Collections in java

Why do we need collections

1.Arrays are fixed in the size ,size of the array cannot be changed once it’s declared

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** [] a=**new** **int** [10]; // if you just remove the number there will br an error

// a[1]="prasanna"; cannot be done

a[1]=15;

}

Can only add 10 elements you cannot add 11th element and if you add 5 elements then 5 elements size will waste

2.Arrays can hold only same type of the elements // see above piece of the code

3. arrays are not implemented on any general data structure that means arrays are not giving any in built method play with arrays like contains,add etc ...

To overcome all these problems we need to use collections

Difference between arrays and collections

|  |  |
| --- | --- |
| Arrays | collections |
| Wrt to memory arrays not recommended  As memory will be lost if we declare 100 size and enter only 2 | Wrt memory collections are recommended |
| Wrt performance arrays are recommended | Wrt performance collections are not recommended as addition of one element new memory will be created and old elements will be copied to newly created memory |
| Arrays can contain primitive and object data types ex : can contain int,integer | Collections can only contain object data types means integer not int |

What is collection ?

Group of individual objects represented as single entity is called collection

What is collection Framework?

To represent the collections there are several interfaces and classes are there these are called collection framework

Collection interface

This interface contains common methods which all collections can have for example

Add element

Delete element

Contains....

This is considered as the root interface of collection framework

What is collections

Collection is the interface but collections is class

Collections class defines some utility method like sort, searching for the collection objects

List interface

1.List is the child of the collection interface

2.if you want to re-present group individual objects as single entity where duplicate elements are allowed and insertion order need to be maintained then list need to be used

What are different implementation classes of list ?

1.Array List

2.Linked List

3.Vector (stack is sub class of the vector class)

Set interface

1.child interface of collection

2. if you want to re-present group individual objects as single entity where duplicate elements are not allowed and insertion order need not to be maintained then list need to be used

What are different implementation classes of set?

1.Hashset (LinkedHash set is the subclass of hashset )

Sorted set

is the child interface of the set interface this is used when in collection elements need to inserted in some order

navigable set

this is the child interface of the sorted set this interface has some methoeds which will help in the navigation in the collection

**class that implements this navigable set is Treeset**

Queue interface

I don’t want to concentrate on this subject just skip this

Map interface

If you want to represent group individual object as key value pair then we need to use map interface

Key cannot have duplicate elements

This is not child collection interface

What are different implementation classes of Map?

1.Hash map (Linked has map is child class of map)

2.weakhaspmap

3.Identity has map

4.hashtable ( this has table extends the Dictonary abstract class and has child class as properties)

Sorted map

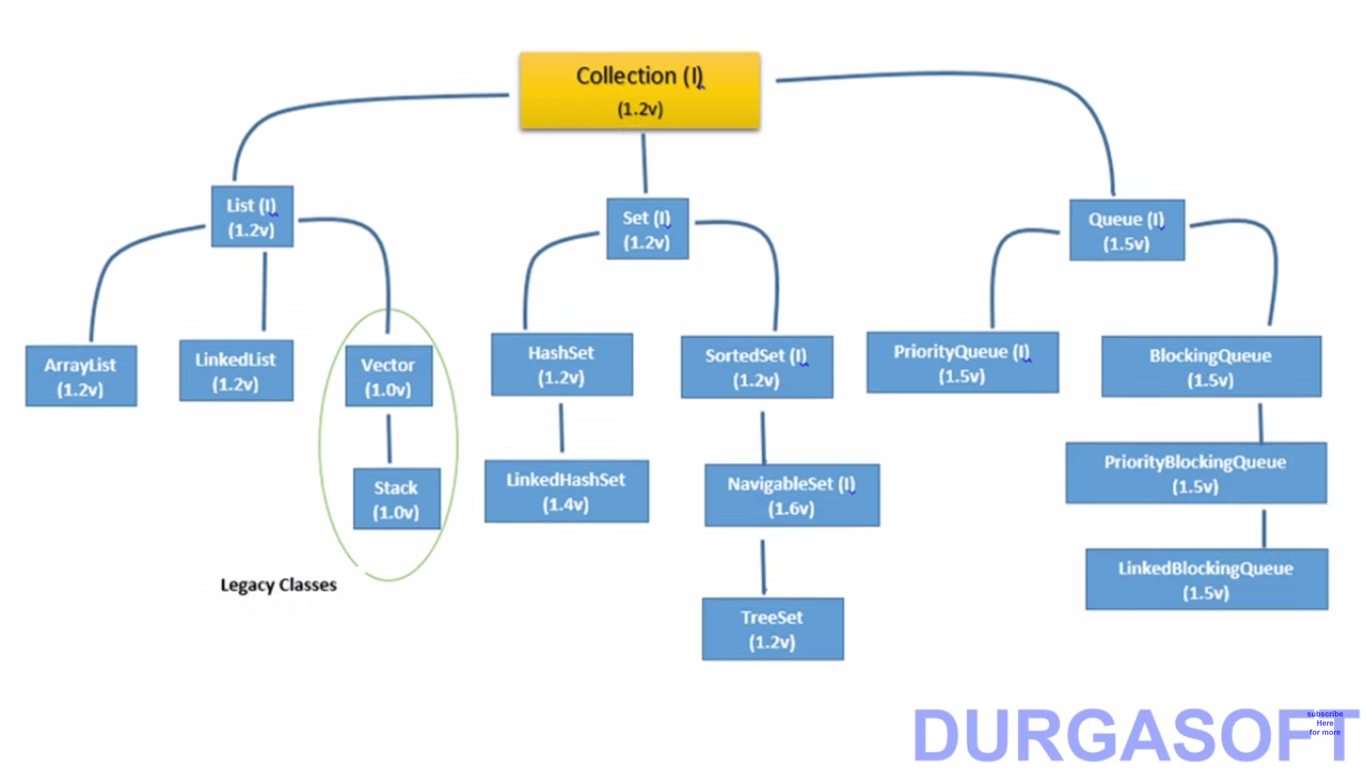
is the child interface of the map interface this is used when group of elements need represented in key value pairs and elements are inserted is some sort order of the keys then sorted maps need to be used

