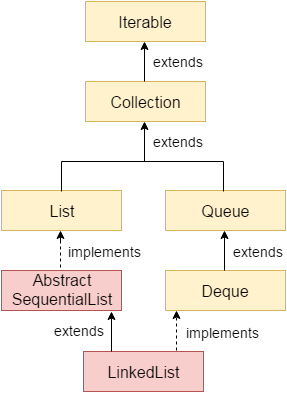
Java\_LinkedList\_05\_March

# Java LinkedList class

* Java LinkedList class uses a doubly linked list to store the elements. It provides a linked-list data structure. It inherits the AbstractList class and implements List and Deque interfaces.
* Java LinkedList class can contain duplicate elements.
* Java LinkedList class maintains insertion order.
* Java LinkedList class is non synchronized.
* In Java LinkedList class, manipulation is fast because no shifting needs to occur.
* Java LinkedList class can be used as a list, stack or queue.

### Hierarchy of LinkedList class



### Doubly Linked List

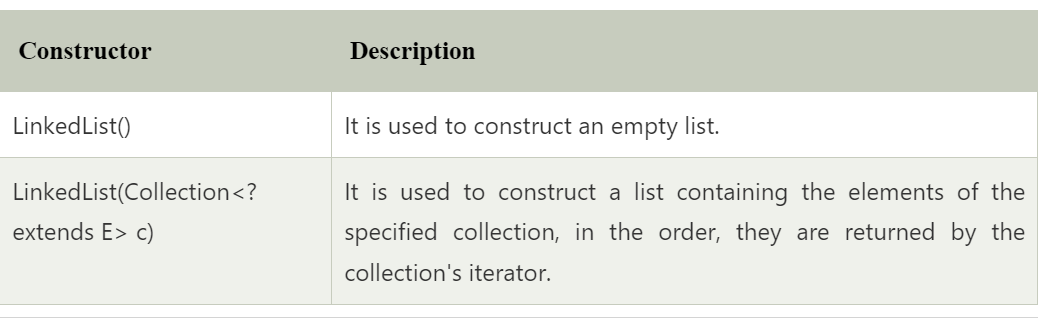
In the case of a doubly linked list, we can add or remove elements from both sides.



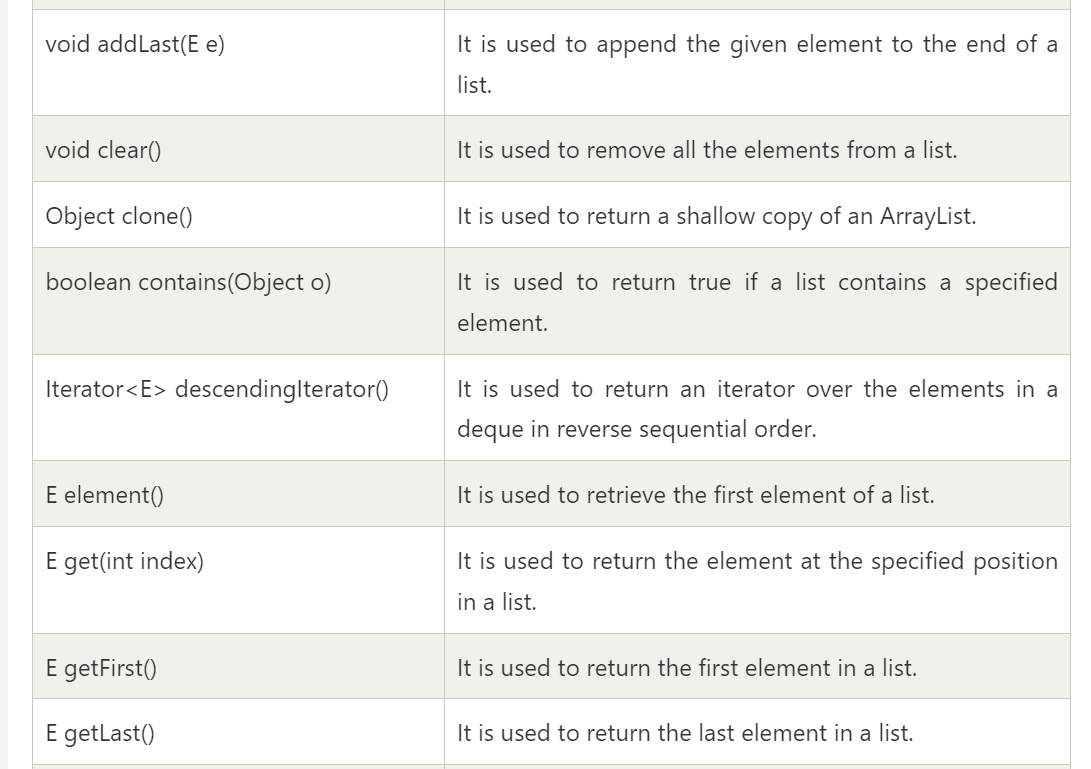
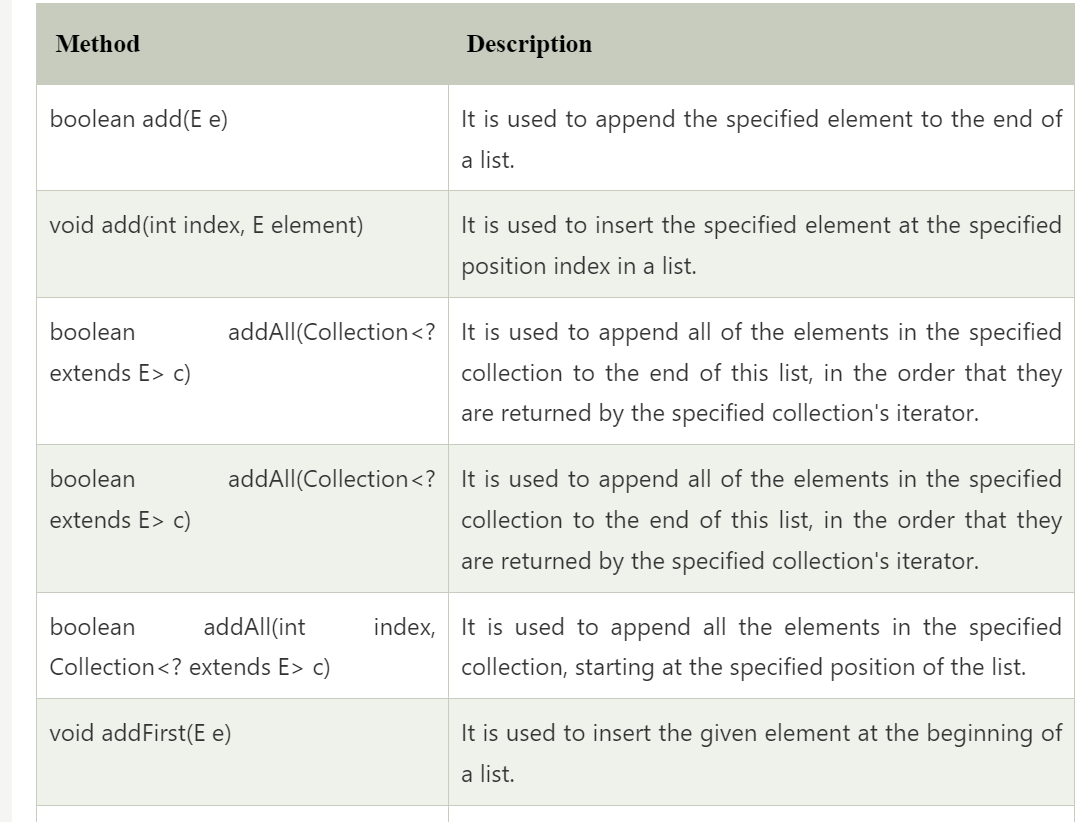
### LinkedList class declaration

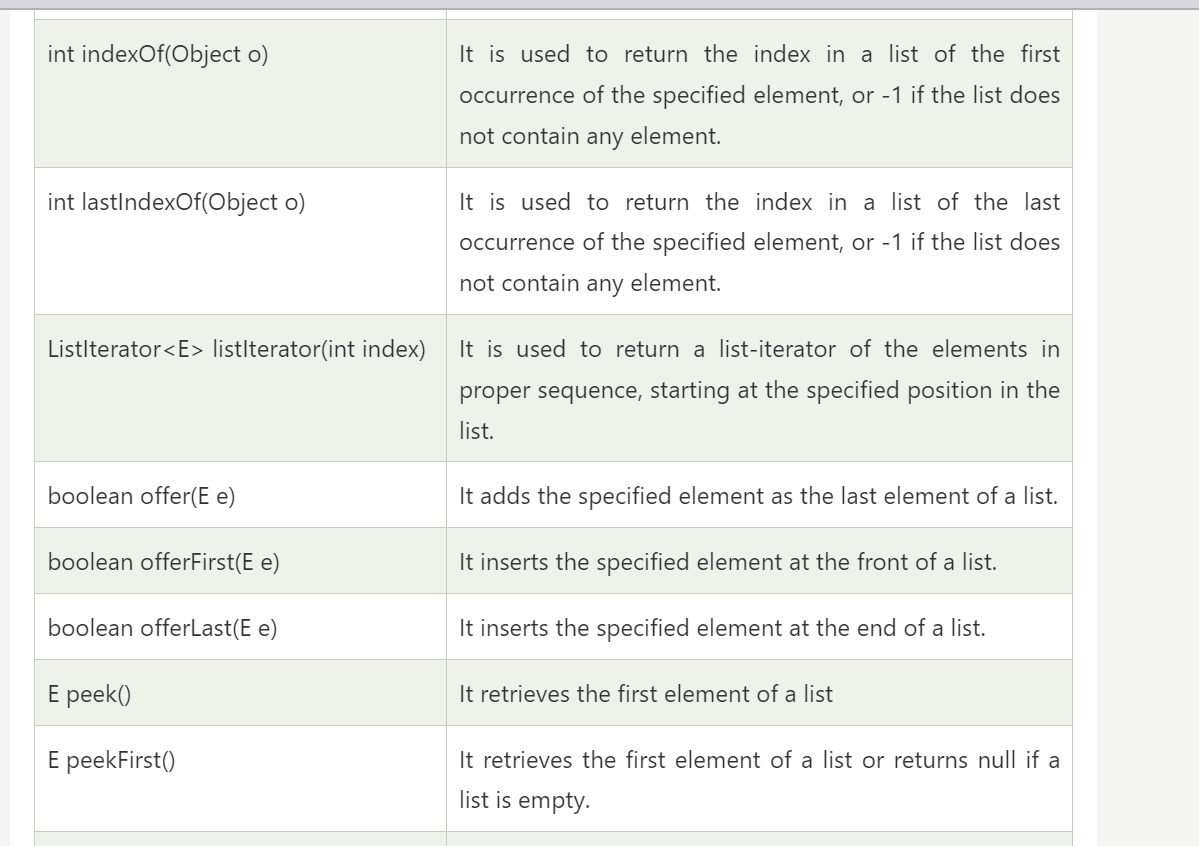
1. **public** **class** LinkedList<E> **extends** AbstractSequentialList<E> **implements** List<E>, Deque<E>, Cloneable, Serializable

