**Collection Framework – Part\_05**

* **Differences between ArrayList and LinkedList:**

|  |  |  |
| --- | --- | --- |
| S.No | ArrayList | LinkedList |
| 1 | ArrayList is the best choice, if our frequent operation is retrieval. | LinkedList is the best choice if our frequent operation is insertion or deletion in the middle. |
| 2 | ArrayList is the worst choice if our frequent operation is insertion or deletion in the middle. Because, internally several shift operations are performed. | LinkedList is the worst choice if our frequent operation is retrieval. |
| 3 | In ArrayList the elements will be stored in consecutive memory location and hence retrieval operation will become easy. | In LinkedList the elements won’t be stored in consecutive memory location and hence the retrieval operation will become complex. |

* **Vector:**

1. The underlying data structure is resizable array or growable array.
2. Insertion order is preserved.
3. Duplicates are allowed.
4. Heterogeneous objects are allowed.
5. null insertion is possible.
6. It implements Serializable, Cloneable and RandomAccess interfaces.
7. Every method present in the Vector is synchronized andh hence Vector object is thread-safe.

**Constructors:**

**Vector v = new Vector();**

Creates an empty Vector object with default initial capacity 10.

Once Vector reaches its max capacity, then a new Vector object will be created with

New capacity = Current capacity \* 2

**Vector v = new Vector(int initialCapacity);**

Creates an empty Vector object with specified initial cpactiy.

**Vector v = new Vector(int initialCapacity, int incrementalCapacity);**

**Vector v = new Vector(Collection c);**

Creates an equivalent Vector object for the given Collection. This constructor meant for interconversion between Collection objects.

**Vector specific methods:**

To add objects:

add(Object o); 🡪 C

add(int index, Object o); 🡪 L

addElement(Object o); 🡪 V

To remove objects:

remove(Object o) 🡪 C

removeElement(Object o) 🡪 V

remove(int index) 🡪 L

removeElementAt(int index) 🡪 V

clear() 🡪 C

removeAllElement() 🡪 V

To get objects:

Object get(int index) 🡪 L

Object elementAt(int index) 🡪 V

Object firstElement() 🡪 V

Object lastElement() 🡪 V

Other methods:

int size();

int capacity();

Enumeration elements();

Example:

import java.util.\*;

class VectorDemo{

public static void main(String[] args){

Vector v = new Vector();

System.out.println(v.capacity()); //10

for(int i=0;i<=10;i++){

v.addElement(i);

}

System.out.println(v.capacity());//10

v.addElement(“A”);

System.out.println(v.capacity());//20

System.out.println(v);//[1,2,….,10,A]

}

}

* **Stack:**

It is the child class of Vector.

It is a specially designed class for Last In First Out [LIFO] order.

**Constructor:**

Stack s = new Stack();

**Methods:**

Object push(Object o);

To insert an object into the Stack.

Object pop(Object o);

To remove and return top of the Stack.

Object peek()

To return top of the Stack without removal.

