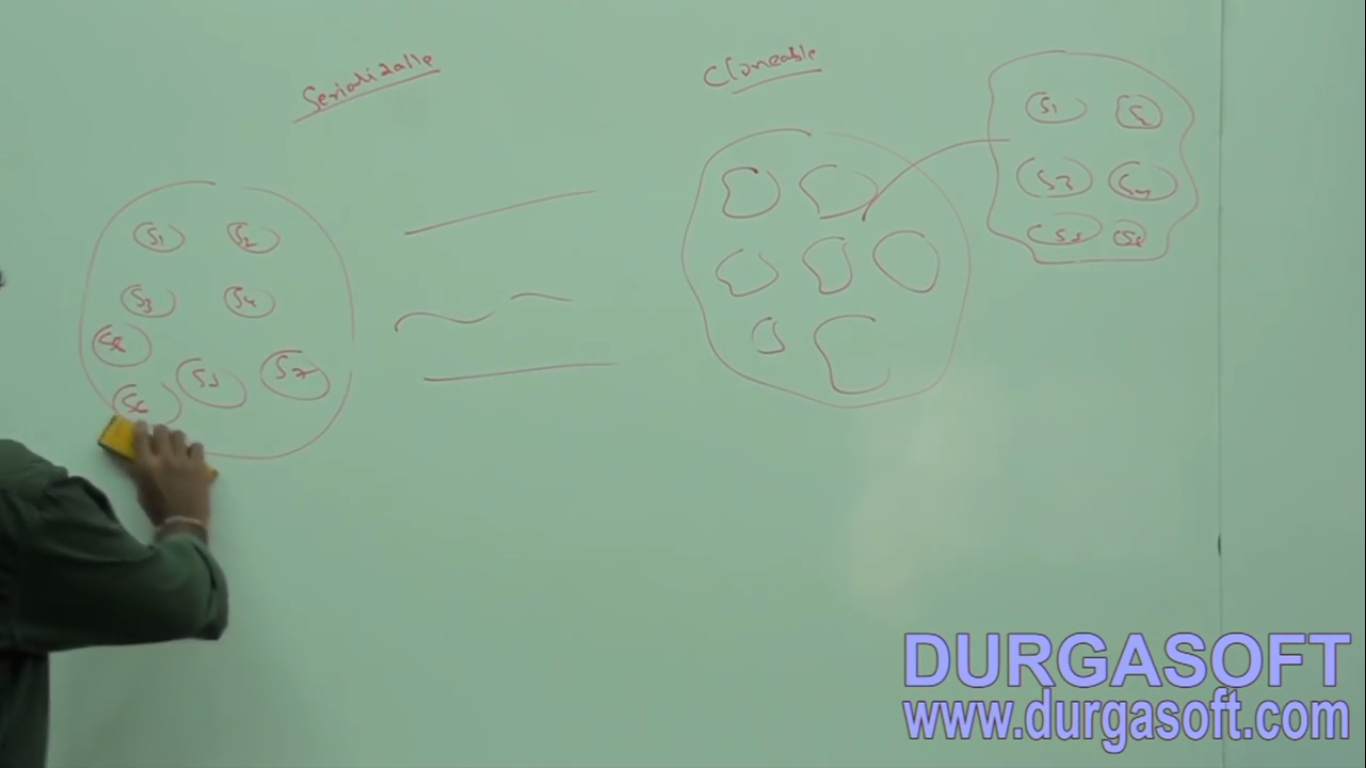
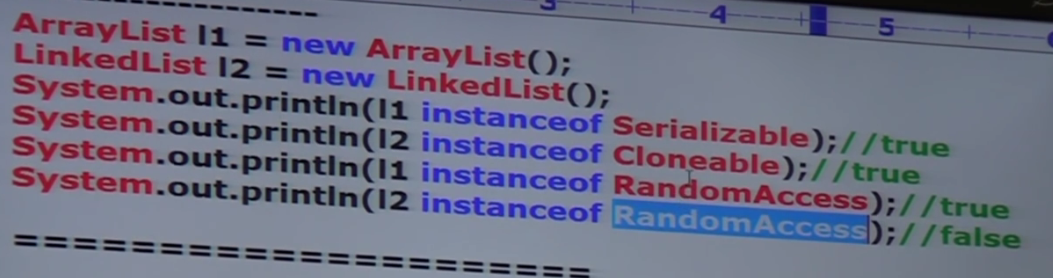
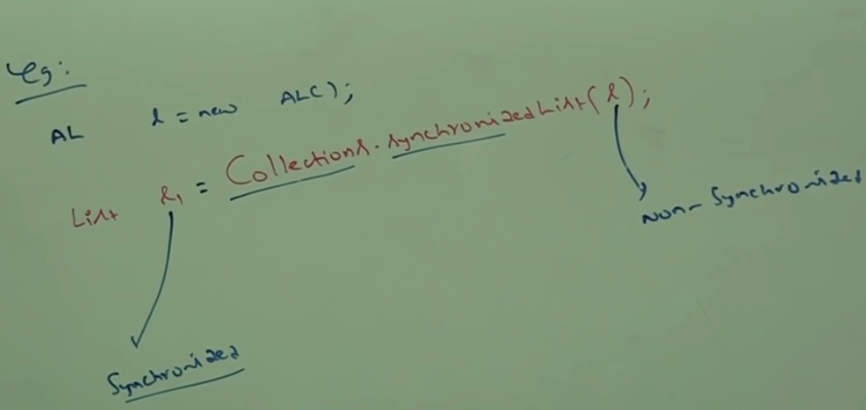
1. Usually we use collection to hold and transfer objects from one location to another location. To provide support for this requirement, every location class implements **Serializable** interface.  
   Every Collection implements **Cloneable** interface too. Because after reaching the destination, if some operation goes wrong with the collection, again we need to request the source for the collection. So to avoid this overhead, we immediately clone the reached collection and perform operation on cloned one.   
   