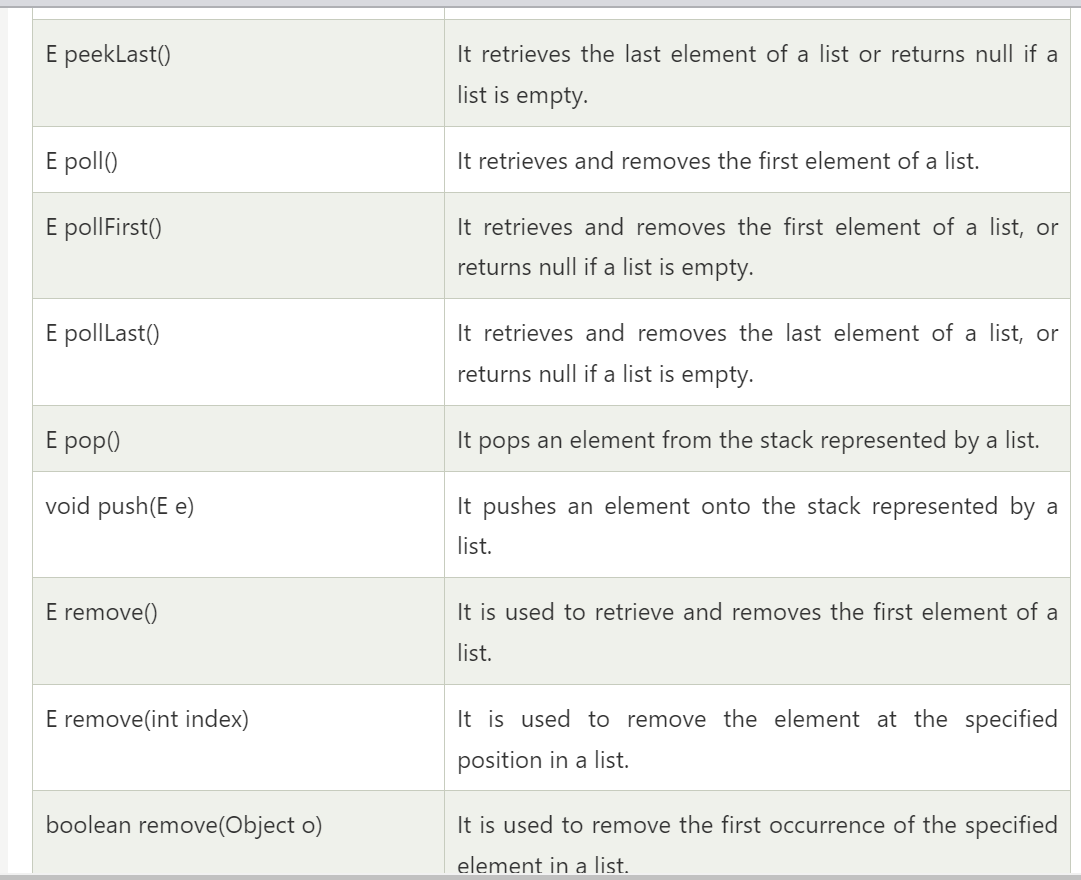
### Constructors of Java LinkedList

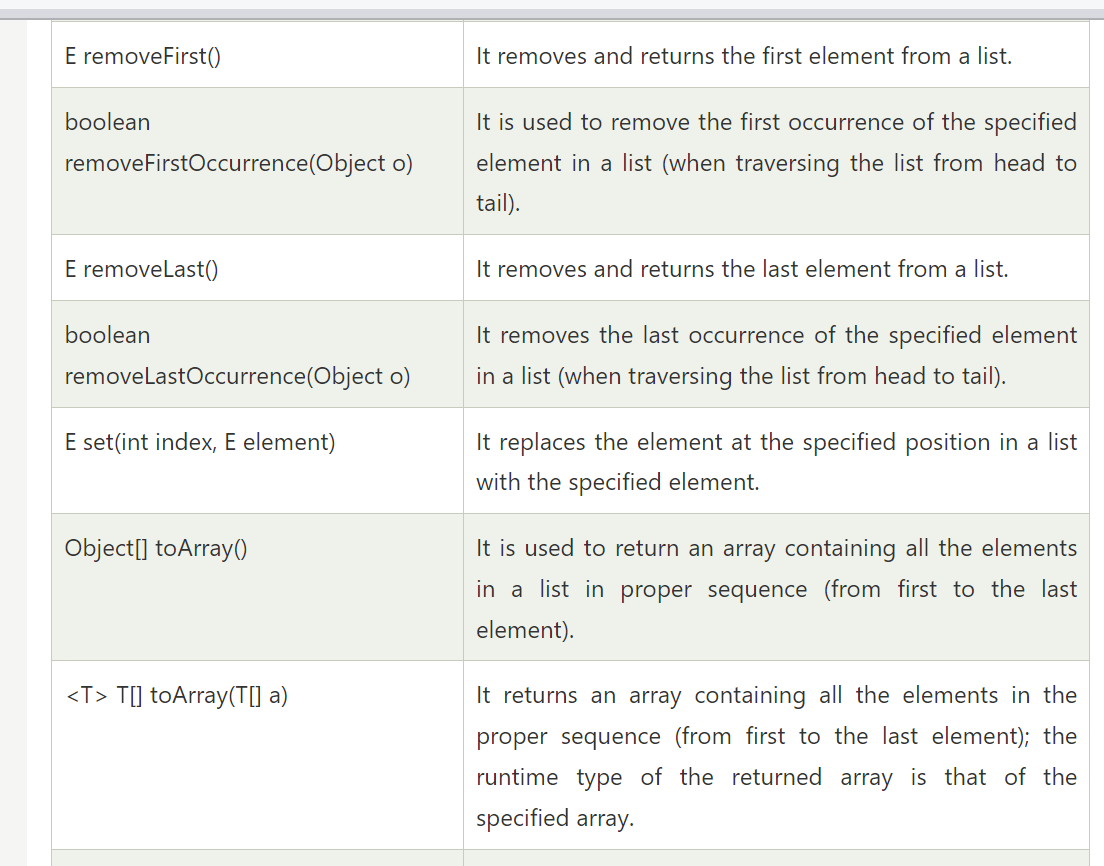


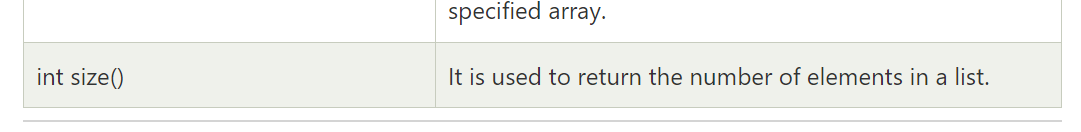
### Methods of Java LinkedList



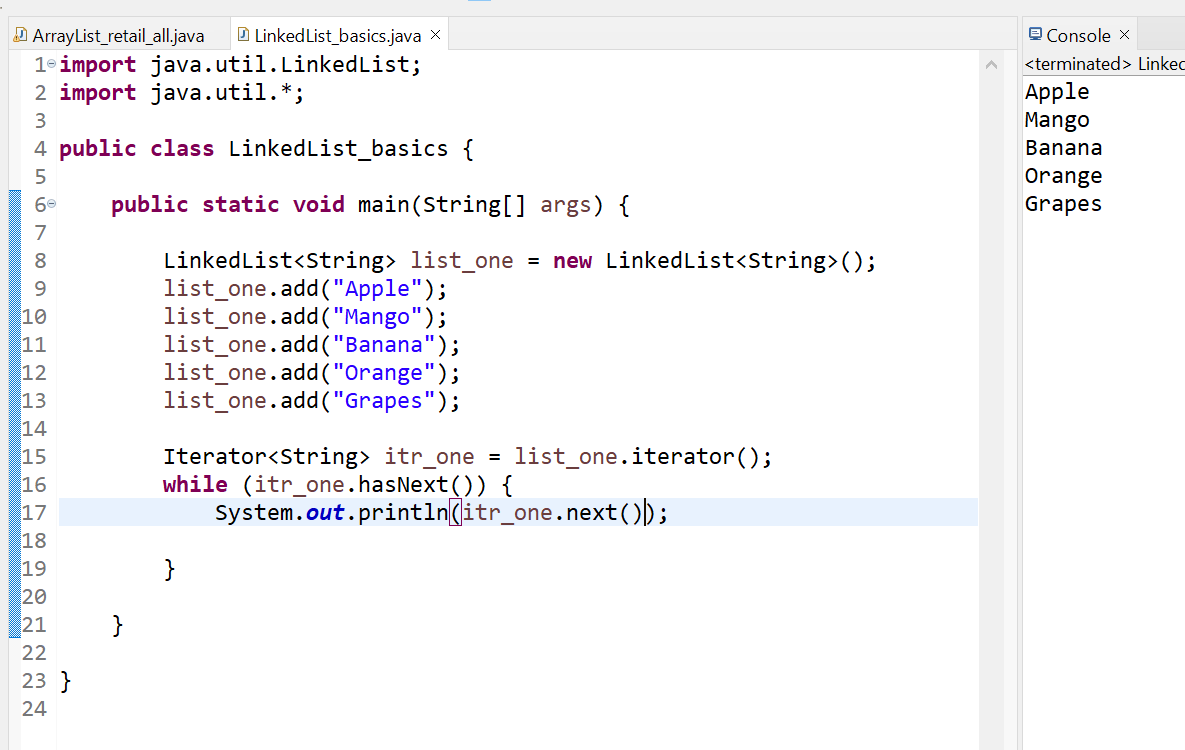




