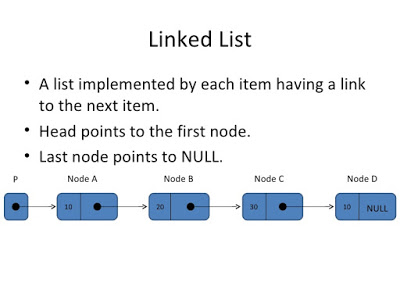
## How do you find length of a Singly Linked list using Loop and Recursion 2 Ways to find length of linked list in Java

Anyway let's come back to the question , everybody knows that in the singly linked lists the last element will point to ["null"](http://javarevisited.blogspot.com/2014/12/9-things-about-null-in-java.html) element , So the first answer would most of the times would be "I will use a counter and increment till it reaches the end of the element".  
  
Once you reach at the end of linked list, the value of counter would be equal to the total number of elements encountered i.e. length of the elements.

[](https://1.bp.blogspot.com/-m3hrAoB-Dag/Vz_mYkD63VI/AAAAAAAAF9s/wFWE3YtNRy4kLywOZ8tw1Xw6yHnYavakwCLcB/s1600/single-linked-list-in-java%2Busing%2BGenerics.jpg)

Here is the sample code to implement this algorithm, you can use the [linked list implementation](http://java67.blogspot.com/2016/01/how-to-implement-singly-linked-list-in-java-using-generics-example.html) given in this article for understanding the purpose. We are assuming the linked list class hold reference of the head, a pointer which points to the first node in the linked list. To learn more about linked list data structure, you can pick a good on data structure and algorithms e.g.  [Introduction to Algorithms](http://aax-us-east.amazon-adsystem.com/x/c/QgVNzGYS72E4AZMsx2b9XwwAAAFhdm2H_wEAAAFKAVRAlRg/https:/assoc-redirect.amazon.com/g/r/http:/www.amazon.com/Introduction-Algorithms-Edition-Thomas-Cormen/dp/0262033844/ref=as_at?creativeASIN=0262033844&linkCode=w61&imprToken=1a1NBAs69GkmEoXWowvPGA&slotNum=1&tag=javamysqlanta-20) by Thomas Cormen.

**Iterative Solutions**  
  
public int length(){

int count=0;

Node current = this.head;

while(current != null){

count++;

current=current.next()

}

return count;

}

If you answer this question without any difficulty most interviewer will ask you to write a ["recursive" solution](http://javarevisited.blogspot.com/2012/12/recursion-in-java-with-example-programming.html)for this problem, just to check how you deal with recursion if your first answer would have been recursive they will ask you to write an "iterative solution" as shown above.  
  
And now, here is *how to find length of singly linked list using recursion in Java*:  
  
**Recursive Solution:**  
public int length(Node current){

if(current == null){ //base case

return 0;

}

return 1+length(current.next());

}

You can see that we have used the fact that last node will point to null to terminate the recursion. This is called the base case. It's very important to identify a base case while coding a recursive solution, without a base case, your program will never terminate and result in StackOverFlowError.  
====================================  
How to find middle element of LinkedList in Java in one pass in Java

Java program to find middle element of LinkedList in one pass

[How to find middle element of Linked List in Java with Example](http://2.bp.blogspot.com/-wrzDeQGAe1I/TWu8pLuLr4I/AAAAAAAAADE/V017G-6Q61w/s1600/java_logo_50_50.jpg)Here is complete Java program to find middle node of Linked List in Java. Remember LinkedList class here is our custom class and don’t confuse this class with [java.util.LinkedList](http://javarevisited.blogspot.sg/2012/02/difference-between-linkedlist-vs.html) which is a popular Collection class in Java. In this Java program, our class LinkedList represent a linked list data structure which contains collection of node and has head and tail. Each Node contains data and address part. Main method ofLinkedListTest class is used to simulate the problem, where we created Linked List and added few elements on it and then iterate over them to find middle element of Linked List in one pass in Java.

**// Java program to find middle of linked list**

**class LinkedList**

**{**

**Node head; // head of linked list**

**/\* Linked list node \*/**

**class Node**

**{**

**int data;**

**Node next;**

**Node(int d)**

**{**

**data = d;**

**next = null;**

**}**

**}**

**/\* Function to print middle of linked list \*/**

**void printMiddle()**

**{**

**Node slow\_ptr = head;**

**Node fast\_ptr = head;**

**if (head != null)**

**{**

**while (fast\_ptr != null && fast\_ptr.next != null)**

**{**

**fast\_ptr = fast\_ptr.next.next;**

**slow\_ptr = slow\_ptr.next;**

**}**

**System.out.println("The middle element is [" +**

**slow\_ptr.data + "] \n");**

**}**

**}**

**/\* Inserts a new Node at front of the list. \*/**

**public void push(int new\_data)**

**{**

**/\* 1 & 2: Allocate the Node &**

**Put in the data\*/**

**Node new\_node = new Node(new\_data);**

**/\* 3. Make next of new Node as head \*/**

**new\_node.next = head;**

**/\* 4. Move the head to point to new Node \*/**

**head = new\_node;**

**}**

**/\* This function prints contents of linked list**

**starting from the given node \*/**

**public void printList()**

**{**

**Node tnode = head;**

**while (tnode != null)**

**{**

**System.out.print(tnode.data+"->");**

**tnode = tnode.next;**

**}**

**System.out.println("NULL");**

**}**

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

**{**

**LinkedList llist = new LinkedList();**

**for (int i=5; i>0; --i)**

**{**

**llist.push(i);**

**llist.printList();**

**llist.printMiddle();**

**}**

**}**

**}**

**// This code is contributed by Rajat Mishra**  
  
Read more: <http://javarevisited.blogspot.com/2012/12/how-to-find-middle-element-of-linked-list-one-pass.html#ixzz56XSxCeo1>