Linked List

What is Linked List?

A linked list is a data structure used for storing collections of data. A linked list has the following properties.

* Successive elements are connected by pointers
* The last element points to NULL
* Can grow or shrink in size during execution of a program
* Can be made just as long as required (until systems memory exhausts)
* Does not waste memory space (but takes some extra memory for pointers). It allocates memory as list grows.



**Linked Lists Abstract Data Type (ADT)**

The following operations make linked lists an ADT:

**Main Linked Lists Operations**

* + Insert: inserts an element into the list
  + Delete: removes and returns the specified position element from the list

**Auxiliary Linked Lists Operations**

* + Delete List: removes all elements of the list (disposees the list)
  + Count: returns the number of elements in the list
  + Find *nth* node from the end of the list



Singly Linked List

Generally “linked list” means a singly linked list. This list consists of a number of nodes in which each node has a *next* pointer to the following element. The link of the last node in the list is NULL, which indicates the end of the list.

Type declaration of Linked List:

package com.skg.linkedList.singlyLinedList;

public class ListNode {

private int data;

private ListNode next;

public ListNode(int data) {

this.data = data;

}

public void setData(int data) {

this.data = data;

}

public int getData() {

return this.data;

}

public void setNext(ListNode listNode) {

this.next = listNode;

}

public ListNode getNext() {

return this.next;

}

}

**Basic Operations on a List**

* Traversing the list
* Inserting an item in the list
* Deleting an item from the list

**Traversing the list**

Let us assume that the *head* points to the first node of the list. To traverse the list we do the following.

* Follow the pointers.
* Display the contents of the nodes (or count) as they are traversed.
* Stop when the next pointer points to NULL.