RandomAccess(I)

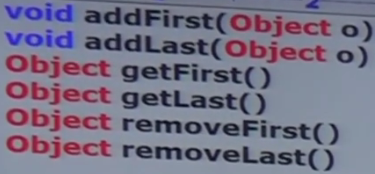
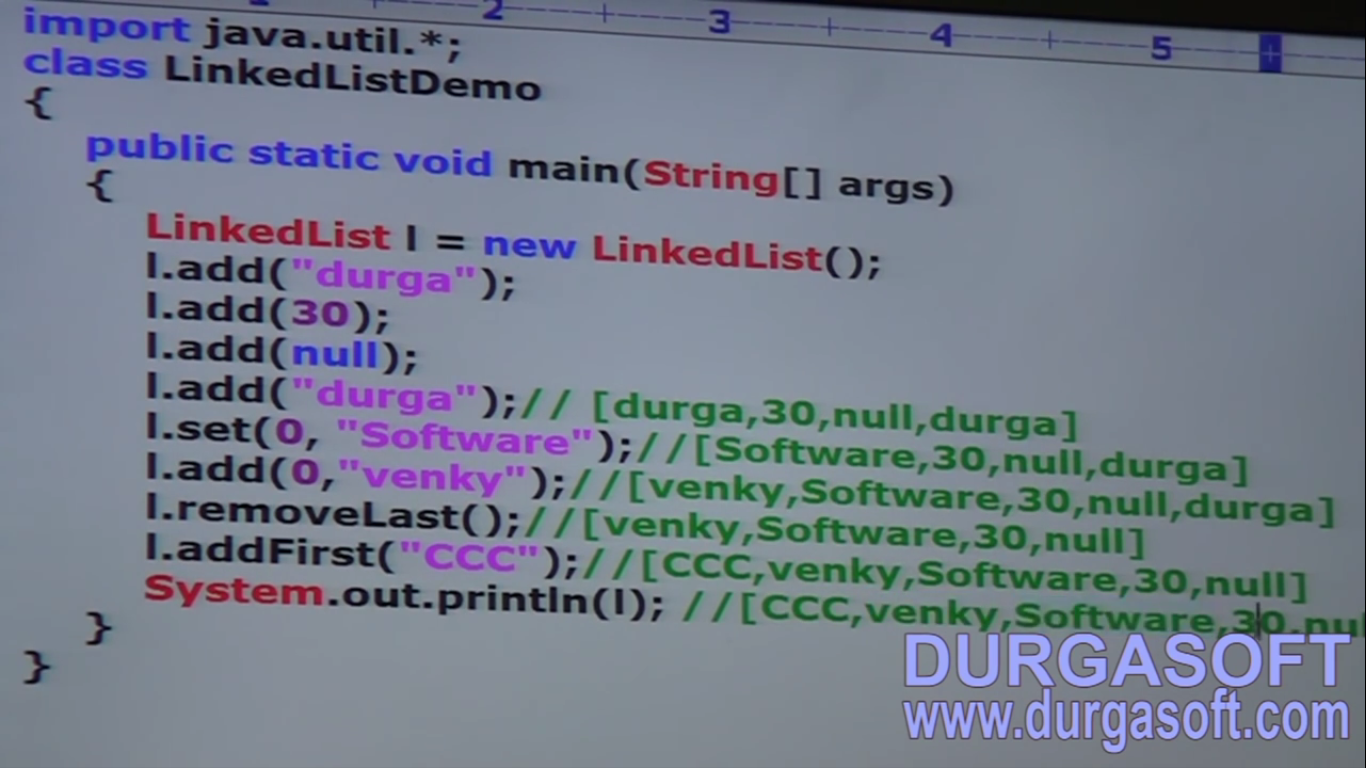
1. ArrayList and Vector classes implement **RandomAccess** interface. So that any random element, we can access with the same speed.
2. In java.util package.
3. **Marker interface** where required ability will be provided automatically by the JVM.
4. 
5. **ArrayList**:
   1. ArrayList is the **best choice** If our frequent operation is **retrieval operation** because ArrayList implements **RandomAccess**(I).
   2. ArrayList is the **worst choice** if our frequent operation is “**insertion or deletion**” in the middle as many **shift operations** are required.
6. **Solution**: LinkedList
7. Before that we talk about something related to interview room.

