**LINKED LIST**

* Direct or random access of elements is not possible.
* In linked list, the successive elements, need no to occupy contiguous space in memory.
* Print all nodes of a LL:

void doSomething(**List** head)

{

**if**(head != **NULL**)

{

**print**(head->data);

 doSomething(head->next);

}

}

* Print all nodes in Reverse order:

void doSomething(**List** head)

{

**if**(head != **NULL**)

{

 doSomething(head->next);

**print**(head->data);

}

}

* It is easy to insert and delete elements in Linked List.
* Sequential access is allowed in a typical implementation of Linked Lists.
* The size of array has to be pre-decided, linked lists can change their size any time.
* Linked Lists does not allow accessing data in random manner.
* LL are best suited for size of structure and DS is constantly changing.

**Reverse a singly LL:**

void reverse(List head) {

List prev = NULL; List current = head; List next;

while (current != NULL) {

next = current->next; current->next = prev;

prev = current; current = next;

}

head = prev; // <- this is the missing line

}

* **Print the List**

static void forwardPrint(Node head) {

Node current = head;

while (current != null) {

System.out.print(current.data + "-");

current = current.next;

}

}

static void backwardPrint(Node head) {

if (head == null) return;

backwardPrint(head.next);

System.out.print(head.data + "-");

}

* Inserting node n1 at the end of the list.

void doSomething(**List** head, Node n1)

{

**if**(head == **null**)

        head = n1;

**else**

     {

**List** t, curr;

**for**(curr = head; (t = curr -> next) != **NULL**; curr = t);

curr->next = n1;

     }

}