navigable map

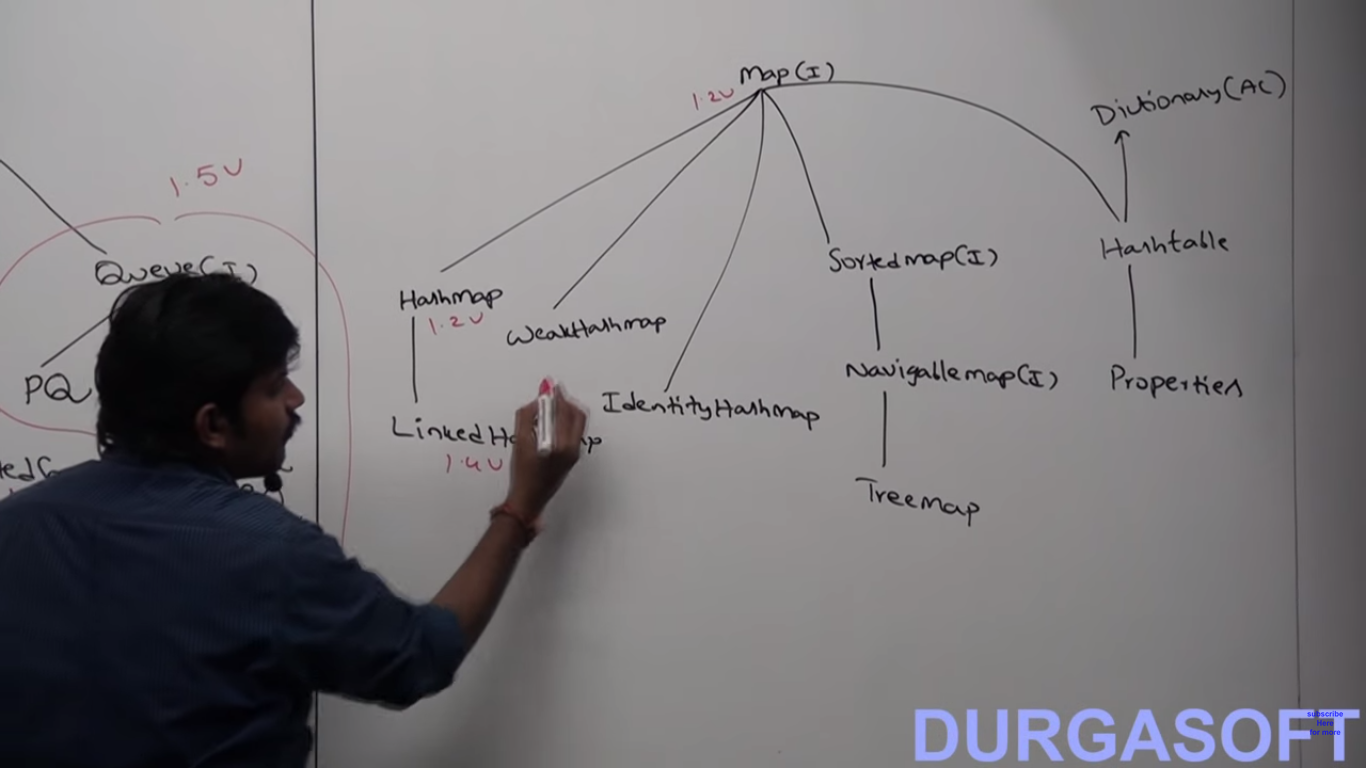
this is the child interface of the sorted map this interface has some methoeds which will help in the navigation in the collection

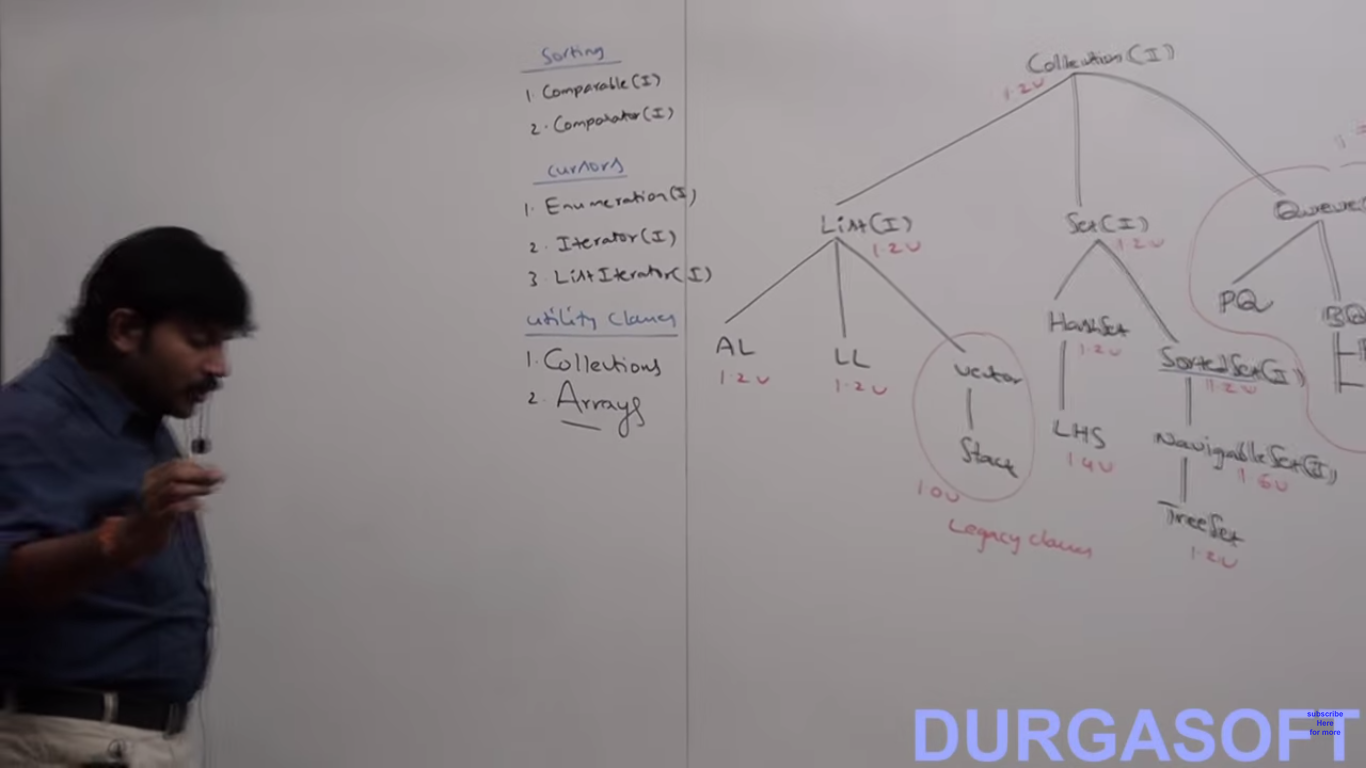
**class that implements this navigable set is Treemap**

