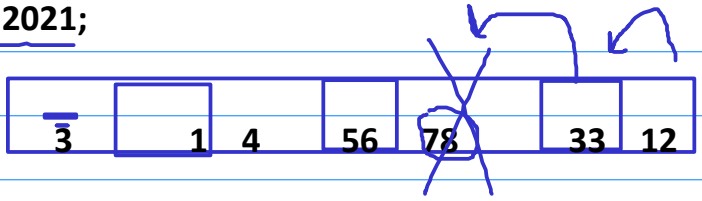
```
dd=dd+1; 32
    if(dd>noOfDays[mm]) {
        dd=1;
        mm++;
        if(mm>12) {
            mm=1;
            yyyy++;
        }
    }
```

31

```
int dd, mm, yyyy;
    dd = 31;
    mm = 12;
    yyyy = 2021;
```

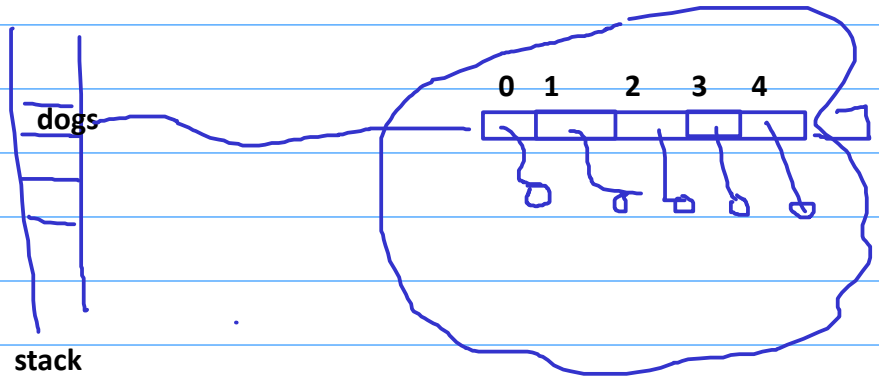


+ : random access is very fast : O(1)

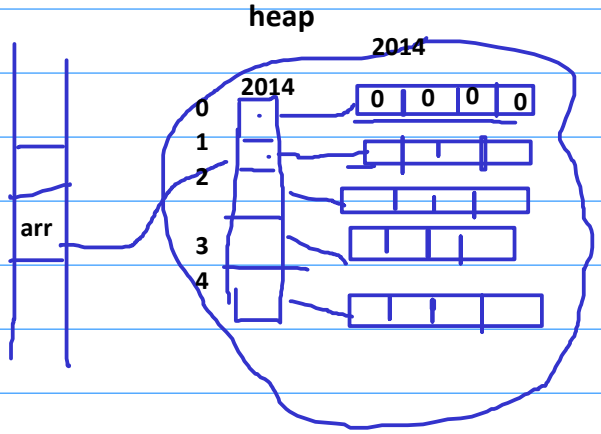
-: add/del element inside array is a slow process

O(N)

```
Dog []dogs=new Dog[5];
//how many dogs are there in this array?
```



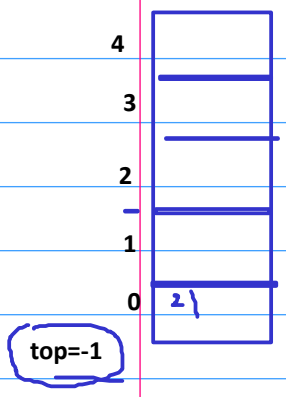

```
int arr[][]=new int [5][4];

    //?
    for(int temp[]:arr) {
        for(int tempVal:temp) {
            System.out.print(tempVal+" ");
        }
        System.out.println();
    }
```



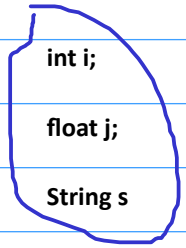
$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix}$$

$AI = A$



ADT? "Abstract data type"




```
class Stack{
}
```

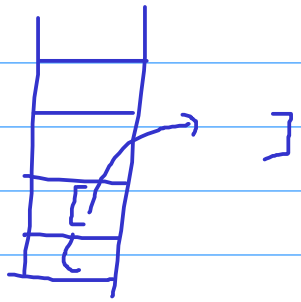
Array
top=-1

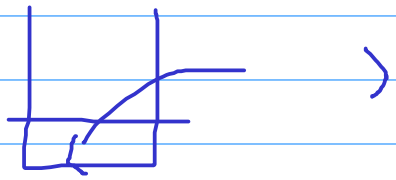
```
return top==-1;
```

stack applications:

validity of an expression

(2X3 + [5+2/7])