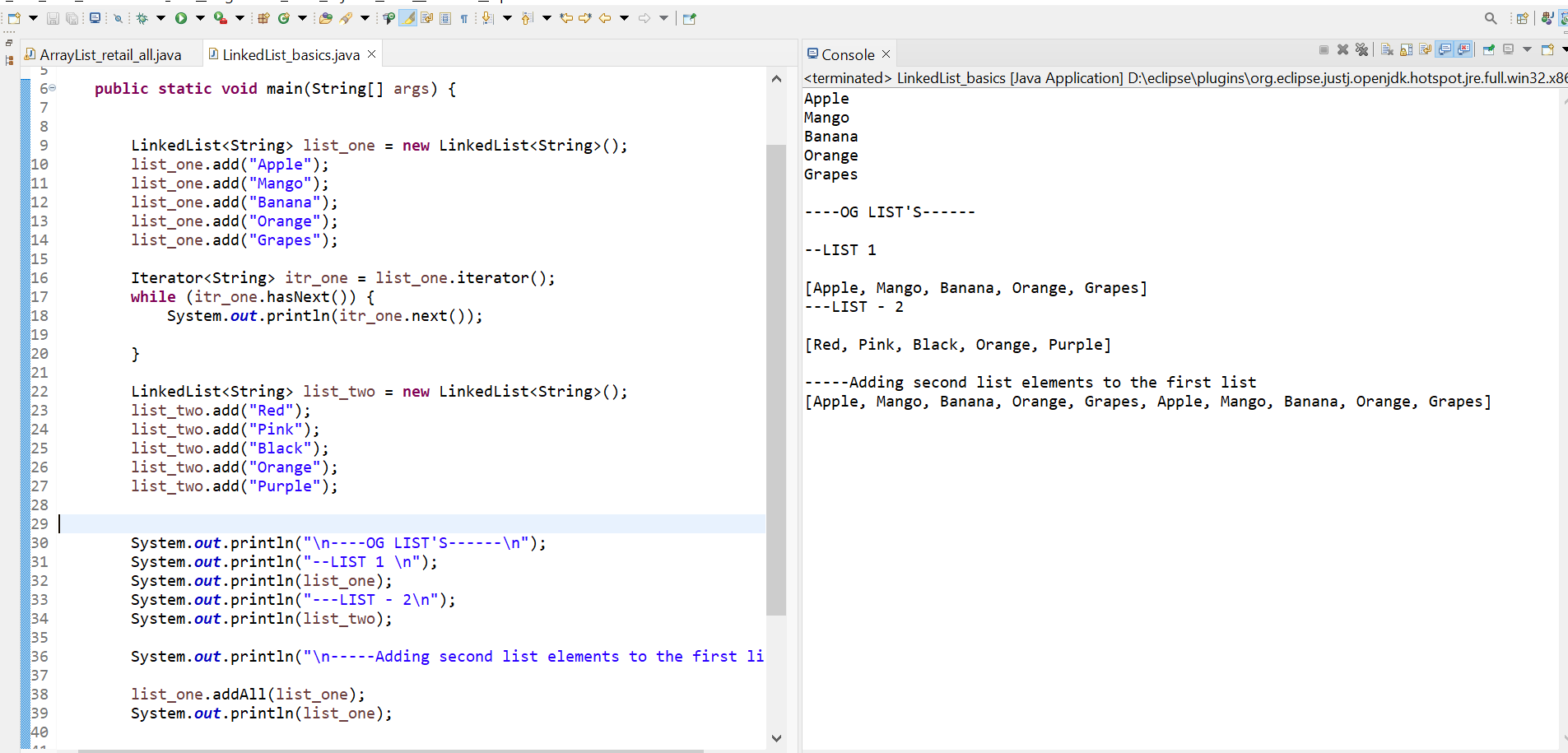




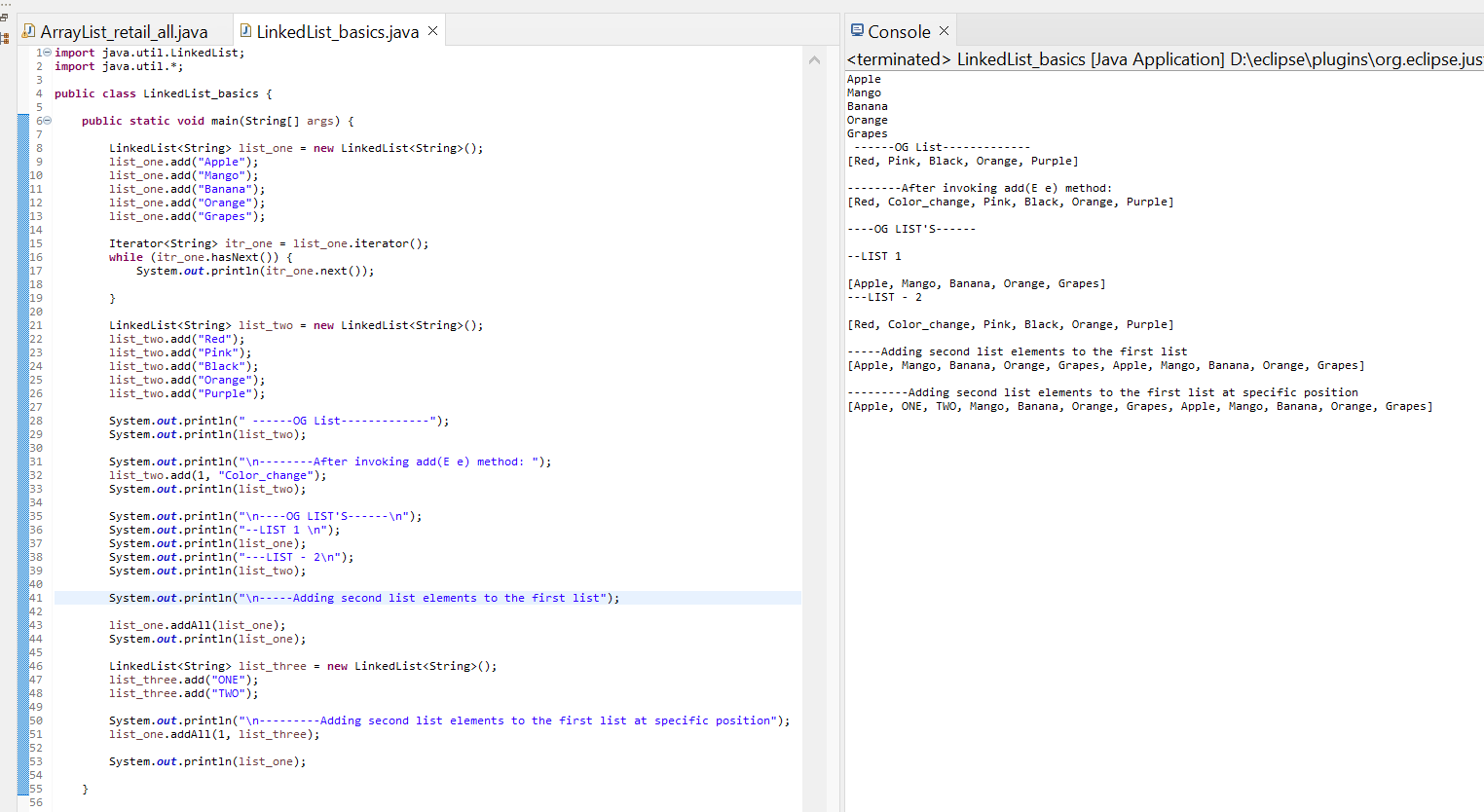
### Java LinkedList Example



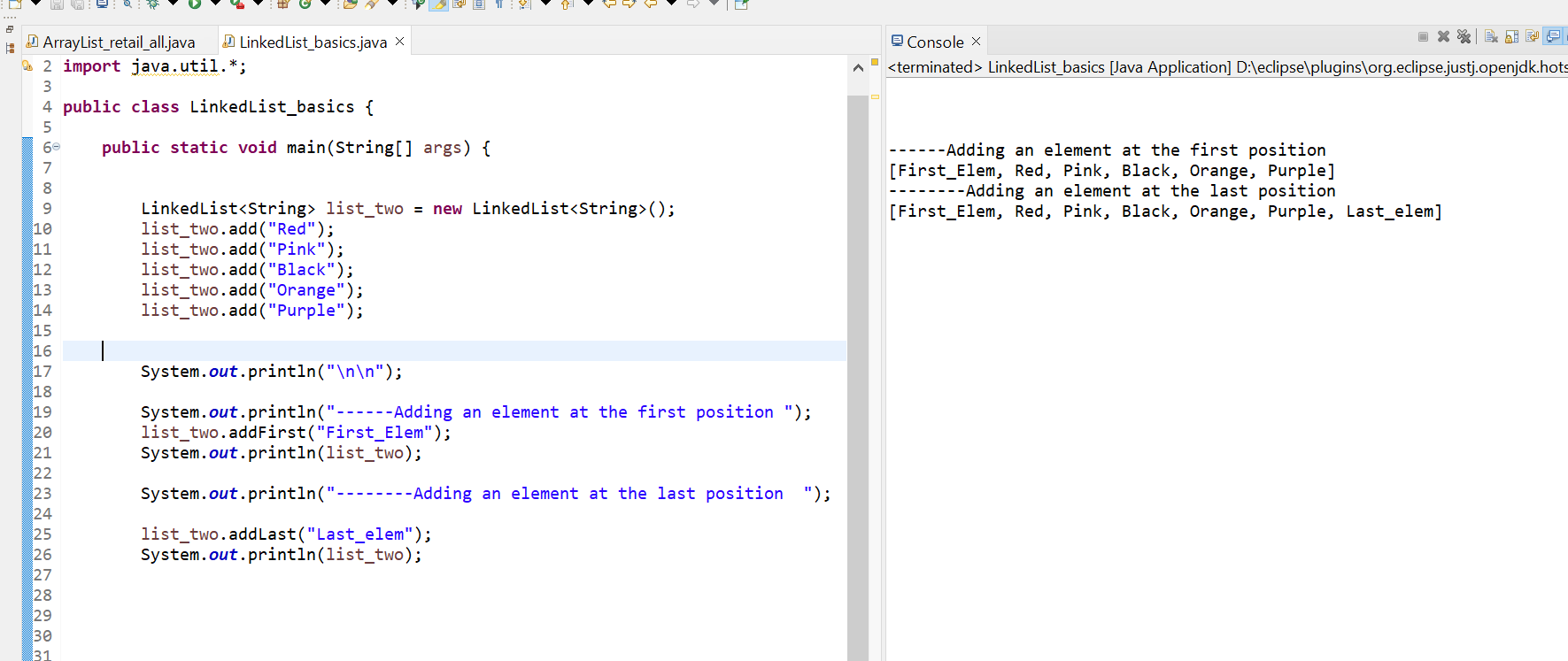
### Java LinkedList example to add elements



