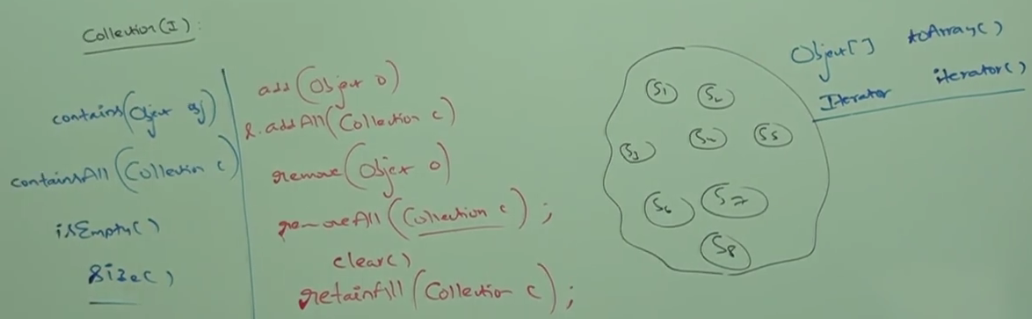
Collection(I)

1. If we want to **represent** a group of individual objects as a **single entity**, then go for collections.
2. **Collection interface** defines the most common methods which are applicable for any collection.
3. **Methods:**
   1. add(Object o)
   2. addAll(Collection c)
   3. remove(Object)
   4. removeAll(Collection c)

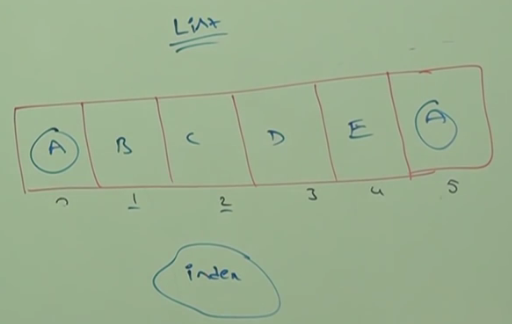
kinds of remove

* 1. clear()
  2. retainAll(Collection c) : Opposite to removeAll(Collection)
  3. contains(Object)
  4. containsAll(Collection)
  5. isEmpty()
  6. size()
  7. **Object[] toArray()**: We have performed all insertions and deletion operations on Collection so utilized memory and flexibility. Now for performance purpose, convert into array.
  8. **Iterator iterator()**: To get element one by one. We need a cursor.