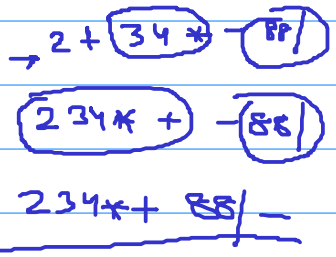




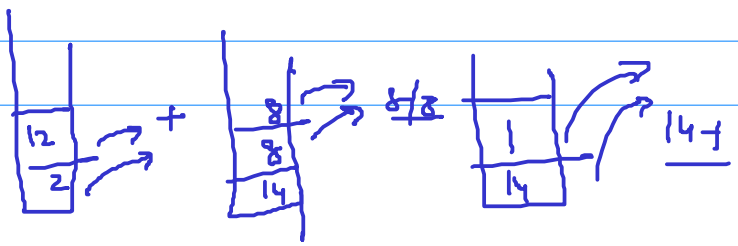
evaluation of an expression

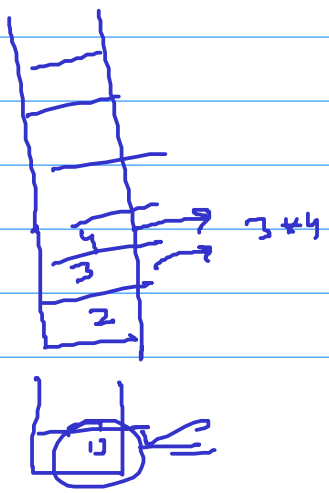
infix ---> postfix

2+3*4-8/8

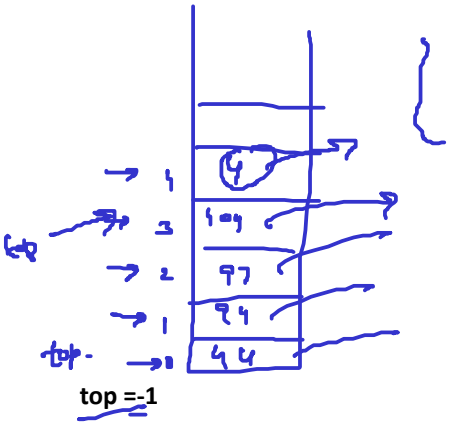


postfix, prefix, infix



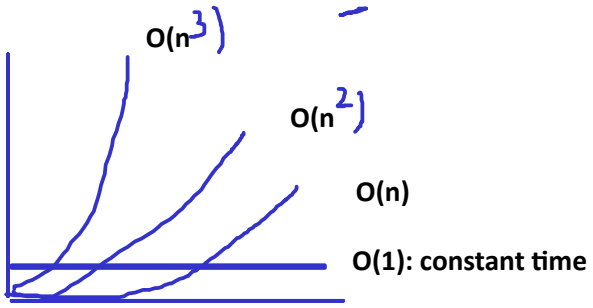


```
Stack stack=new Stack(5);
        stack.push(44);
        stack.push(94);
        stack.push(97);
        stack.push(404);
        stack.push(4);
        System.out.println(stack.pop());
        System.out.println(stack.pop());
        System.out.println(stack.pop());
        System.out.println(stack.pop());
        System.out.println(stack.pop());
```




```
public void push(int data) {
        if (!isFull()) {
            arr[++top] = data;
        } else
            System.out.println("stack is full");
    }

    public int pop() {
        if (!isEmpty()) {
            return arr[top--];
        } else {
            System.out.println("stack is empty");
            return -999;
        }
    }
```