**Adding second list elements to the first list at specific position**

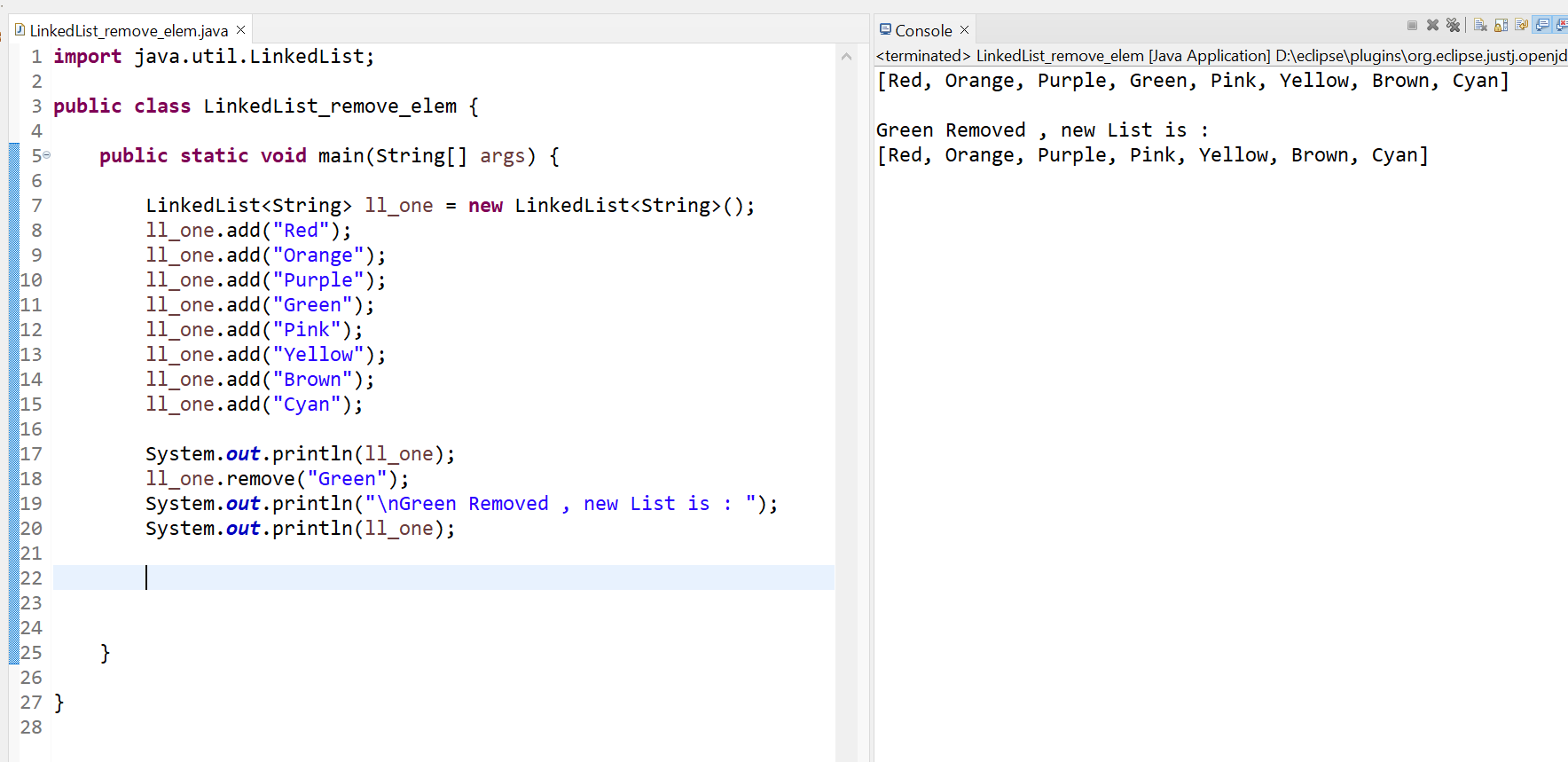
****

**Adding an element at the First and last position**

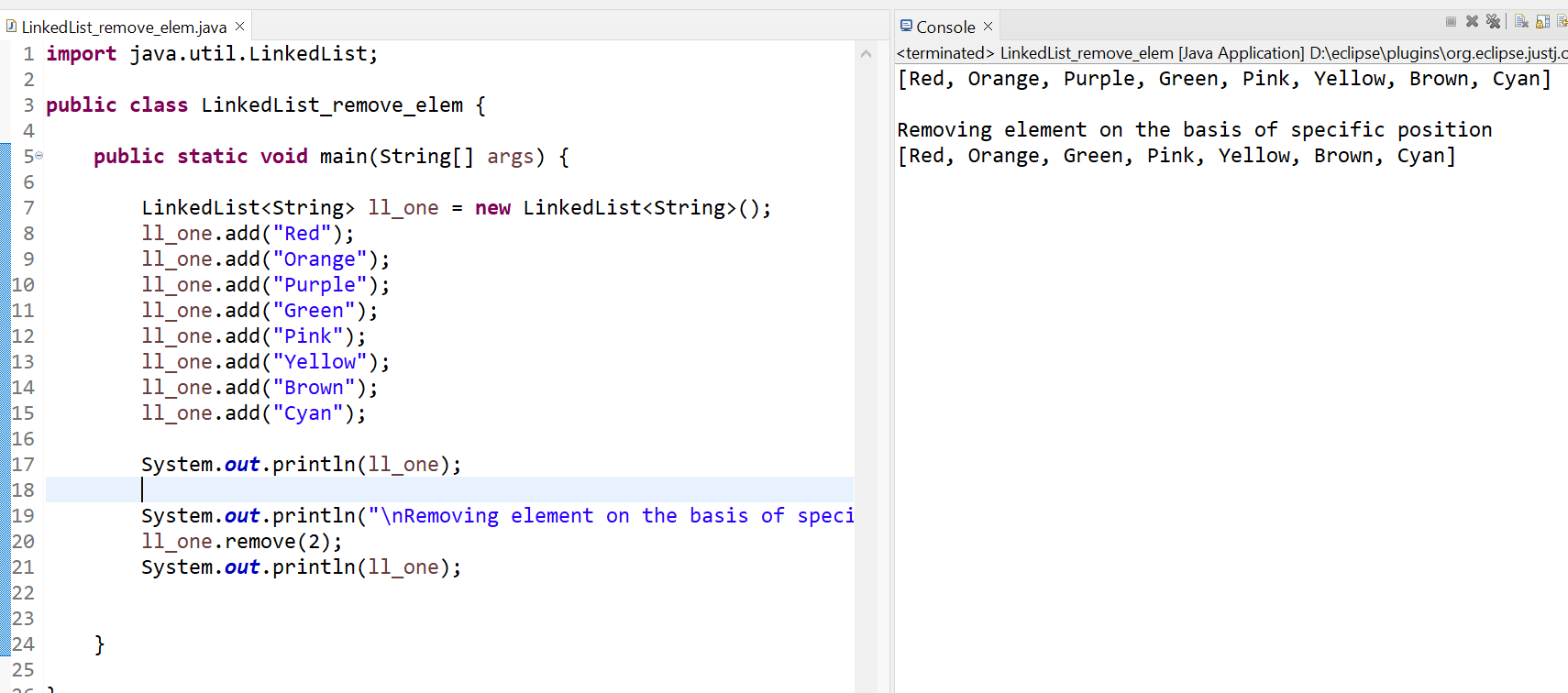
****

### Java LinkedList example to remove elements

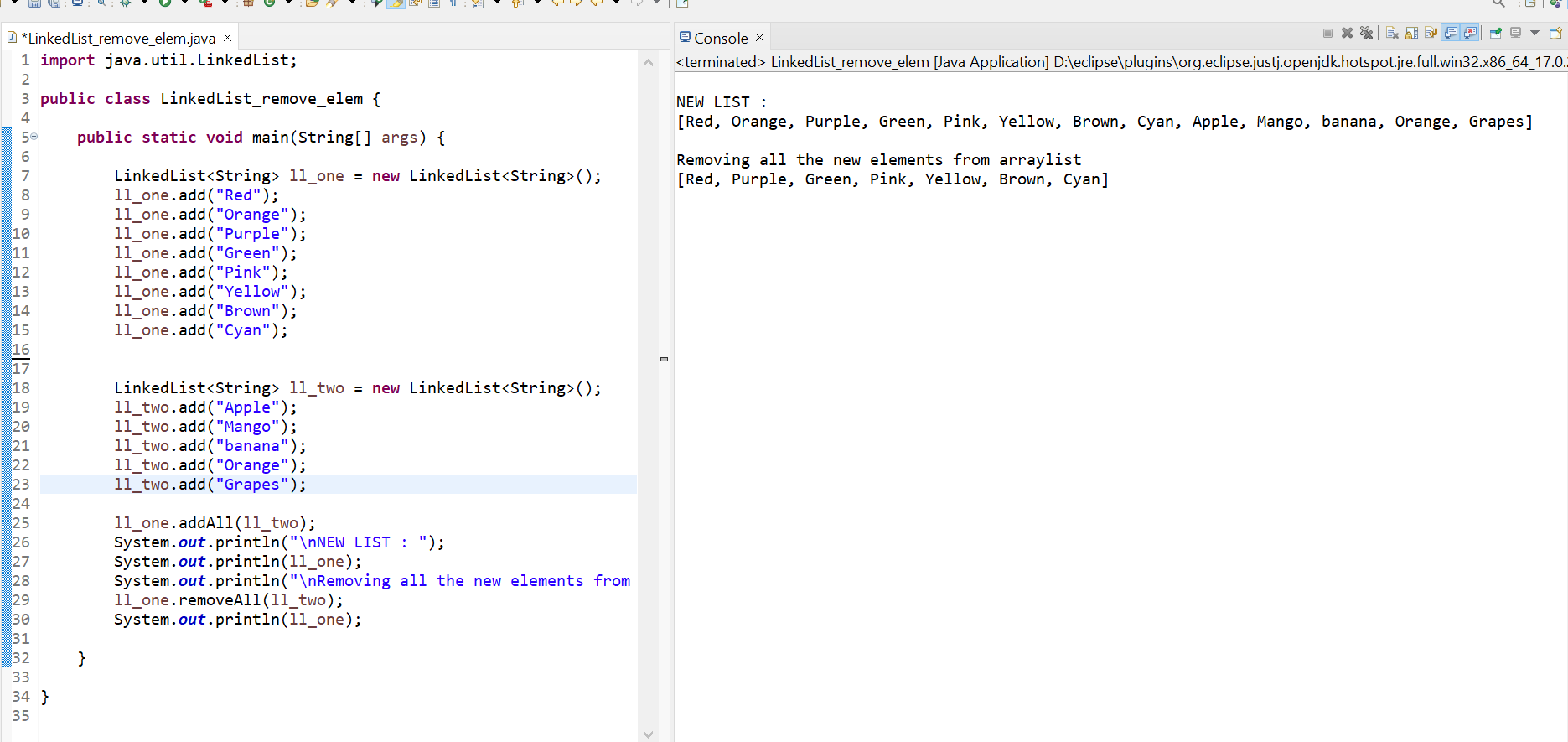
**Remove element by the value :**

****

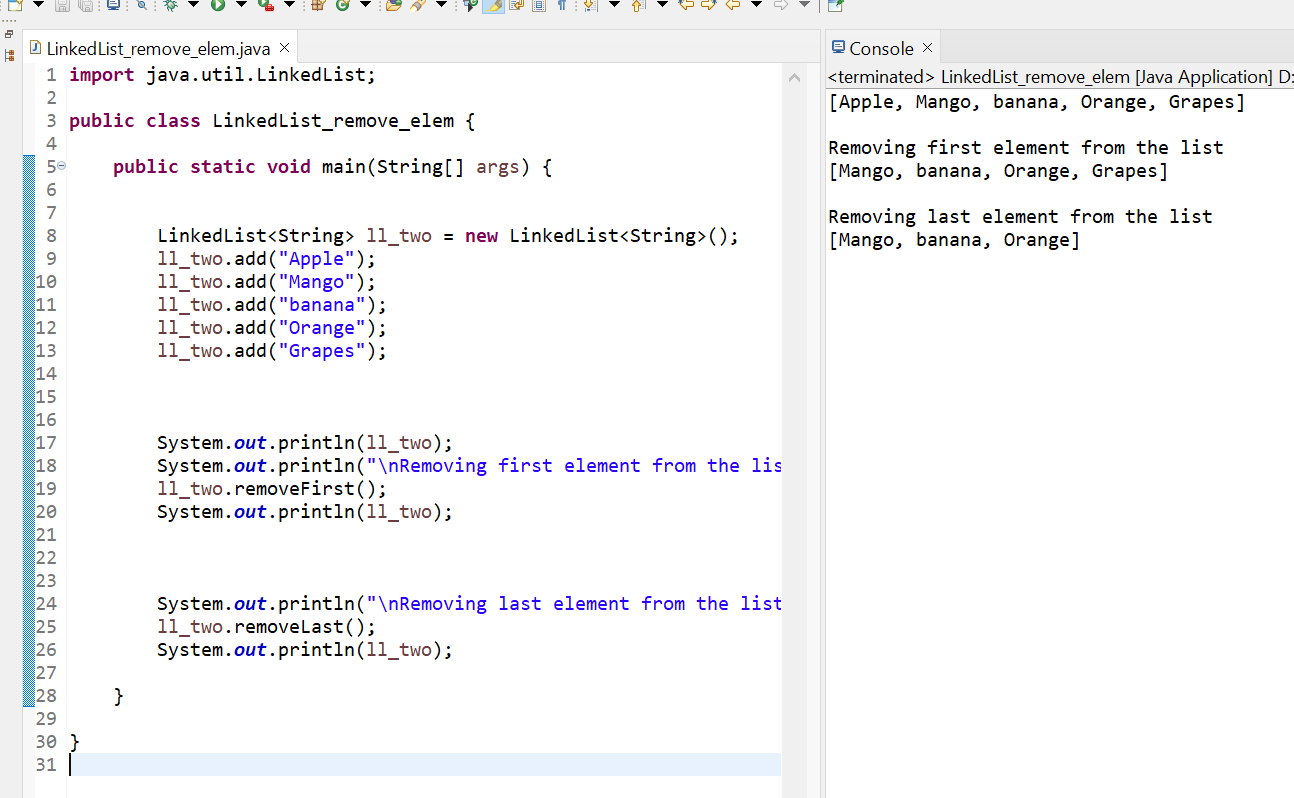
**Remove element from the specific position :**