# Difference b/w ArrayList and Vector

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| Every method is **non-synchronized** |  |
| At a time, Multiple threads are allowed to operate, hence it’s **not thread-safe**. | Only one thread is allowed to operate, so **thread-safe** |
| Relatively **performance wise high**, because threads don’t require for waiting. | **Performance wise low**, threads have to wait |
| **Non-Legacy** as introduced in 1.2v | **Legacy** as introduced in 1.0v |
| **Jatin->** We can’t specify the incremental capacity in constructor. | Incremental Capacity can be specify in constructor |
| **Capacity** can’t be found out as there is no method | Capacity can be found with method capacity() |

Nowadays, in interview, interviewers don’t ask “difference b/w ArrayList and Vector”, they ask “We want thread-safety and we want to use ArrayList. How to get thread-safely version of ArrayList?”  
**How to get synchronized version of ArrayList object?**🡪By default, ArrayList is **non-synchronized**, but we can get synchronized version of ArrayList object by using **synchronizedList()** static method of **Collections** class.   
**syntax**:  
  
  
Similarly, we can get synchronized version of Set and Map objects by using the following methods of **Collections** class.  


LinkedList(C)

1. **Properties**:
   1. **Underlying Data** Structure: Doubled Linked List.
   2. **Insertion Order**: Preserved.
   3. **Duplicate objects**: Allowed
   4. **Heterogenous objects**: Allowed
   5. **Null Insertion**: Allowed
   6. **Implemented Interfaces**: Serializable, Cloneable but not RandomAccess
   7. **Application**: If our frequent operations are insertion and deletion in the middle  
      **Worst Choice**: If frequent operations are retrieval operations.
2. **Constructors**:   
   NOTE: Capacity/initial Size concept is not applicable onto LinkedList as LinkedList elements are not stored in continuous memory location.   
   ArrayList contains 3 constructors so LinkedList contains 2 methods as initial capacity doesn’t apply to it.
   1. **LinkedList():** Creates an empty LinkedList object.
   2. **LinkedList(Collection)**: Creates an equivalent linked list object for the given collection.
3. **Method:** LinkedList class specific methods.  
   Usually, we can use LinkedList to develop stacks and queues. To provide support for this requirement, LinkedList class defines the following specific methods.
   1.   
      These 6 methods are applied only on LinkedList.
4. **Example**:  
   
5. **d**
6. d
7. d