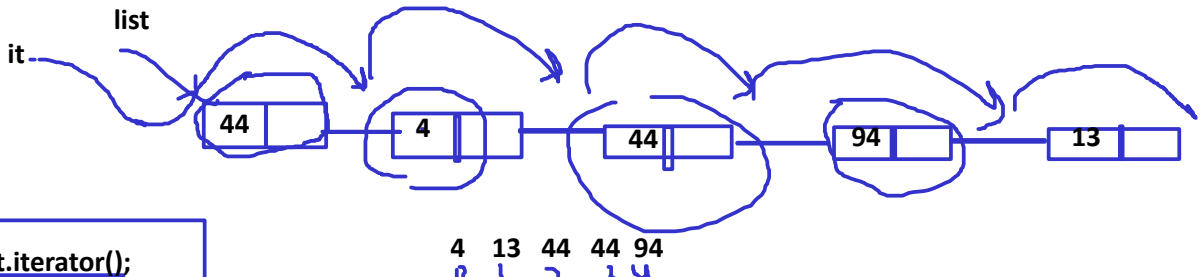
"it depends"



Arrays	LinkedList
radom access is very fast O(1)	only option of linear seach O(n)
can not grow at run time	possbile
not flexible	more flexible
add/del an element inbetween is slow	it is just pointer manipulation => fast

2 option : readymade / create ur own

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

        list.add(44);
        list.add(4);
        list.add(44);
        list.add(94);
        list.add(13);
```



4 13 44 44 94
0 1 2 3 4

```
Iterator<Integer> it=list.iterator();
    while(it.hasNext()) {
        System.out.println(it.next());
    }
```

```
int index=Collections.binarySearch(list, 3);
```

-(-2+1)
=0

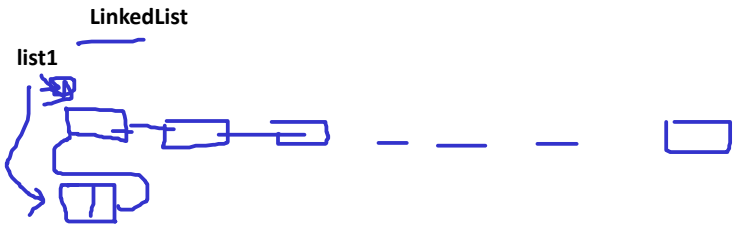
11

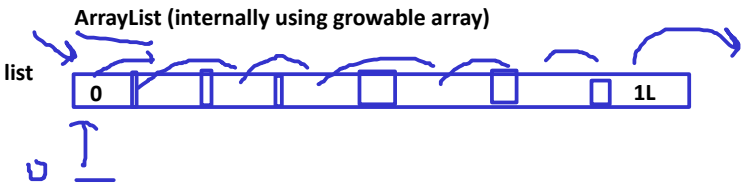
-2+1
=-1

prove the performace of LL is better then AL

DLL (Growable array)

```
private static void doBenchmark(List<Long> list1) {
        //first i will add 1L element into list
        for(long i=0;i<1E5; i++) {
            list1.add(i);
        }

        long start=System.currentTimeMillis();
        for(long i=0;i<1E5; i++) {
            list1.add(0, i);
        }

        long end=System.currentTimeMillis();

        System.out.println("time taken: "+ (end-start)+" ms");
    }
```





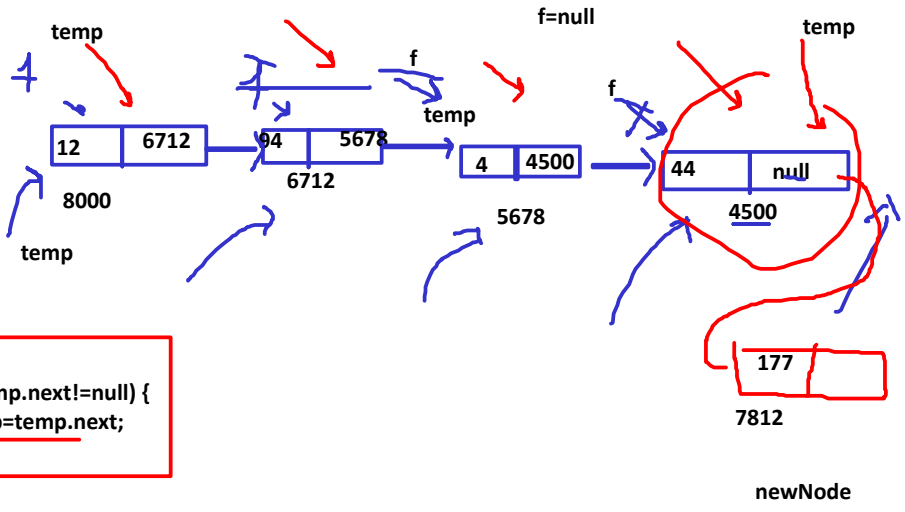
how to create my own link list?

LL is self ref structure?