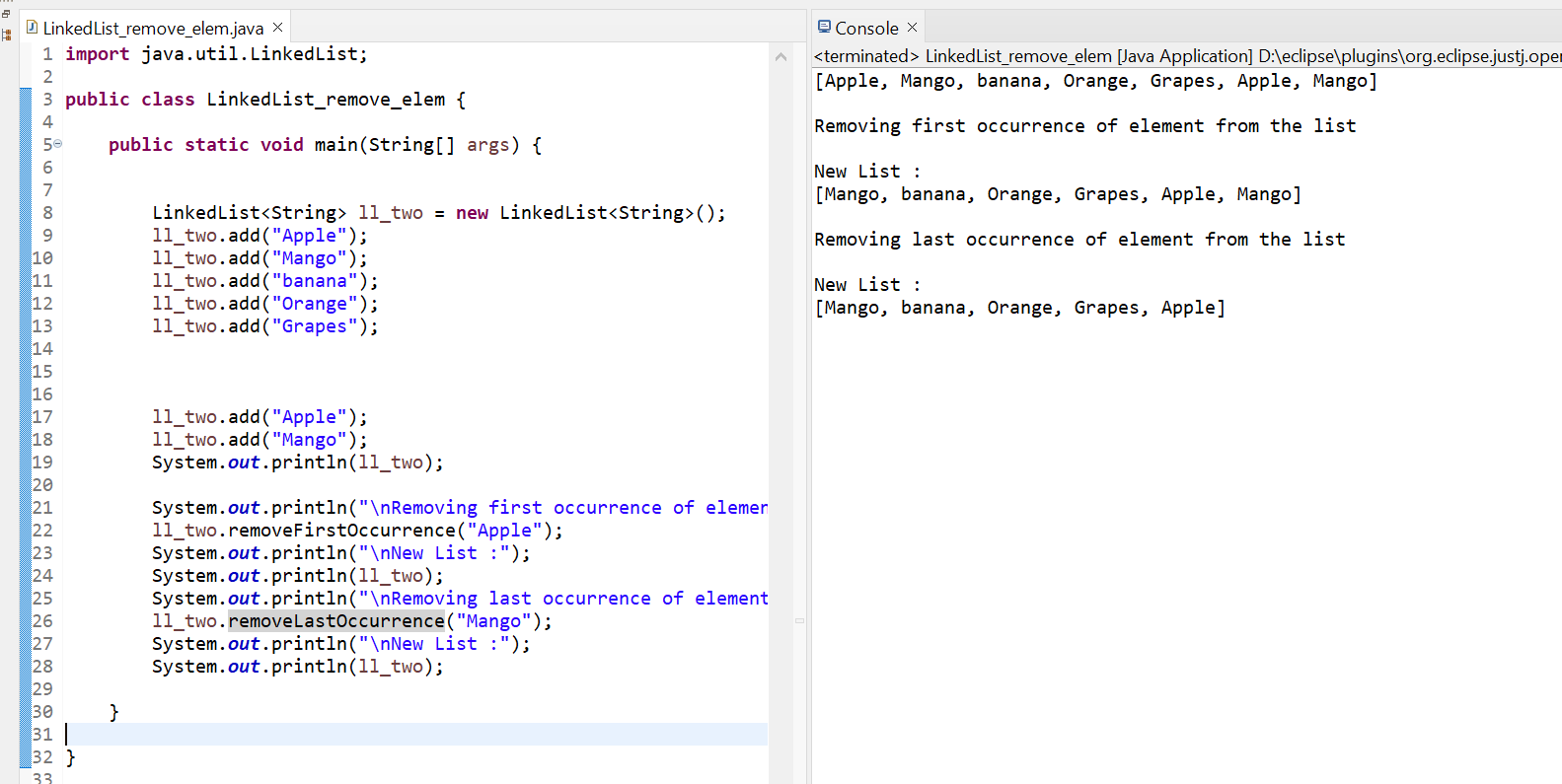
****

**Remove all the new elements :**

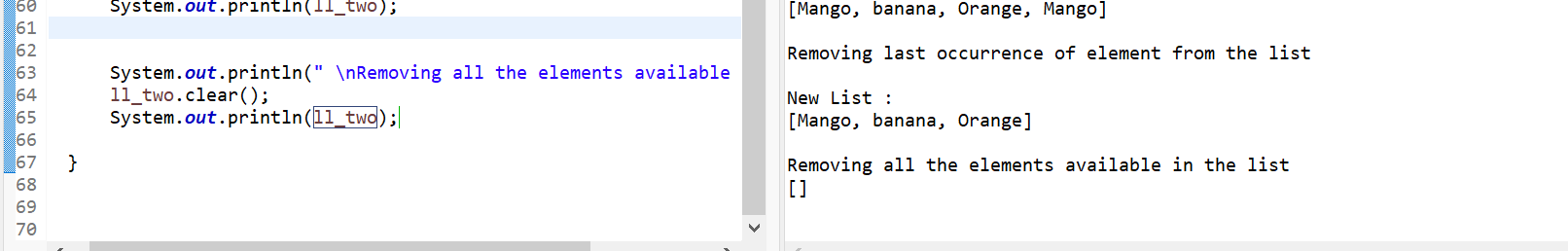
****

**Removing first & Last element from the list**

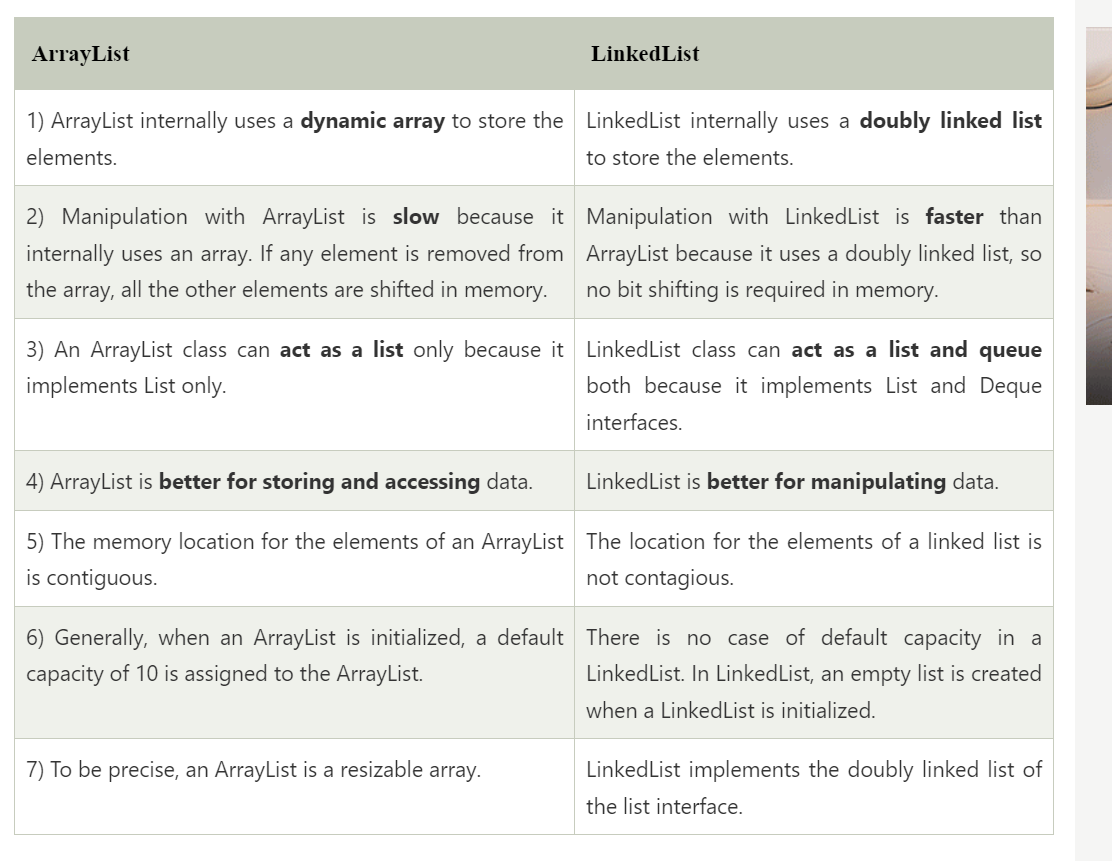
**Removing first and last occurrence of element from the list**

****

**Removing all the elements available in the list**

****

# Difference Between ArrayList and LinkedList

****

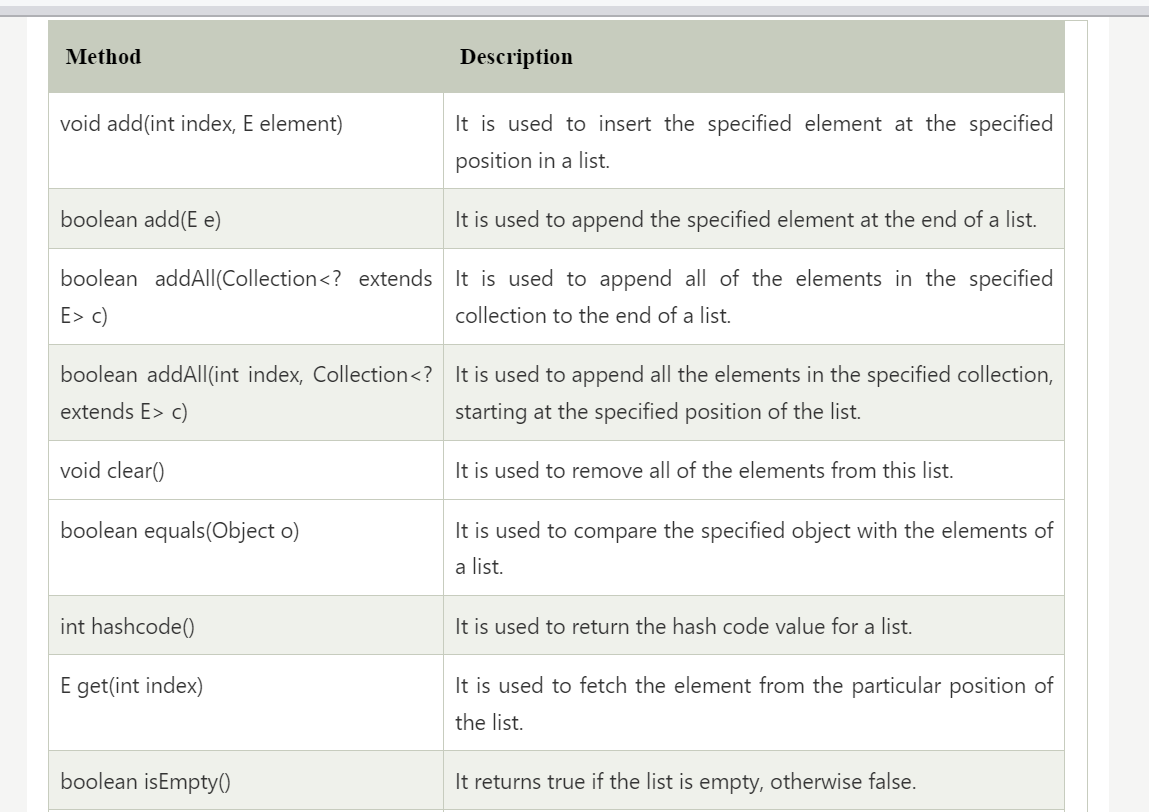
# Java List

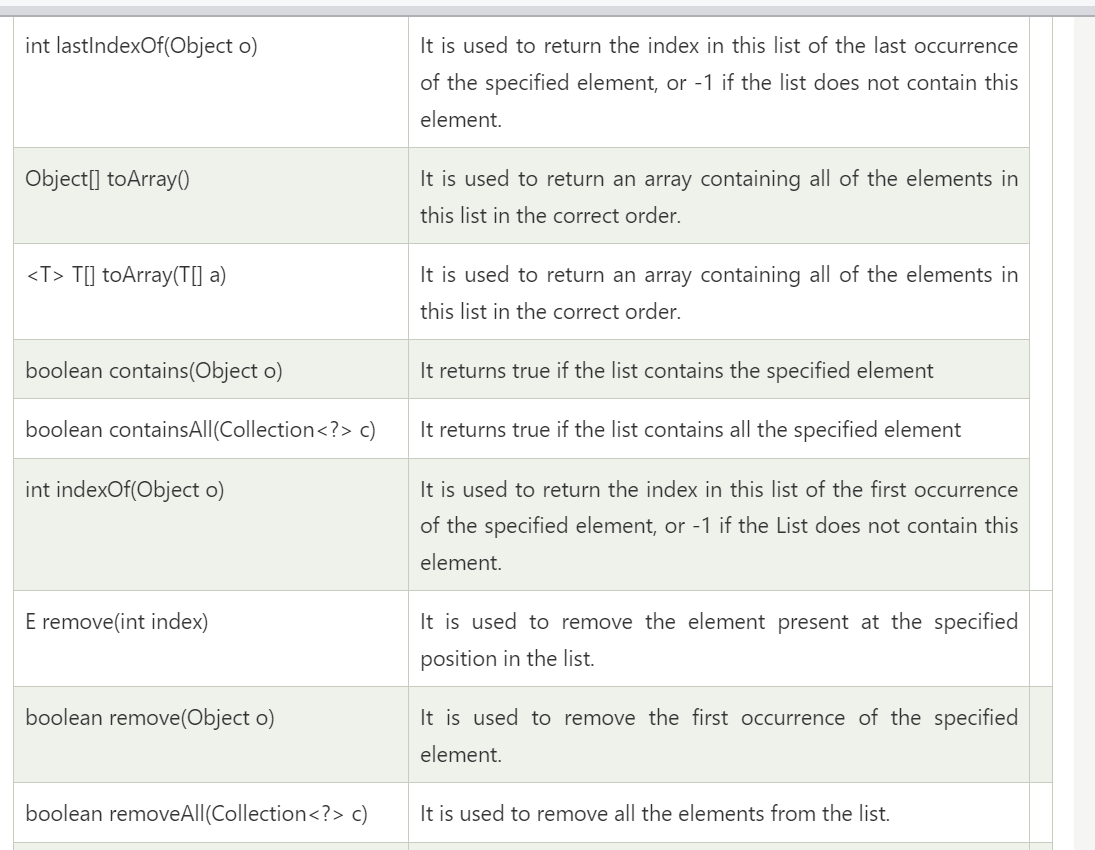
* List in Java provides the facility to maintain the ordered collection.
* It contains the index-based methods to insert, update, delete and search the elements. It can have the duplicate elements also.