"**A picture is worth a thousand words**"



**Map**

****



Comparable Interface

If u want to have natural sorting order then we need to use comparable interface

Comparator Interface

If you want to use your own sorting order then we need to use comparator interface

Cursors

If you want run through the collection then u need to use cursor then you should use cursor

There are 3 types of cursor

Enumeration

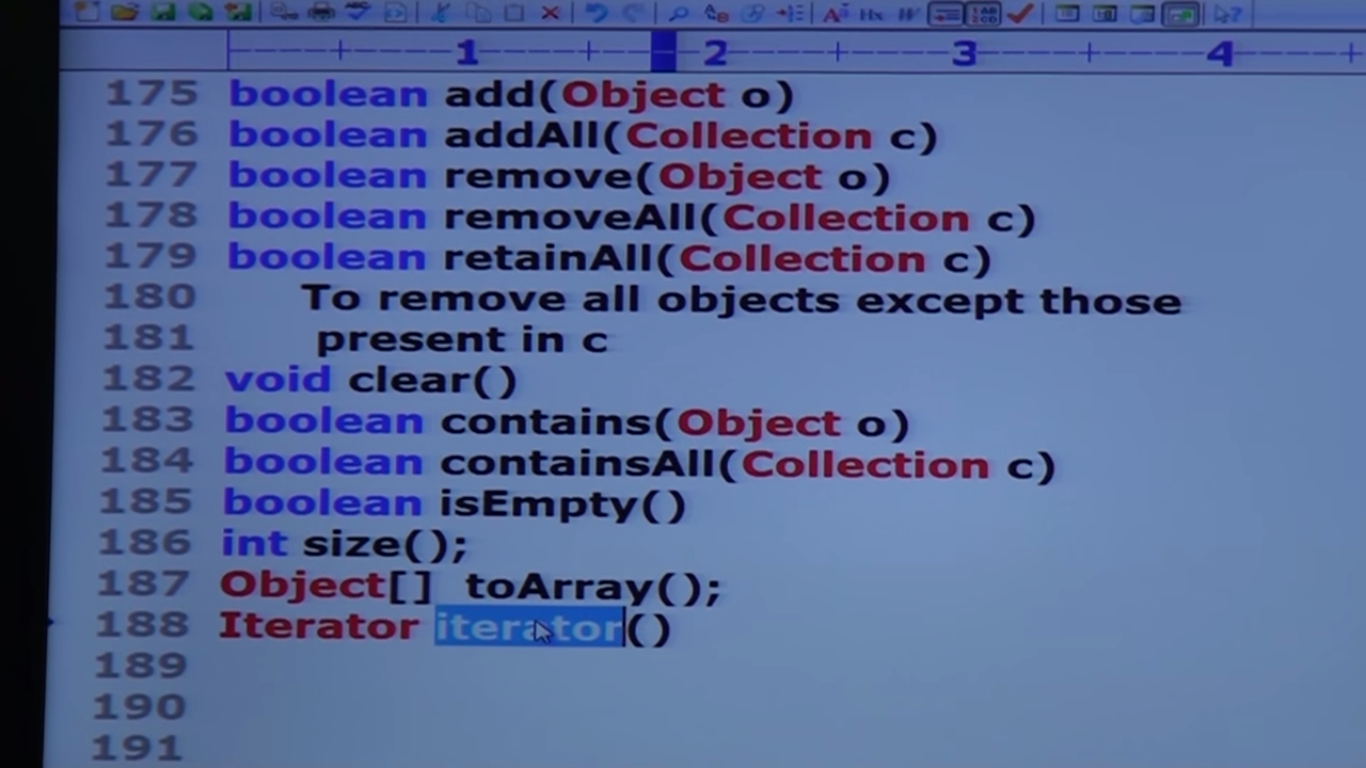
Iterator

Listiterator

Utility classes

Collections and arrays are 2 utility classes

Important method of collection interface



Note:

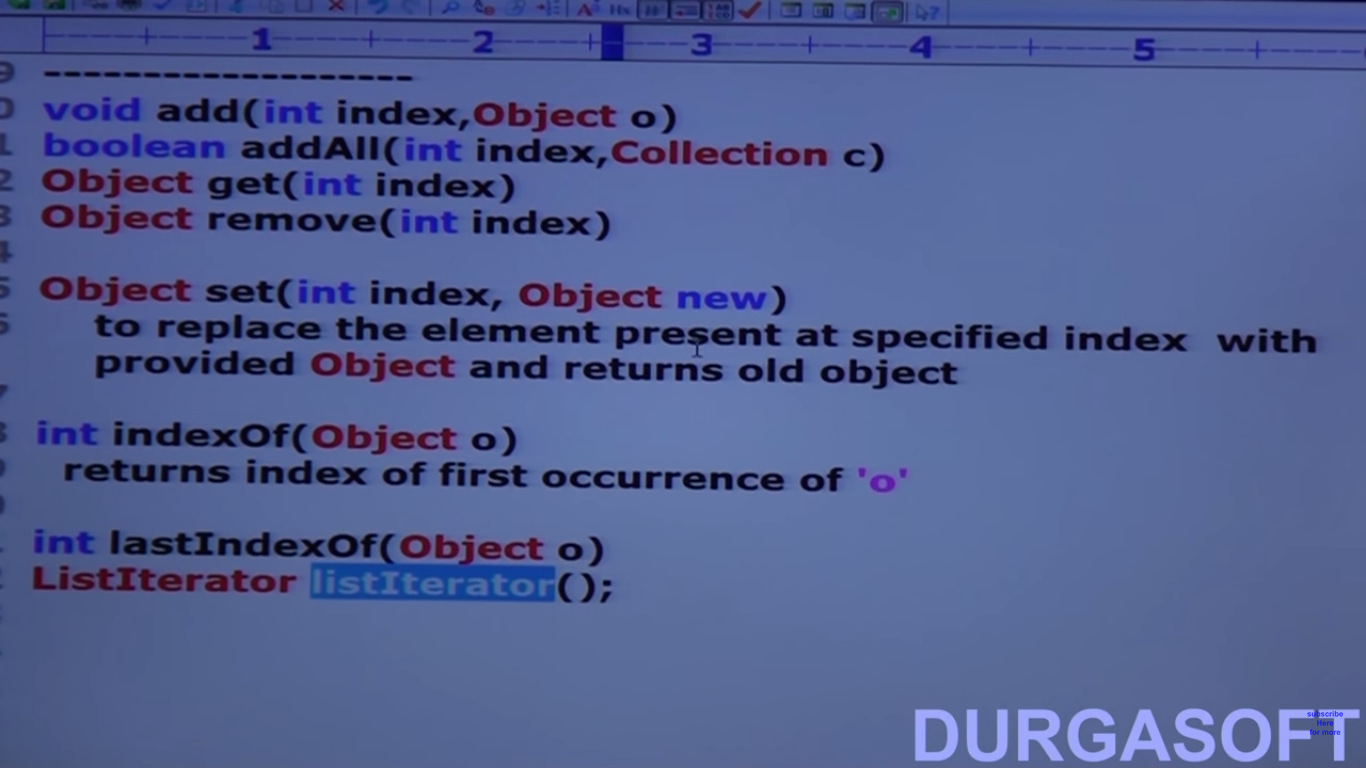
Collection interface doesn’t contain any method to retrieve the object there is no concrete class which would directly extend this collection interface

My guess : there are many interface that extends the collection interface then are many class which would extend these sub interfaces

Detailed study of the List

In the list we allow duplicate objects and insertion order is retained so now the question is how do we identify the duplicate elements in the List ? answer for this question is Index, by using the index we will able to identify the duplicate elements **SO INDEX PLAY IMP ROLE IN LIST**

**What are the common method in the LIST INTERFACE ?**

****

Example of addall

List<Integer> l2=**new** LinkedList<>();

l2.add(50);

l2.add(60);

List<Integer> list1=**new** ArrayList<Integer>();

list1.add(10);

list1.add(20);

list1.add(60);

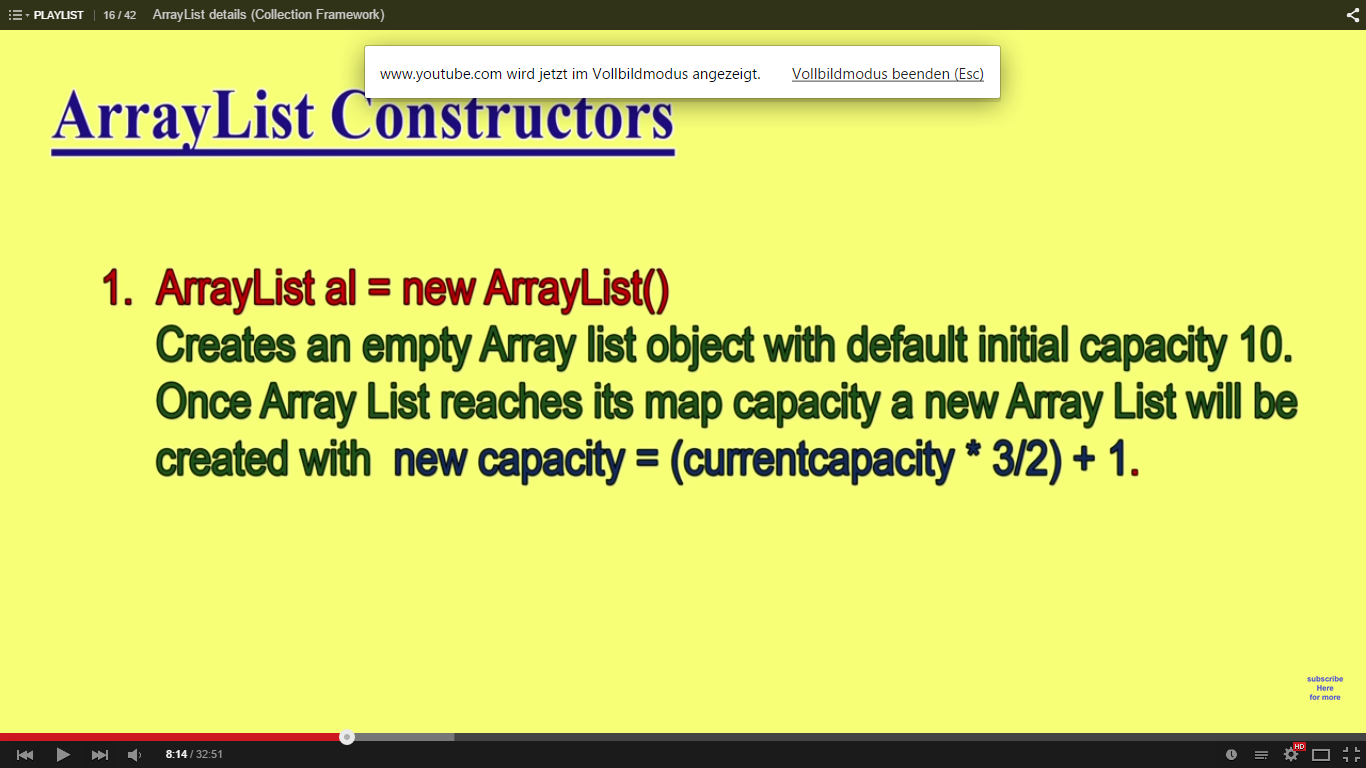
list1.addAll(l2);