```
class Node{
        int data;
        Node next;

}
```

```
class LL{
        Node f, l;

        addFirst
        addLast

        delFirst
        delLast
        print
        ...
}
```

```
void insertFirst(int data) {
        Node temp=new Node(data);
        temp.next=f;
        f=temp;
    }
    void displayList() {
        Node temp=f;
        while(temp!=null) {
            System.out.print(temp.data +"->");
            temp=temp.next;
        }
    }
```

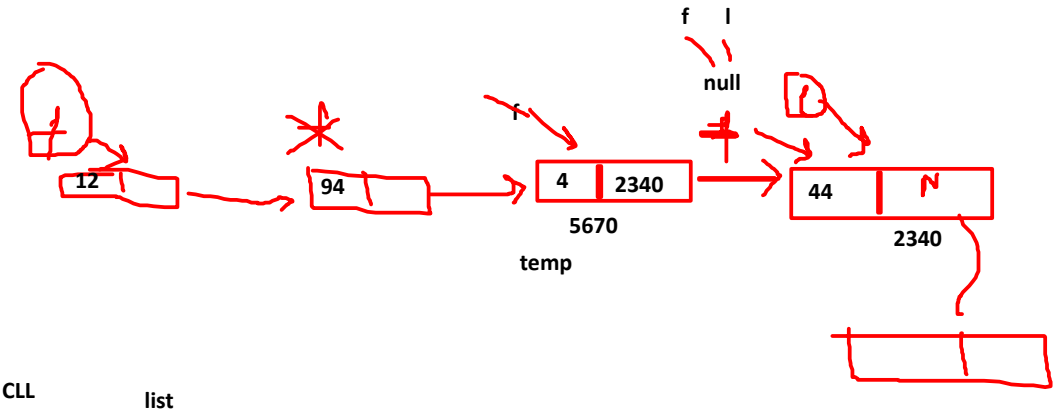



```
list.insertFirst(44);
        list.insertFirst(4);
        list.insertFirst(94);
        list.insertFirst(12);

        list.displayList();
```

```
Node temp=f;
    while(temp.next!=null) {
        temp=temp.next;
    }
```

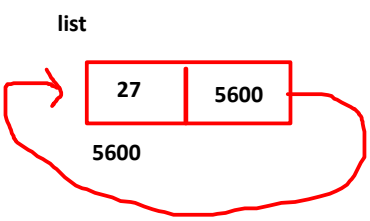
Initially f=l=null;

```
void insertFirst(int data) {
    Node temp=new Node(data);
    if(isEmpty()) {
        l=temp;
    }
    temp.next=f;
    f=temp;
}
```



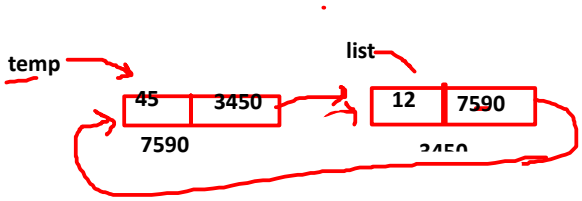

```
list.insertFirst(44);
list.insertFirst(4);
list.insertFirst(94);
list.insertFirst(12);
```

CLL




```
class CLL{
    Node list;
    public insertFront(int data){
    Node tempNode=new Node(data);

    if(list==null){
        list=tempNode;
    }else
        tempNode.next=list.next;

    list.next=tempNode;
}
```

list=null



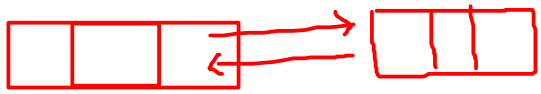
Print method

```
Node temp=list.next;
while(temp!=list){
    sysout(temp.data);
    temp=temp.next;
}
```

3 how to impl stack and queues using LL?

LIFO

DLL




```
class Node{

    int data;
    Node next, prev;
}
```

```
class DLL{

    Node first, last;

    ....
    public boolean isEmpty(){
        return first==null;
    }

    ///....

}
```

```
public void insertFirst(int data) {
    Node tempNode=new Node(data);      //333
    if(isEmpty()) {
        last=tempNode;
    }else {
        first.prev=tempNode;
    }
    tempNode.next=first;
    first=tempNode;
}
```

first=last=null;