* We can also store the null elements in the list.
* The List interface is found in the java.util package and inherits the Collection interface. It is a factory of ListIterator interface.
* Through the ListIterator, we can iterate the list in forward and backward directions. The implementation classes of List interface are ArrayList, LinkedList, Stack and Vector. The ArrayList and LinkedList are widely used in Java programming

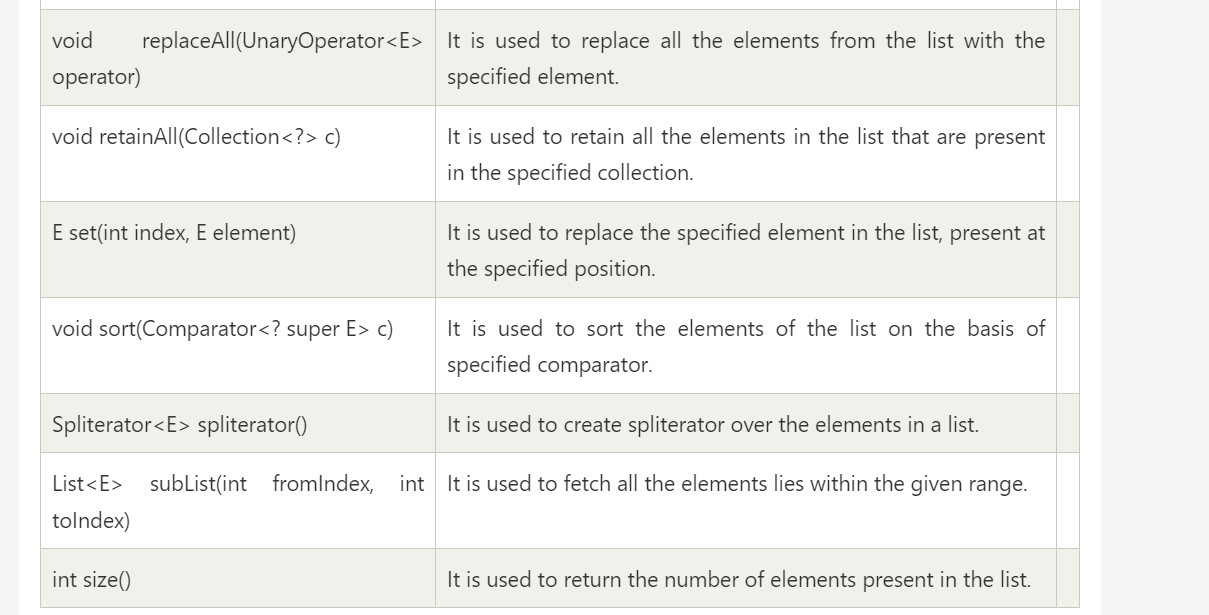
### List Interface declaration

1. **public** **interface** List<E> **extends** Collection<E>

### Java List Methods







### Java List vs ArrayList

List is an interface whereas ArrayList is the implementation class of List.