Array List class

1. Hetrogeneous elements insertion is possible [Except treeset and treemap all places null heterogeneous elements are allowed]

2. null insertion is possible

3. undelying data structure is Resizable array or grow able array



**Constructor in Array list**

Arraylist al=new arraylist(int capacity) // which set the dafult capacity to the given value

Arraylist al=new Arryalist(collection c )//

**Sample program:**

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

ArrayList a=**new** ArrayList();

// default capacity is 10 when new element is added then new capacity will=(cc\*3/2)+1

a.add("a");

a.add(10);

a.add(**null**);

System.*out*.println(a);// [a, 10, null]

a.add(2,"K");

a.add("N");

System.*out*.println(a);// [a, 10, K, null, N]

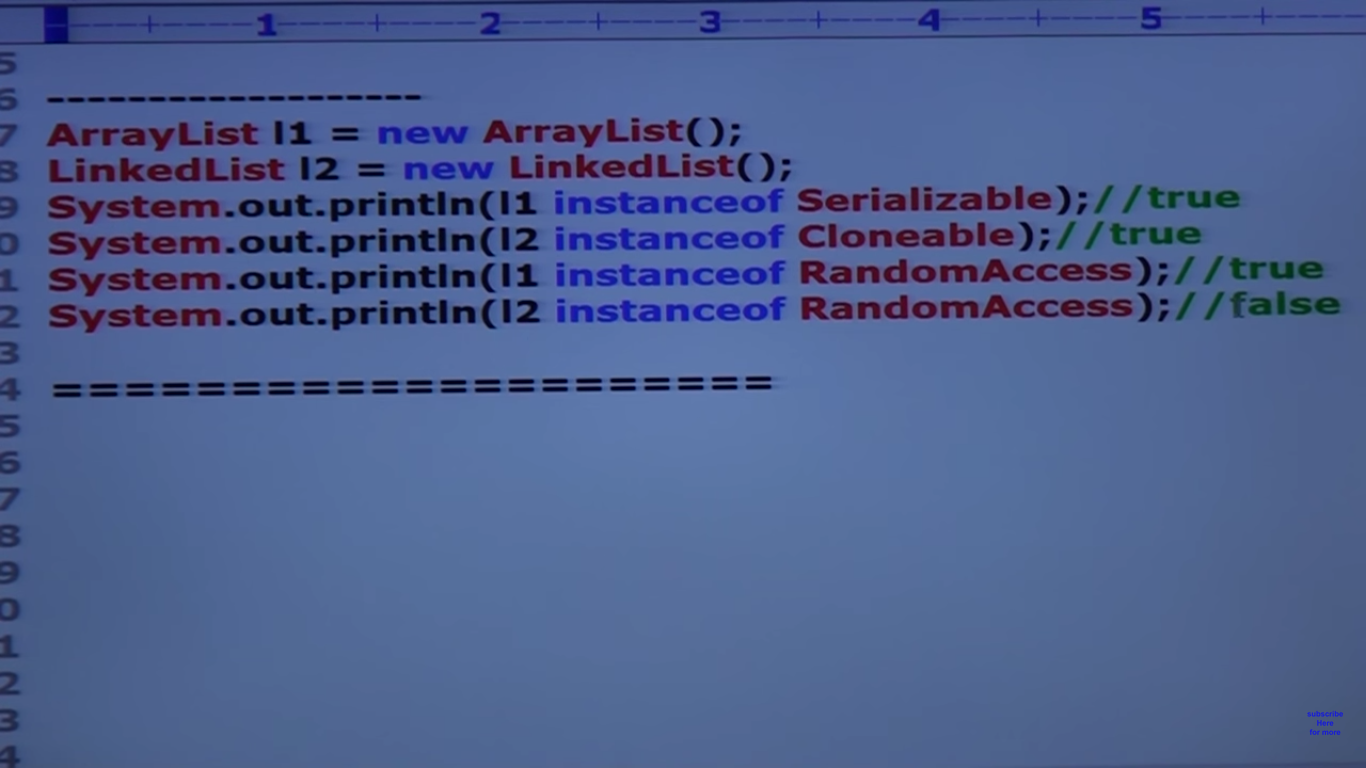
**Sereializable**

To transfer the object over the network class must implement serializable interface once it’s received at receiver copy of the same object will be created this is also called cloning so all these collection class implements clone able interface and serializable

Note: Arraylist and vector implements Random access interface advantage of this is any element can be with same speed that means be it 100th or 1000th element in arraylist same time is required