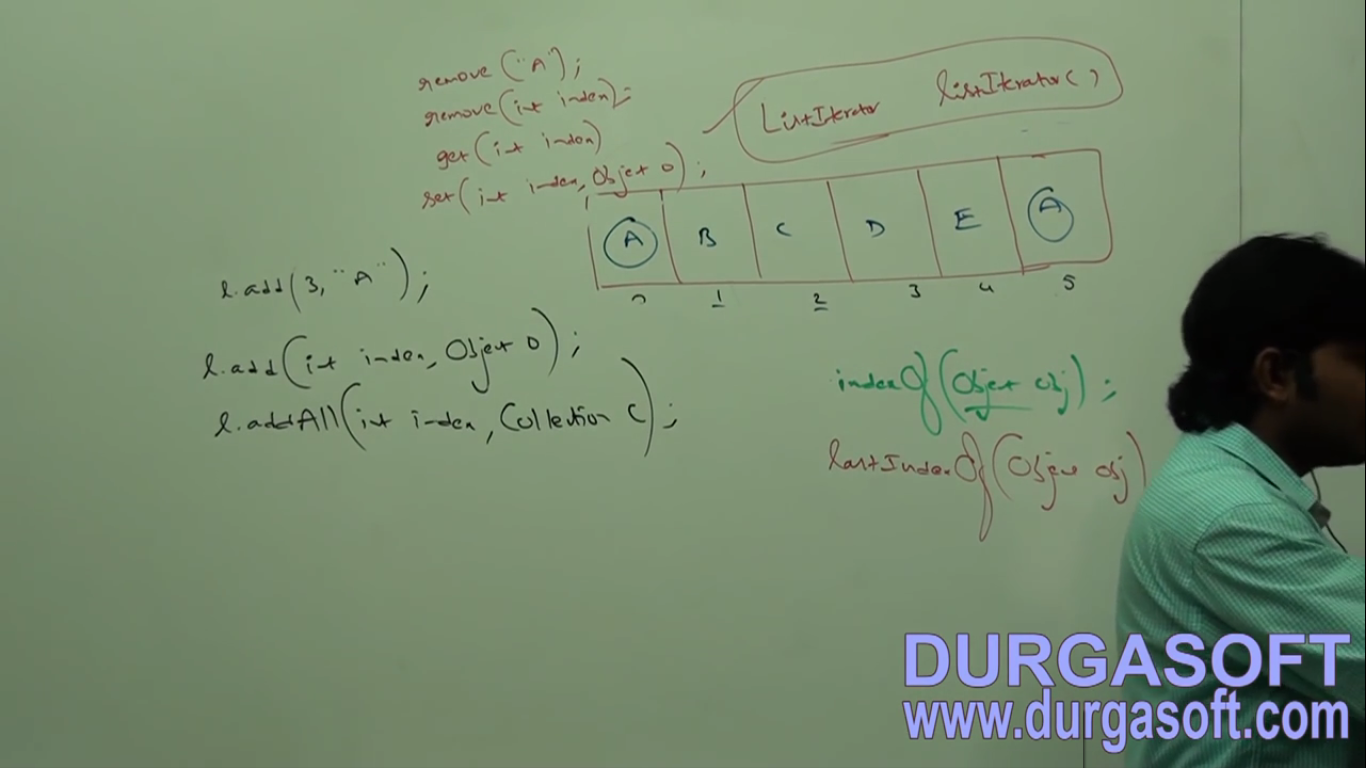
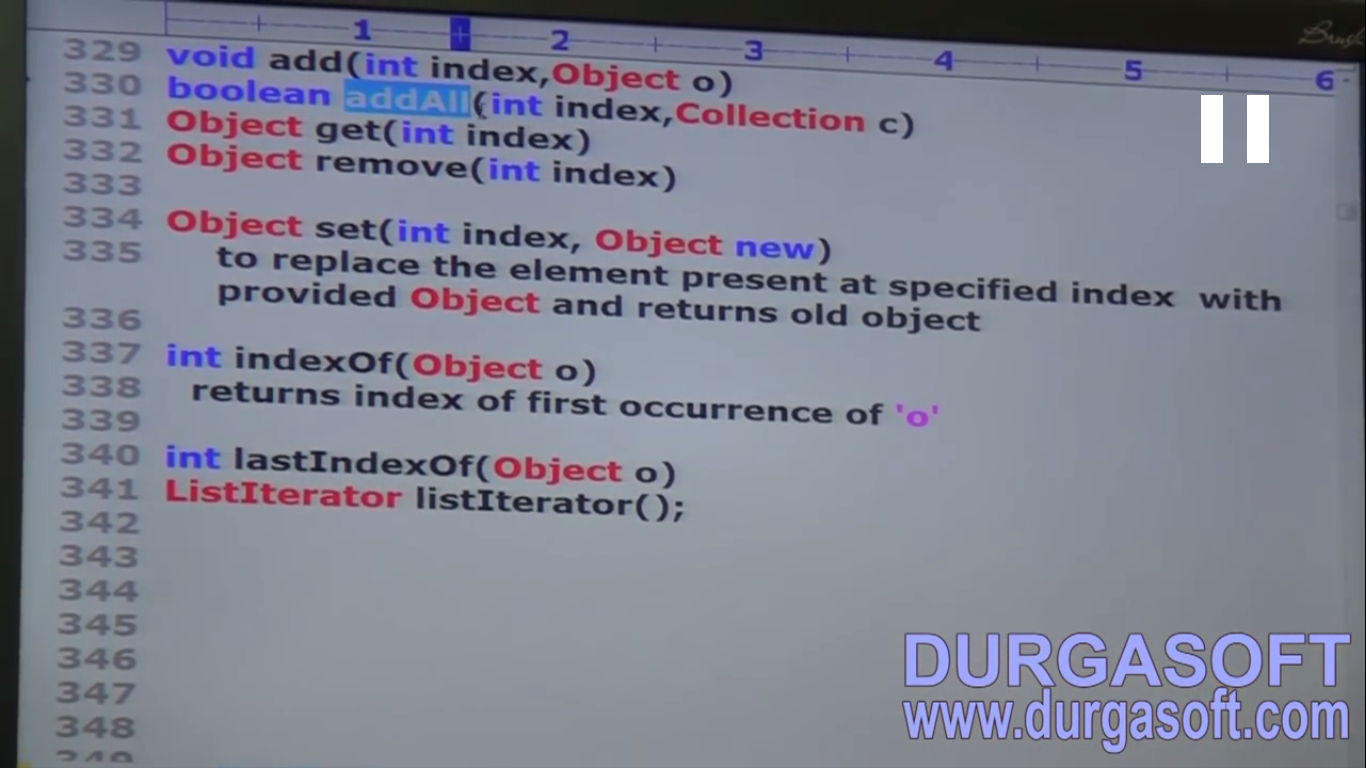
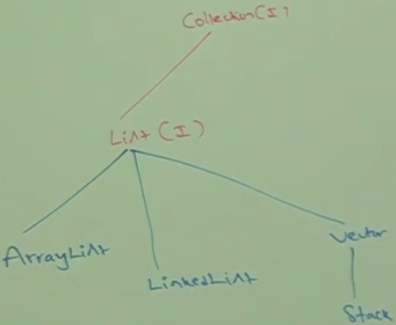


