# LINKEDLIST

- LinkedList INTERNAL WORKING?

  (1) Internally doubly linkedList uses.

  (2) better for frequent insertion and deletion

  (3) linkedList has slower random access because it has to traverse list from the beginning to reach desired index.

  (4) linkedList requires more memory as compare to array List.

- CODE EXAMPLE:

```
    class Node {
    public int value;
     public Node next;
}
public class LinkList_01 {

    public static void main(String[] args) {

        // for understanding linkedList
     /*
      Node node1 = new Node();
      Node node2 = new Node();
           node1.value=1;
           node1.next=node2;
           node2.value = 4;
           node2.next = null
       */


        LinkedList<Integer> list = new LinkedList<>();

        // add ()
        list.add(2);
        list.add(3);
        list.add(4);

        // get () because java has provided get still it will use loops
and time complexity O(n)
        list.get(0);

        // addFirst ()
        list.addFirst(7);

        // addLast ()
        list.addLast(11);
```

```
        // getFirst ()
        list.getFirst();

        // getLast ()
        list.getLast();

        // remove () with index
        list.remove(3);

        // remove () with obj
        list.remove(Integer.valueOf(3));

        // removeFirstOccurrence () : if multiple time exist it will
remove first one
        list.removeFirstOccurrence(1);

        // removeLastOccurrence () : if multiple time exist it will
remove last one
        list.removeLastOccurrence(9);

        // removeAll () : remove matching value
        LinkedList<String> li1 = new
LinkedList<>(Arrays.asList("cat","dog","elephant"));
        LinkedList<String> li2 = new
LinkedList<>(Arrays.asList("cat","dog","lion"));
       li1.removeAll(li2);// elephant, lion
        System.out.println(li1);


        // linkedList completed as array list (individually)
behaviour rest in next chapter.
    }
}
```