# STACK (C)

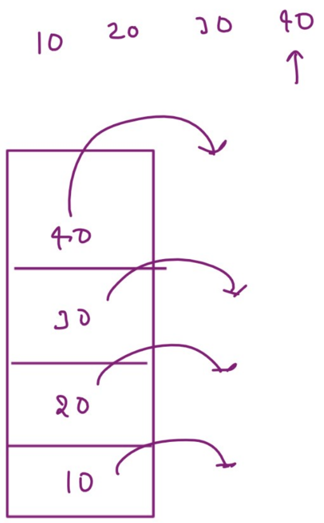
1. The underlying data structure is array
2. It follows LIFO/FILO operations
   Last in first out / First in last out

**Constructor**

Stack S = new Stack ()

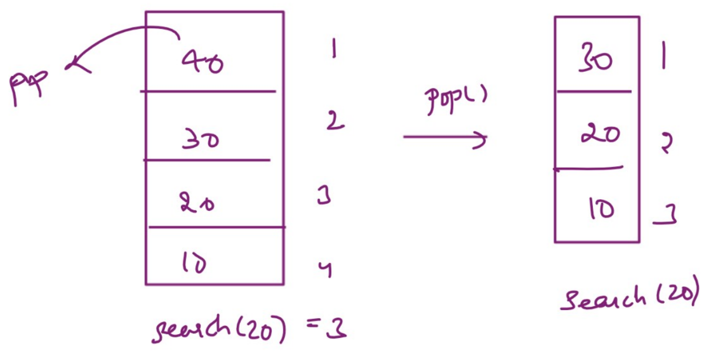
**Methods:**

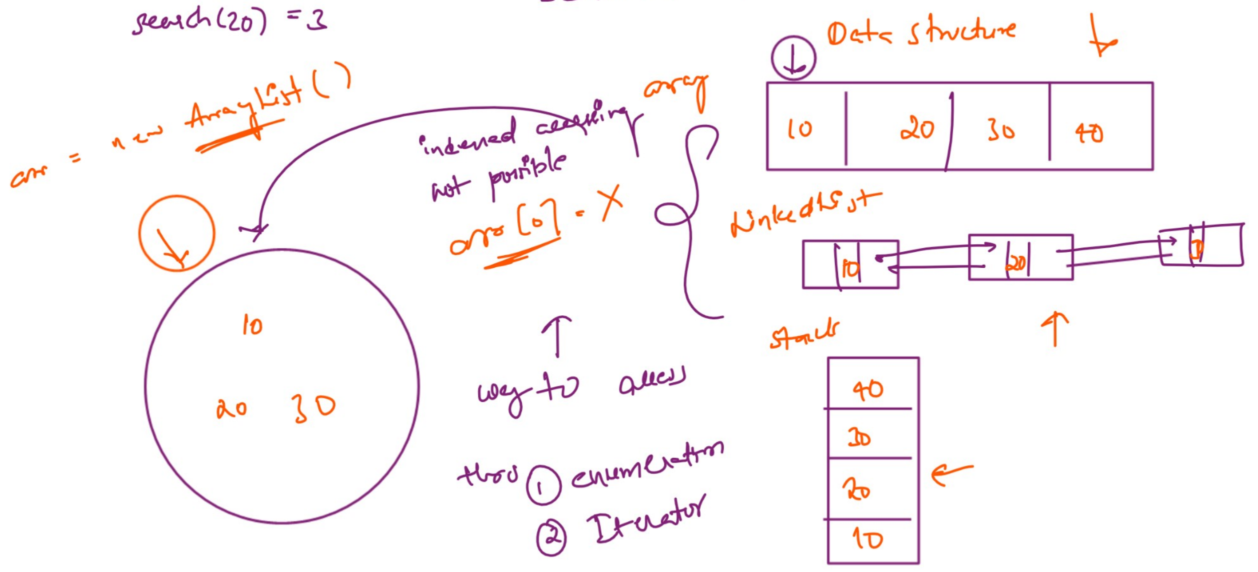
Object push (Object o)
Object pop ()
Object peek ()
boolean empty ()
int search (Object o)



peek () : It returns the element which is present on the top of the stack

search () : It returns the distance of the element that we pass, from the top of the stack





## Enumeration

Enumeration is a cursor, which is used to traverse the collection

→ we can use it for (1) vector
(2) stack

Vector v = new Vector

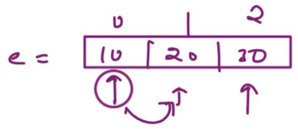
Enumeration $\boxed{e}$ = v.elements ( )
↑

v.elements() is a method present in vector class, when the method is called it internally creates an object of Enumeration type & returns with cursor

Methods ()

① boolean hasMoreElements ( )

② Object nextElement ( )



① → It checks any more element exists in the collection & returns true if exists otherwise false

② It returns the object, where the cursor is present & moves the cursor to the next element

Disadvantage:

→ Only works in legacy classes like vector & stack

Solution:

→ Iterator

# Iterator( )

→ It is present in Iterator (I) not in collection (I)

**properties:**

→ It is used to traverse any/all kind of collections

→ we can get the object of iterator by calling the method iterator()

arr = new ArrayList( )

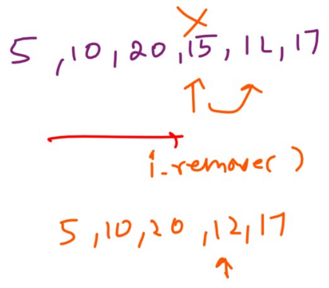
Iterator i = arr.iterator( )

**methods**

hasNext ( )

next ( )

remove ( )



**Disadvantages** (Read only)

(1) only forward traversing is allowed

(2) we can only access the data, but not modify

**Solution**

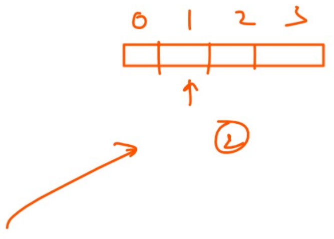
ListIterator

## ListIterator

It is used to traverse the list in both forward & backward direction

**methods**

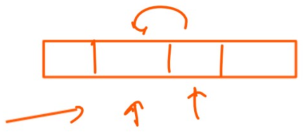
| | |
|---|---|
| boolean | hasNext ( ) |
| Object | next ( ) |
| int | nextIndex ( ) |



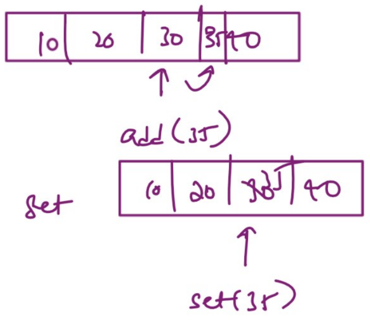
boolean hasPrevious ( )

Object previous ( )

int previousIndex ( )



void { add ( Object o )
set ( Object o )
remove ( ) }



```
Scanner  sc = new Scanner(System.in);
     System.out.println("Enter values");
     while(sc.hasNextInt()) { //10 20 30 40 50 exit
        al.add(sc.nextInt());
     }
```

Note: here scanner object sc will iterate over the values from the keyword until it encounters non-integer value.