Note:Random **access is marker interface it doesn’t contain any method**

**Tricky program**



**Imp notes:**

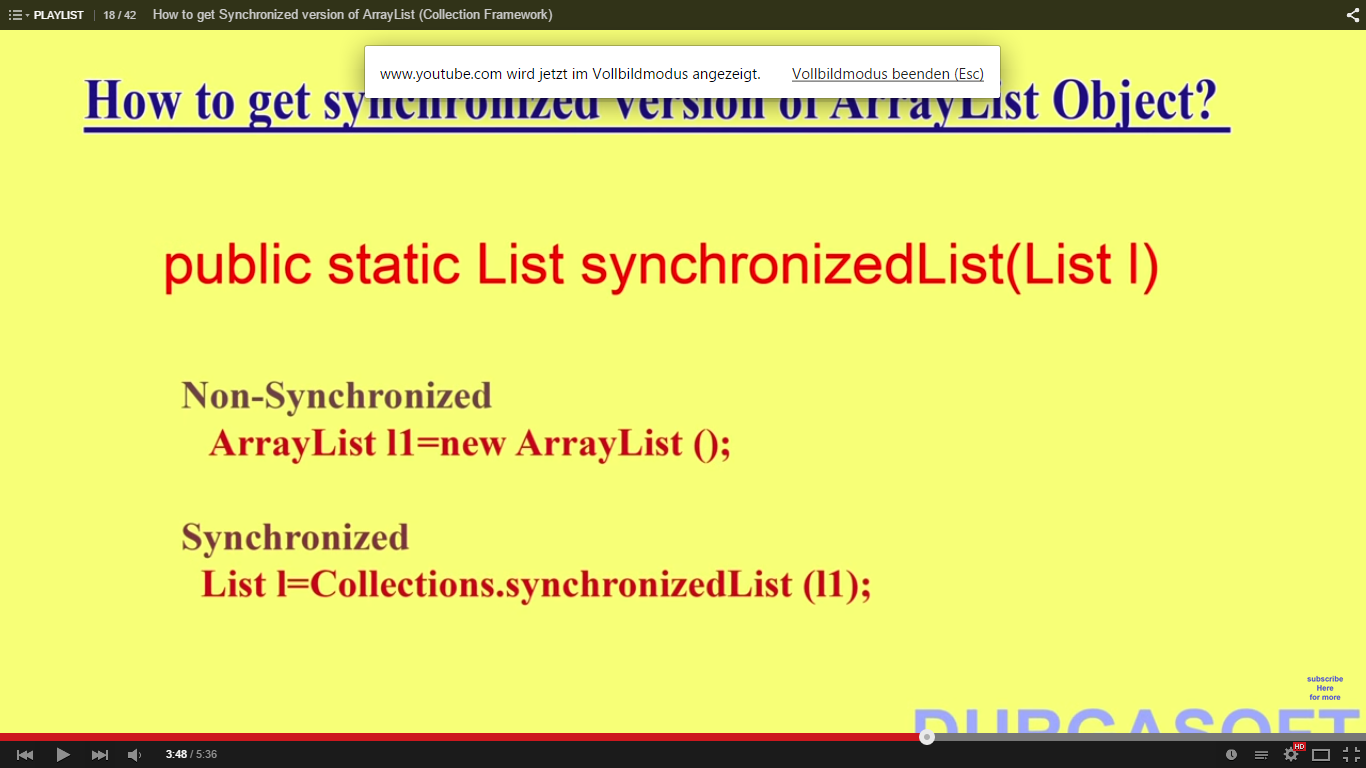
If our operation is mainly the retrieving the data from list then arraylist is best because it’s implements random access interface

If our operation is insertion at middle of the list or removal at the middle [after removing/inserting at the middle would have shift the elements in the list ] so this would take more time

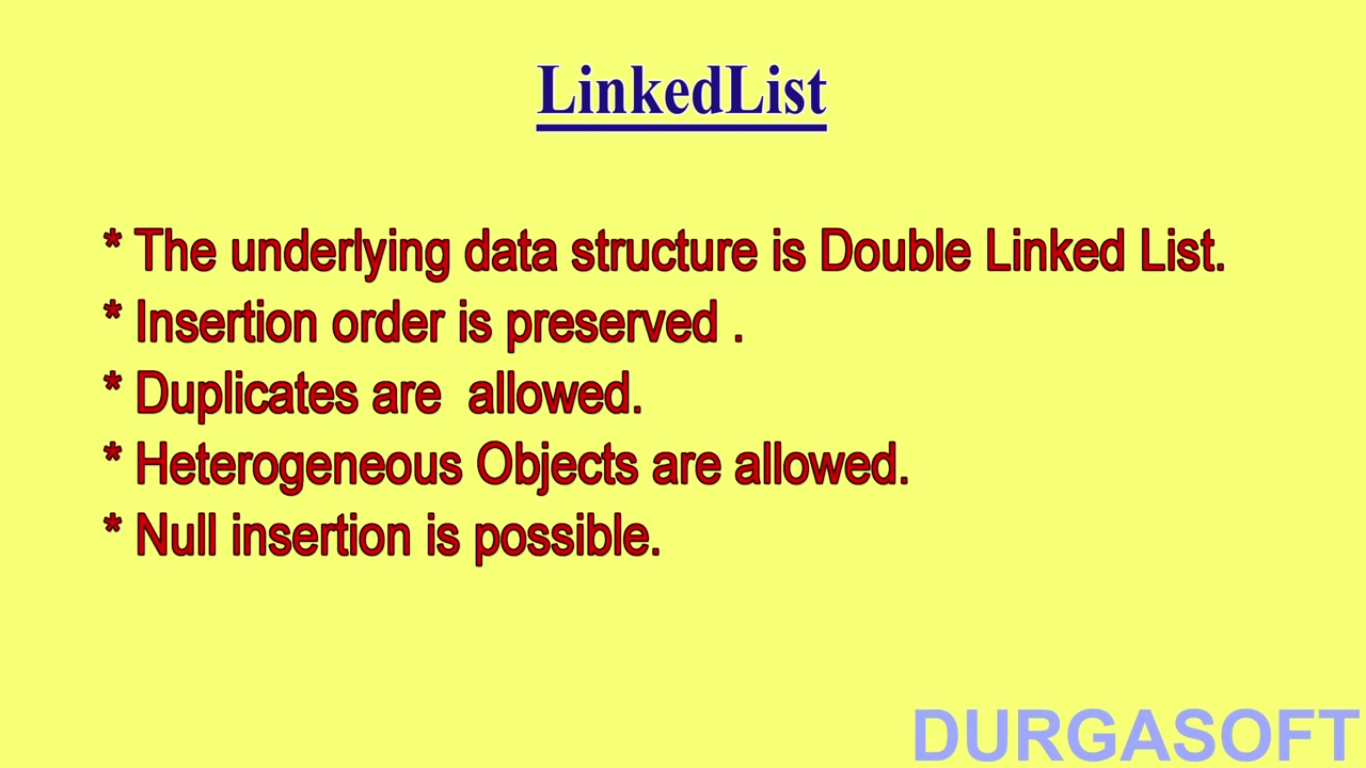
**Difference between arraylist and vector**

|  |  |
| --- | --- |
| **Arraylist** | **vector** |
| Most of methods are not synchronized | Methods are synchronized |
| Because of above reason it’s not thread safety that means multiple threads can accesses object | This is thread safety that means only thread can access this object |
| Becaue of above reason performance will be high | Because of above reason performance will be low |
| It’s introduced in 1.2v java not legacy | It’s introduced in 1.0 it’s legacy |