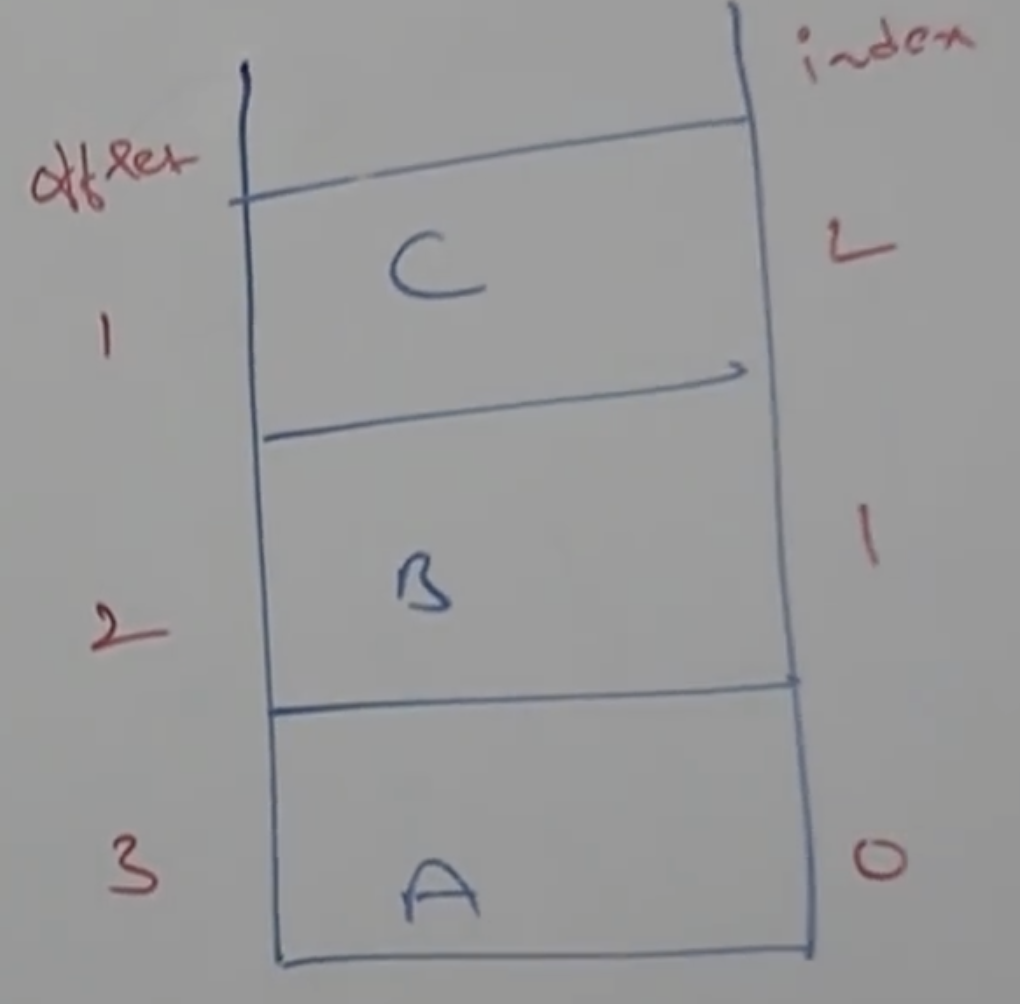
boolean empty()

Returns true, if the stack is empty.

int search(Object o)

Returns offset if the element is available otherwise returns -1

Index vs offset:



Index 🡺 Where the stack elements are located.

offset 🡺 Where the element located from top of the stack.

Example:

import java.util.\*;

class StackDemo{

public static void main(String[] args){

Statck s = new Stack();

s.push(“A”);

s.push(“B”);

s.push(“C”);

System.out.println(s); //[A,B,C] insertion order preserved.

System.out.println(s.search(“A”)); //3

System.out.println(s.search(“Z”)); //-1

}

}

* **The 3 cursors of Java:**

If we want to get objects one by one from the Collection. Then, we should go for cursor.

There are three types of cursors available in Java.

1. Enumeration
2. Iterator
3. ListIterator

* **Enumeration:**

We can use Enumeration, to get objects one by one from legacy collection object.

We can create Enumeration object by using elements() method of Vector class.

public Enumeration elements();

Enumeration e = v.elements();

v 🡪 Vector object.

**Methods:**

public boolean hasMoreElements();

public Object nextElement();

**Example:**

import java.util.\*;

class EnumerationDemo{

public static void main(String[] args){

Vector v = new Vector();

for(int i=0;i<=10;i++){

v.addElement(i);

}

System.out.println(v);

Enumeration e = v.elements();

while(e.hasMoreElements){

Integer I = (Integer)e.nextElement();

if(I%2 ==0){

System.out.println(I); [0,2,4,6,8,10]

}

}

}

}