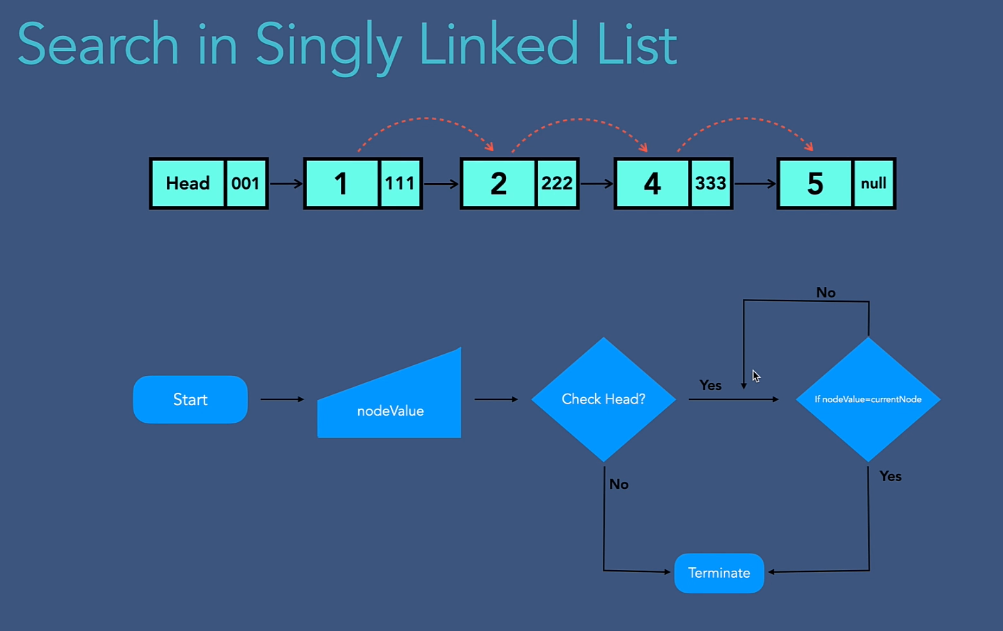
**Traversal of singly linked list**

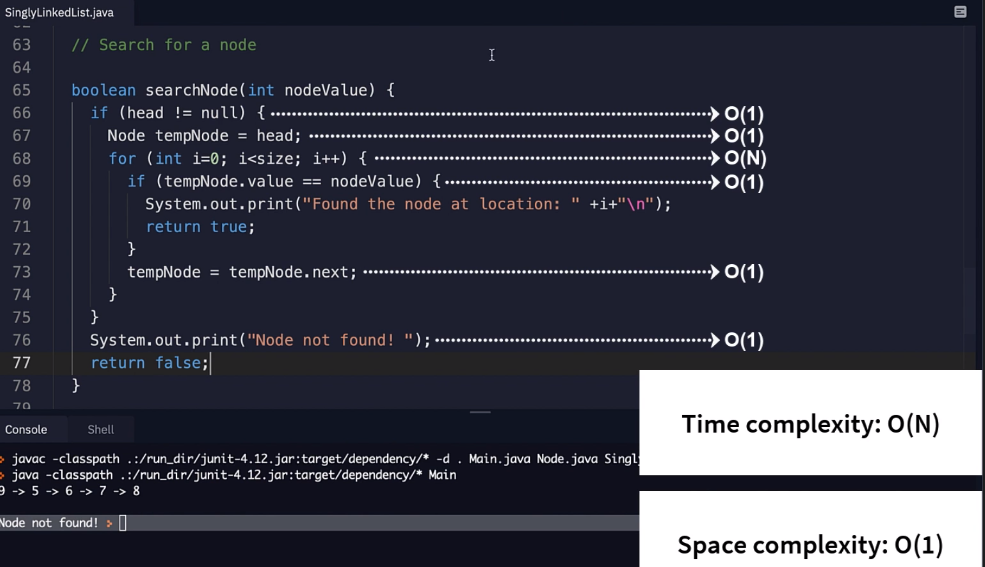
when we reach the last in which the next reference is null, then in this case we are going to terminate our traversal.