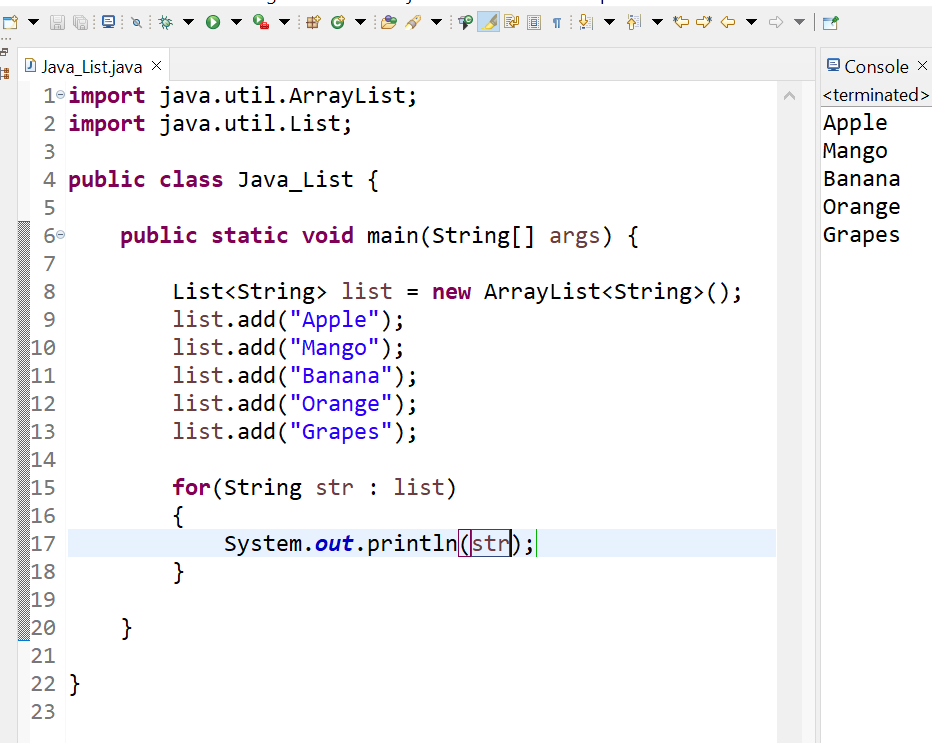
### How to create List

The ArrayList and LinkedList classes provide the implementation of List interface. Let's see the examples to create the List:

1. //Creating a List of type String using ArrayList
2. List<String> list=**new** ArrayList<String>();

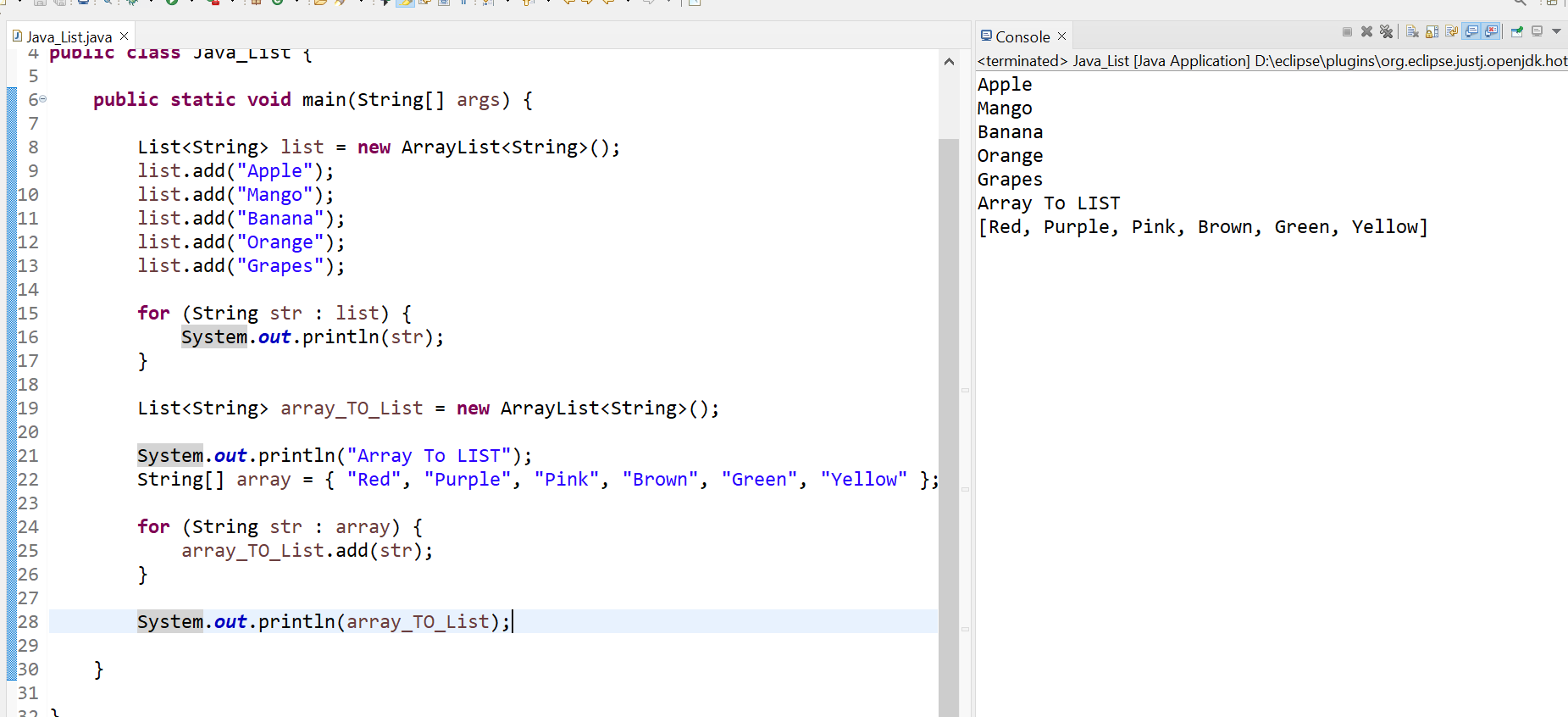
1. //Creating a List of type Integer using ArrayList
2. List<Integer> list=**new** ArrayList<Integer>();
3. //Creating a List of type Book using ArrayList
4. List<Book> list=**new** ArrayList<Book>();
5. //Creating a List of type String using LinkedList
6. List<String> list=**new** LinkedList<String>();

### Java List Example



### How to convert Array to List

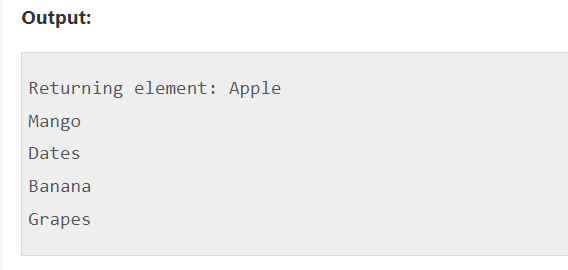
We can convert the Array to List by traversing the array and adding the element in list one by one using list.add() method.



### Get and Set Element in List

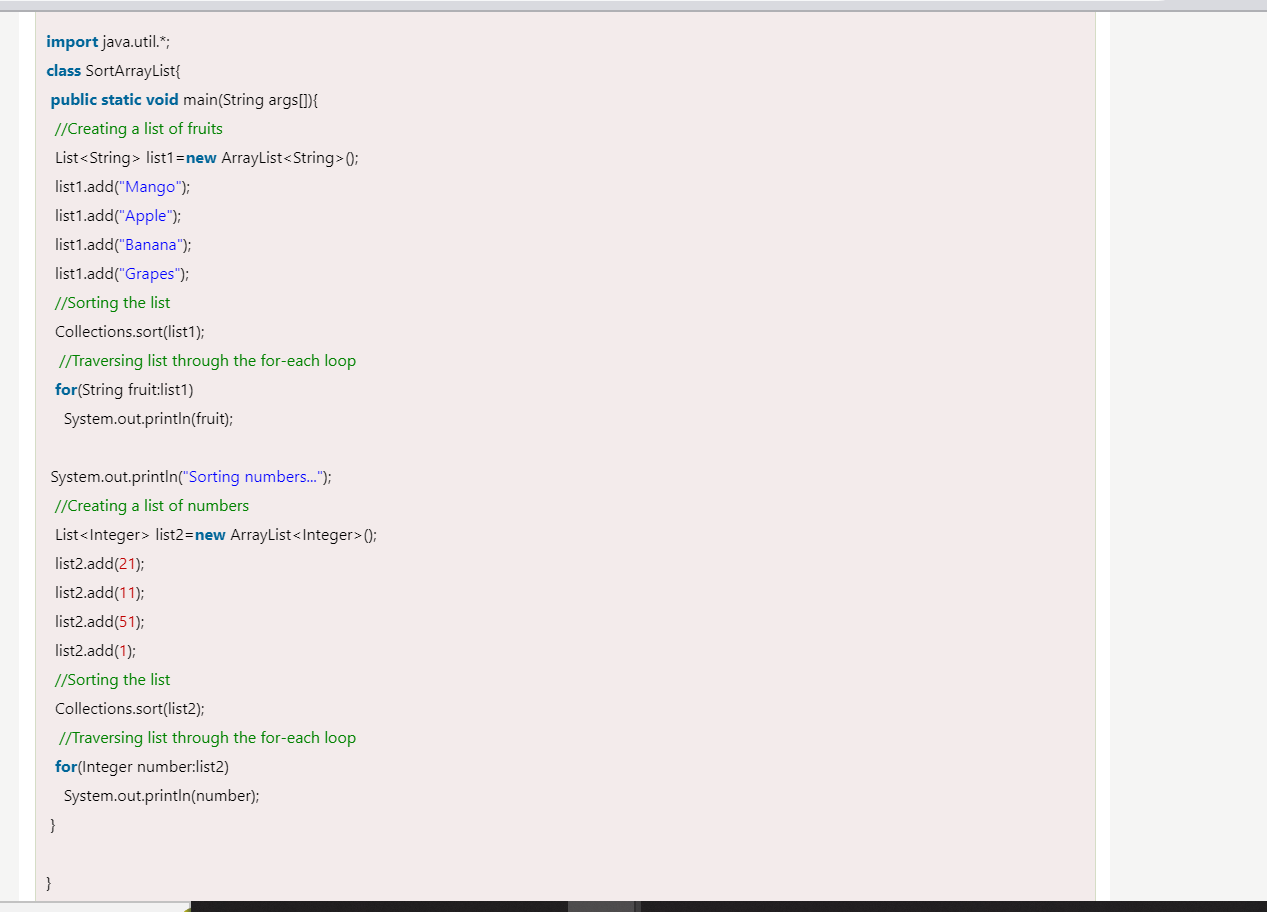
The *get() method* returns the element at the given index, whereas the *set() method* changes or replaces the element.