1. **NOTE**: There is **no concrete** class which implements **Collection(I) interface** directly.

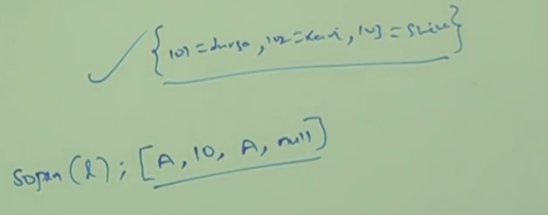
List(I) Interface

1. **Application**:
   1. When we represent a group of individual objects as a single entity where
      1. duplicates are allowed
      2. insertion order is preserved.
2. Child interface of Collection(I)
3. **Some Points**:  
   
   1. **Insertion Order**: Insertion order is preserved by index. First element is stored at index 0, next inserted element is stored at index 1 and so on.
   2. **Duplicate Differentiation**: We can differentiate duplicate objects using index. There are two A objects in above list. They would be differentiated by index.

**NOTE**: Hence index plays very important role in List(I)

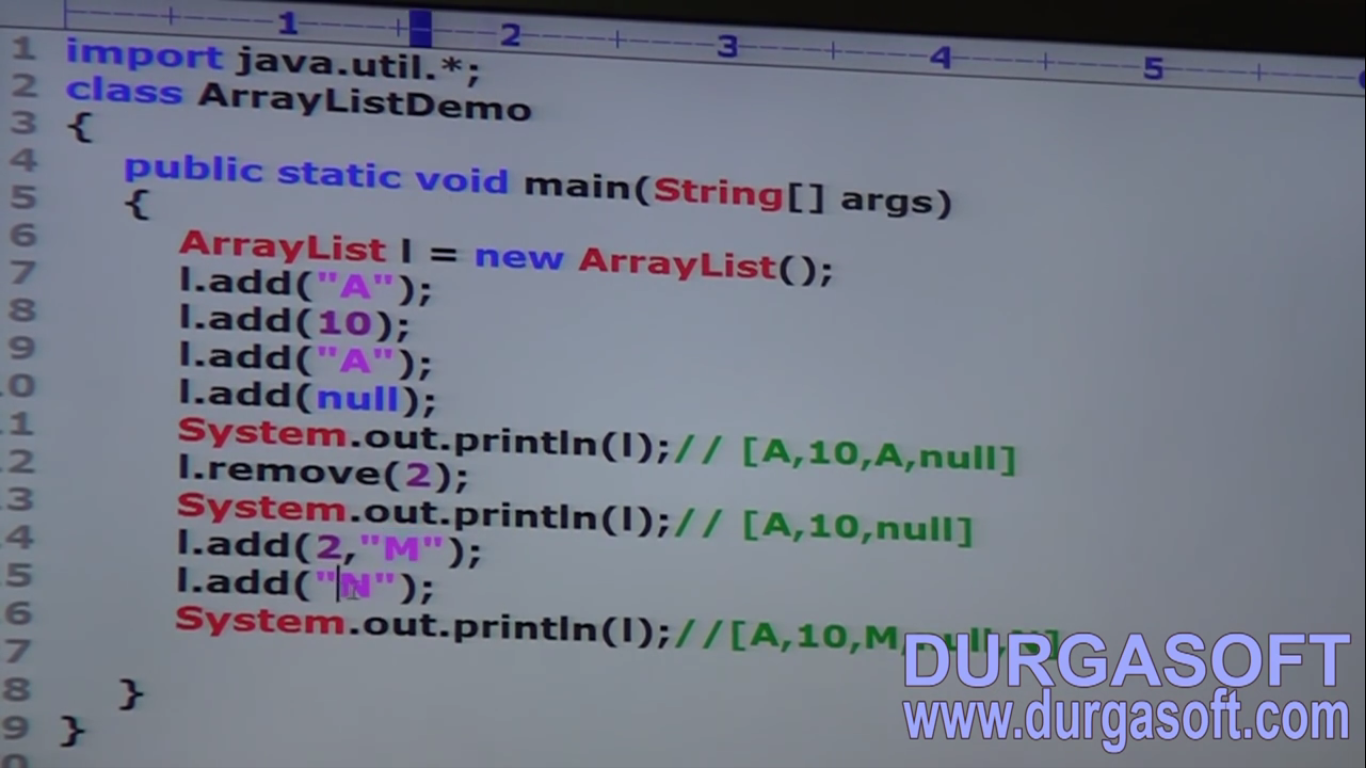
1. Now, next important terminology.
   1. All Collection methods are available to list but List(I) interface defines some list specific methods which talk in index form.
2. List(I) interface defines the following specific methods.   
     
   **Implemented classes for List(I) Interface**  
   
3. **d**

ArrayList(C)

1. **Properties of ArrayList(C)**:
   1. **Underlying Data Structure**: Resizable/ Growable Array.
   2. **Insertion Order Preserved:** Yes.
   3. **Duplicates Allowed**: Yes
   4. **Heterogenous Objects**: Allowed
   5. **Null Insertion**: Allowed
   6. **Implements**: Serializable, Cloneable, RandomAccess
   7. **Thread-Safe**: No
2. **Constructors**:
   1. **ArrayList()**:
      1. Creates an empty ArrayList object with default initial capacity🡪 10
      2. **Default Capacity**: 10
      3. When ArrayList reaches its max capacity then a new ArrayList object is created with   
         **New Capacity** = (int) (Current Capacity \* 3/2 ) + 1
   2. **ArrayList(int initialCapacity)**:
      1. Creates an empty ArrayList object with **specified intial capacity.**
   3. **ArrayList(Collection c)**
      1. Creates an **equivalent** ArrayList object for the given collection.
      2. **NOTE**: This type of constructor is almost all Collection Classes.
   4. 

Map’s toString() content printing style.   
Using curly braces and key=value

Collection’s toString() content printing style.   
square bracket and elements

1. **Example:**  
   We are getting this warning as from version 1.5 onwards, collection generic concept came for type safely.

auto-boxing

1. ArrayList implements **RandomAccess** interface.