**Linked List | Set 1 (Introduction)**

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.

[](https://www.geeksforgeeks.org/wp-content/uploads/gq/2013/03/Linkedlist.png)

**Why Linked List?**  
Arrays can be used to store linear data of similar types, but arrays have following limitations.  
**1)** The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.  
**2)** Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shifted.

For example, in a system if we maintain a sorted list of IDs in an array id[].

id[] = [1000, 1010, 1050, 2000, 2040].

And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).  
Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.

**Advantages over arrays**  
**1)** Dynamic size  
**2)** Ease of insertion/deletion

**Drawbacks:**  
**1)** Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists efficiently with its default implementation. Read about it [here](https://www.geeksforgeeks.org/binary-search-on-singly-linked-list/).  
**2)** Extra memory space for a pointer is required with each element of the list.  
**3)** Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

**Representation:**  
A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL.  
Each node in a list consists of at least two parts:  
1) data  
2) Pointer (Or Reference) to the next node  
In C, we can represent a node using structures. Below is an example of a linked list node with an integer data.  
In Java, LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contains a reference of Node class type.

*filter none*

*brightness\_4*

|  |
| --- |
| class LinkedList  {      Node head;  // head of list        /\* Linked list Node\*/      class Node      {          int data;          Node next;            // Constructor to create a new node          // Next is by default initialized          // as null          Node(int d) {data = d;}      }  } |

*filter none*

*brightness\_4*

|  |
| --- |
| // A simple Java program to introduce a linked list  class LinkedList  {      Node head;  // head of list        /\* Linked list Node.  This inner class is made static so that         main() can access it \*/      static class Node {          int data;          Node next;          Node(int d)  { data = d;  next=null; } // Constructor      }        /\* method to create a simple linked list with 3 nodes\*/      public static void main(String[] args)      {          /\* Start with the empty list. \*/          LinkedList llist = new LinkedList();            llist.head  = new Node(1);          Node second = new Node(2);          Node third  = new Node(3);            /\* Three nodes have been allocated  dynamically.            We have refernces to these three blocks as first,            second and third              llist.head        second              third               |                |                  |               |                |                  |           +----+------+     +----+------+     +----+------+           | 1  | null |     | 2  | null |     |  3 | null |           +----+------+     +----+------+     +----+------+ \*/            llist.head.next = second; // Link first node with the second node            /\*  Now next of first Node refers to second.  So they              both are linked.             llist.head        second              third              |                |                  |              |                |                  |          +----+------+     +----+------+     +----+------+          | 1  |  o-------->| 2  | null |     |  3 | null |          +----+------+     +----+------+     +----+------+ \*/            second.next = third; // Link second node with the third node            /\*  Now next of second Node refers to third.  So all three              nodes are linked.             llist.head        second              third              |                |                  |              |                |                  |          +----+------+     +----+------+     +----+------+          | 1  |  o-------->| 2  |  o-------->|  3 | null |          +----+------+     +----+------+     +----+------+ \*/      }  } |

**Linked List Traversal**  
In the previous program, we have created a simple linked list with three nodes. Let us traverse the created list and print the data of each node. For traversal, let us write a general purpose function printList() that prints any given list.

[**We strongly recommend that you click here and practice it, before moving on to the solution.**](https://practice.geeksforgeeks.org/problem-page.php?pid=700004)

*filter none*

*edit*

*play arrow*

*brightness\_4*

|  |
| --- |
| // A simple Java program for traversal of a linked list  class LinkedList  {      Node head;  // head of list        /\* Linked list Node.  This inner class is made static so that         main() can access it \*/      static class Node {          int data;          Node next;          Node(int d)  { data = d;  next=null; } // Constructor      }        /\* This function prints contents of linked list starting from head \*/      public void printList()      {          Node n = head;          while (n != null)          {              System.out.print(n.data+" ");              n = n.next;          }      }        /\* method to create a simple linked list with 3 nodes\*/      public static void main(String[] args)      {          /\* Start with the empty list. \*/          LinkedList llist = new LinkedList();            llist.head       = new Node(1);          Node second      = new Node(2);          Node third       = new Node(3);            llist.head.next = second; // Link first node with the second node          second.next = third; // Link first node with the second node            llist.printList();      }  } |

**Output:**

1 2 3