****

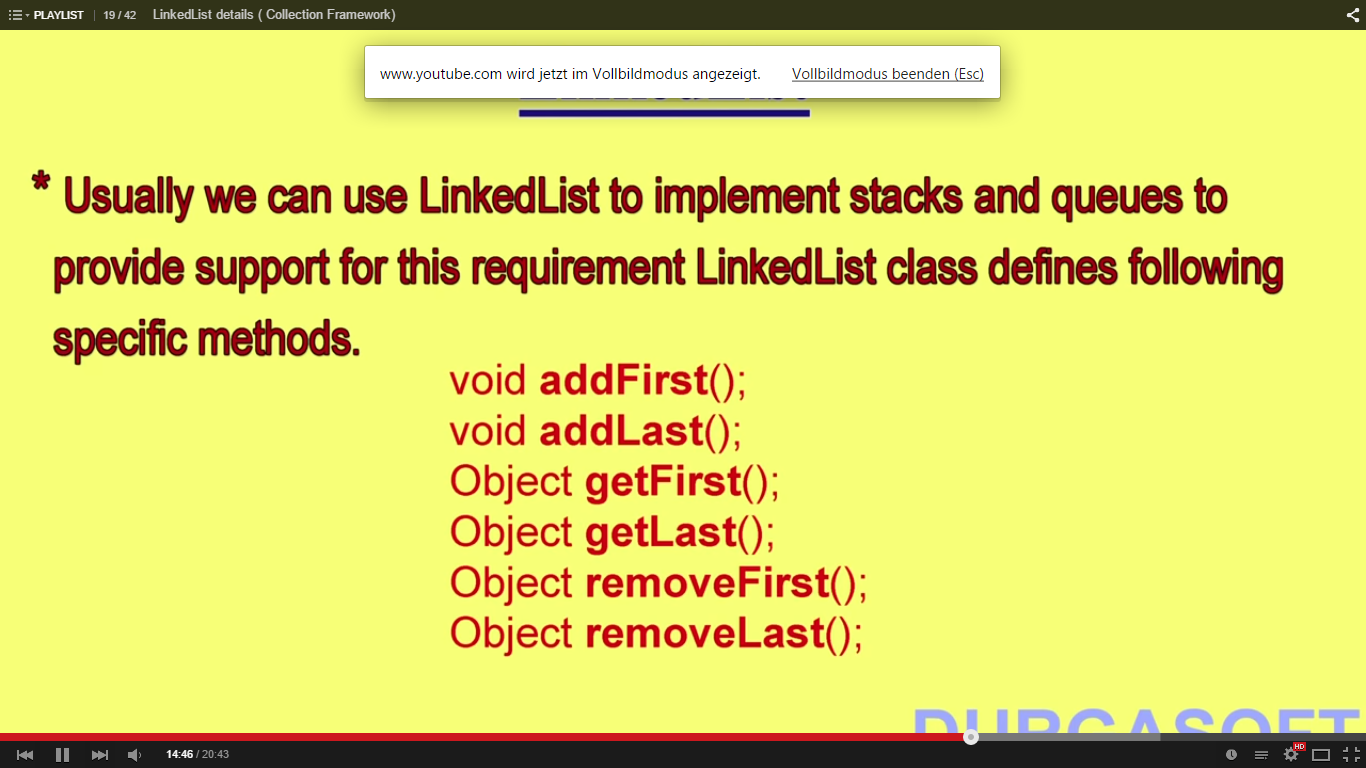
****

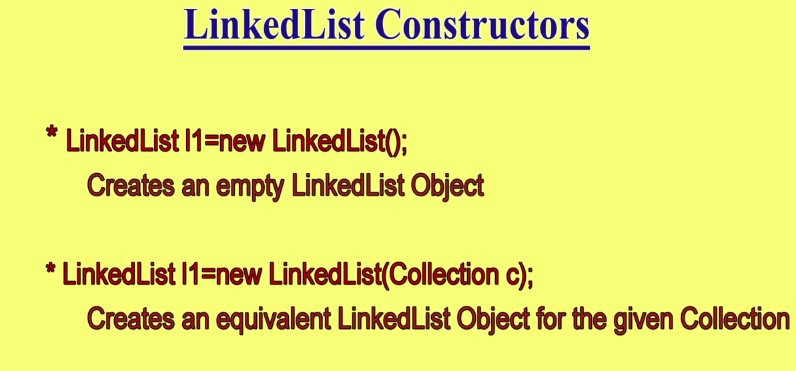
**Linked List in Detail**

****

If our operation is insertion or deletion at the middle then linked list is best solution

If our operation is retrieving the data then this worst solution



****

**Sample program**

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

LinkedList l1=**new** LinkedList();

l1.add("prasanna");

l1.add("gouda");

l1.add("shankar");

System.*out*.println(l1); [prasanna, gouda, shankar]

l1.set(1, "santya");

System.*out*.println(l1); [prasanna, santya, shankar]