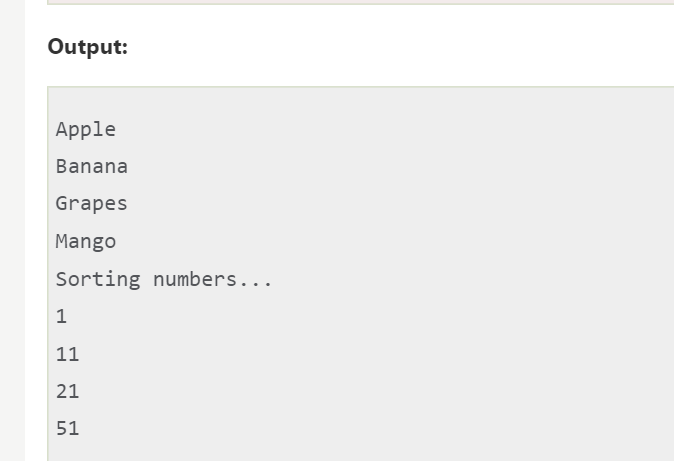




### How to Sort List

There are various ways to sort the List, here we are going to use Collections.sort() method to sort the list element. The *java.util* package provides a utility class **Collections** which has the static method sort(). Using the **Collections.sort()** method, we can easily sort any List.





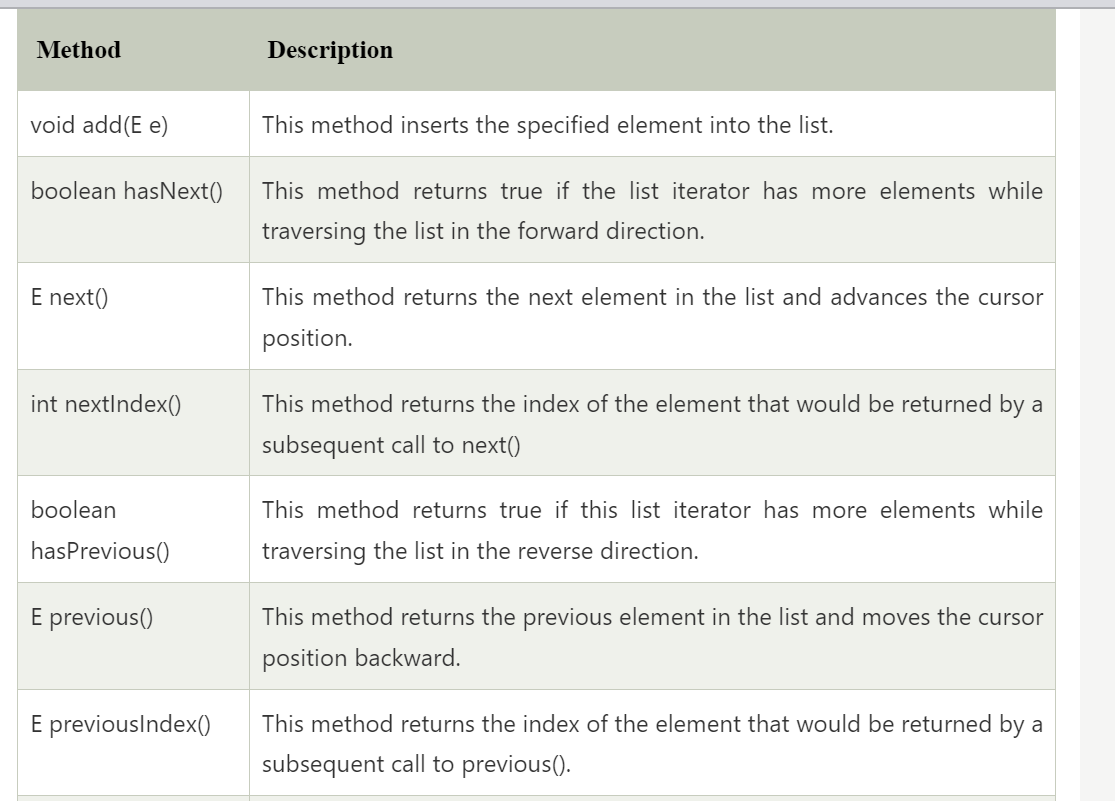
## Java ListIterator Interface

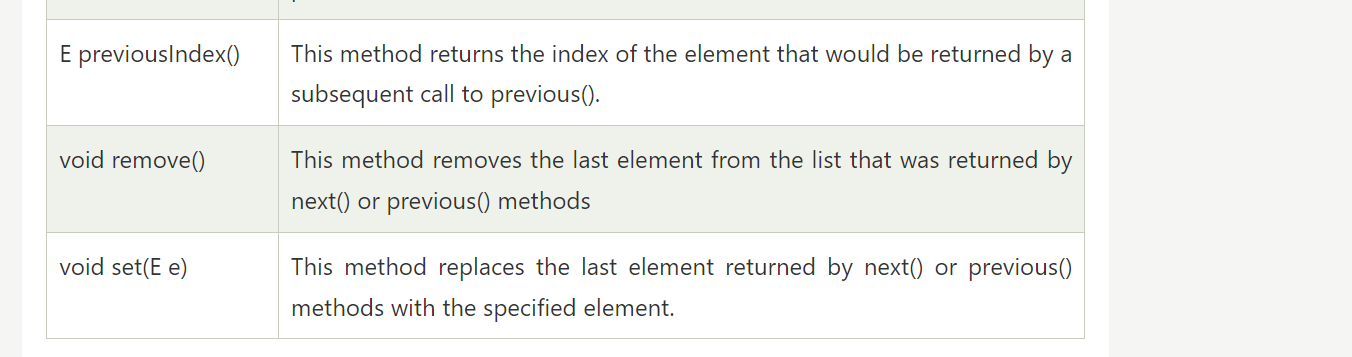
ListIterator Interface is used to traverse the element in a backward and forward direction.

### ListIterator Interface declaration

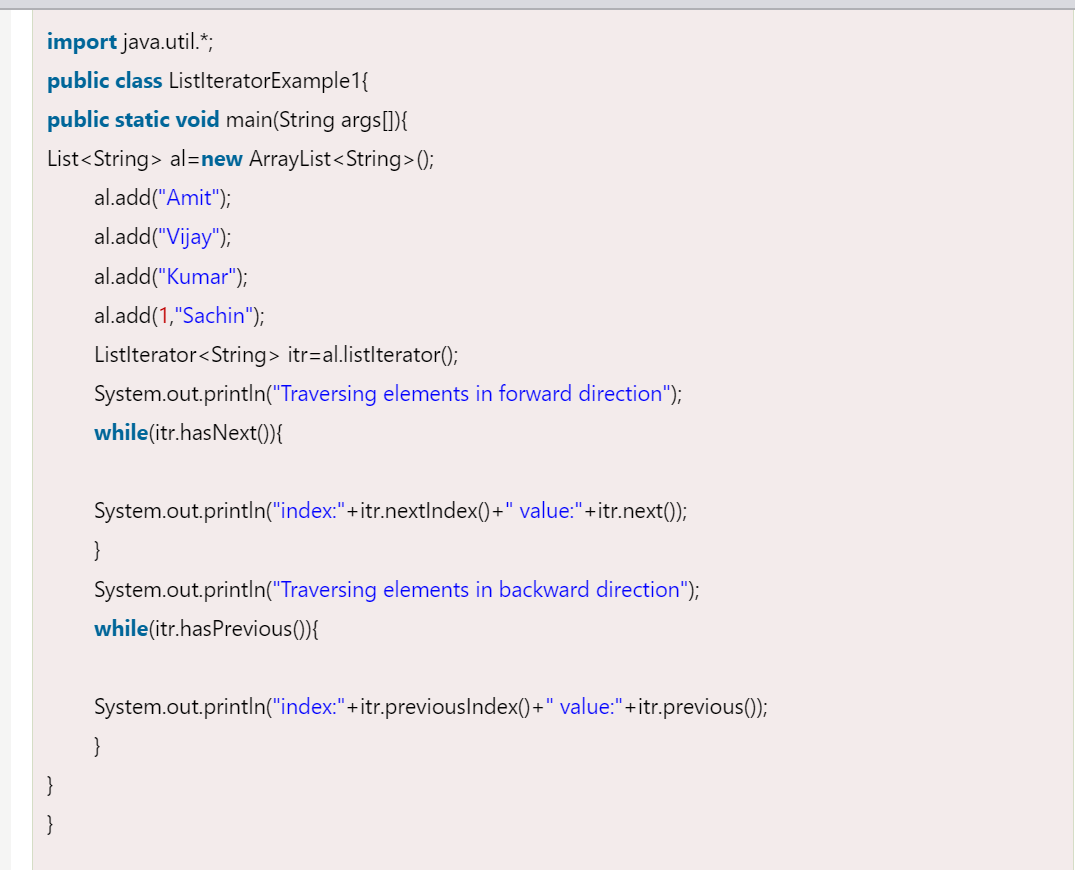
1. **public** **interface** ListIterator<E> **extends** Iterator<E>

### Methods of Java ListIterator Interface:





### Example of ListIterator Interface



OUTPUT :