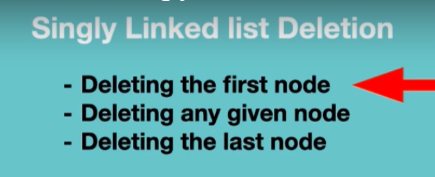


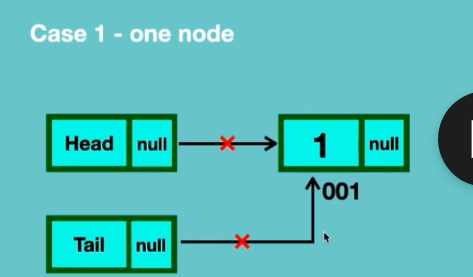
**Searching**

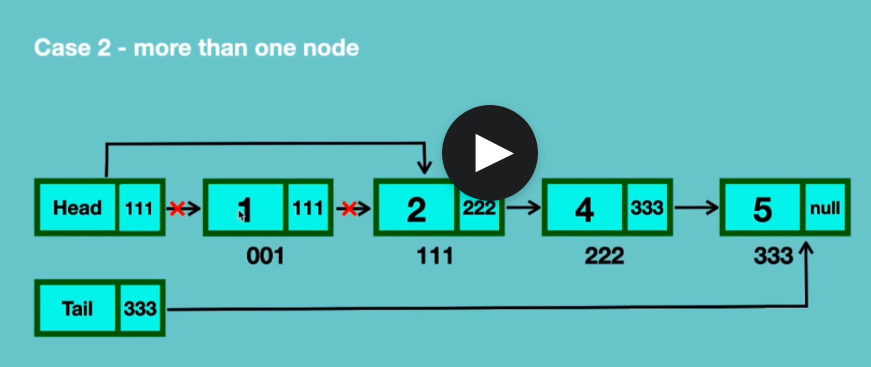


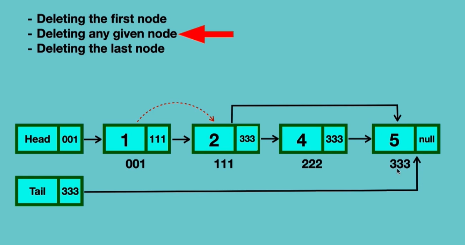


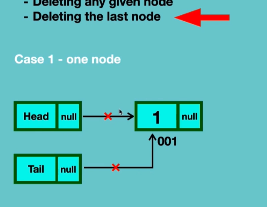
**Deletion of Node**

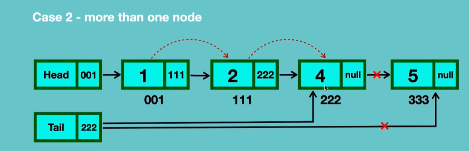


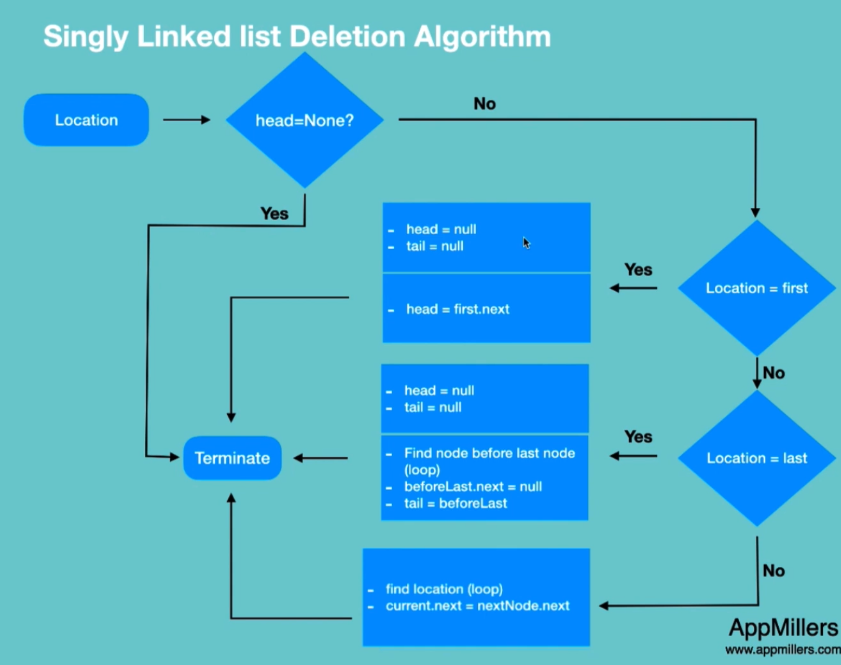




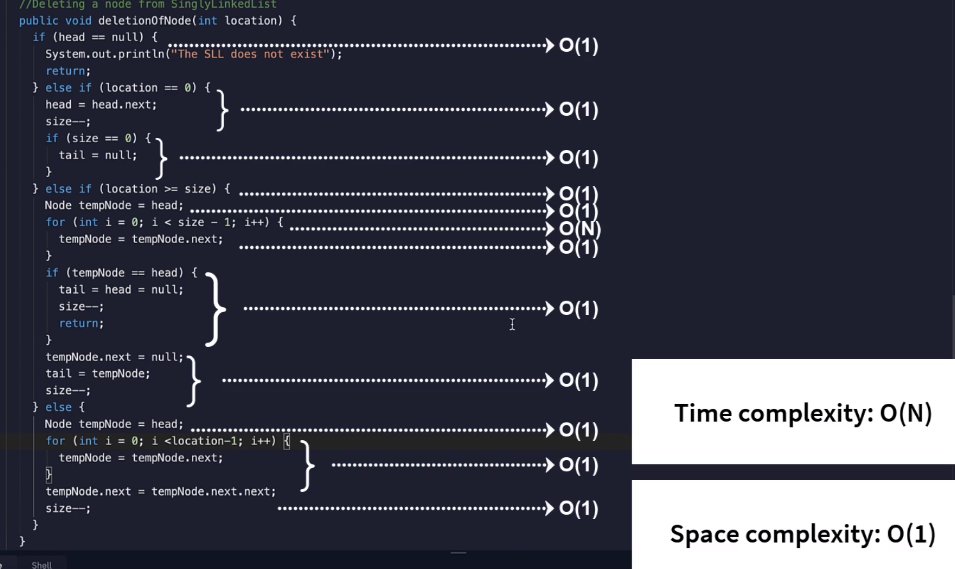








**Deletion method in Singly Linked List**



**Delete entire singly linked list**

All we have to do, we have to disconnect from head and tail references from the first note and from the last node.



**Time and space complexity of singly linked list**