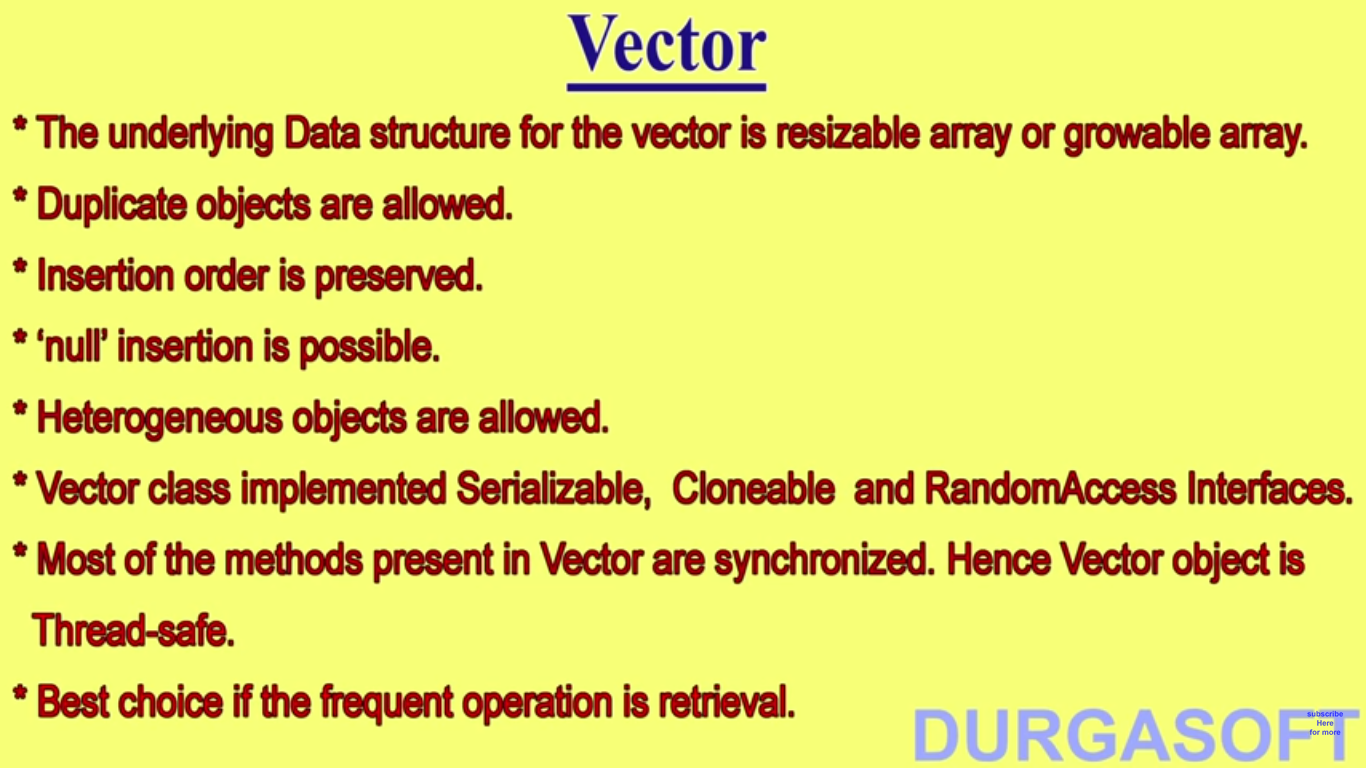
l1.add(2, "pinky");

System.*out*.println(l1); [prasanna, santya, pinky, shankar]

}

**Null can also be inserted**

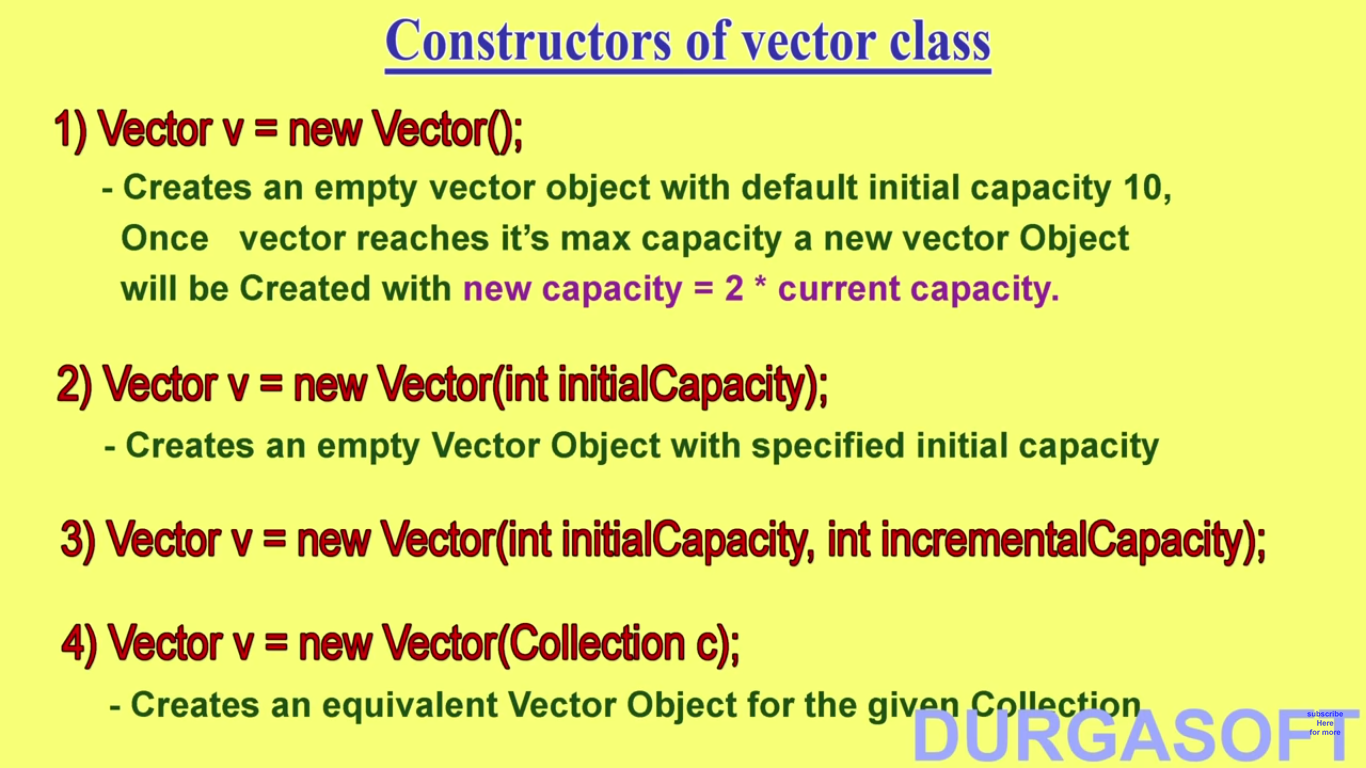
**Vector**

****

**Methoeds specific to vector**

Skipped

**Constructors in vectors**

****