Java - The List Interface

Advertisements

[Previous Page](https://www.tutorialspoint.com/java/java_collections.htm)

[Next Page](https://www.tutorialspoint.com/java/java_collections.htm)

The List interface extends **Collection** and declares the behavior of a collection that stores a sequence of elements.

* Elements can be inserted or accessed by their position in the list, using a zero-based index.
* A list may contain duplicate elements.
* In addition to the methods defined by **Collection**, List defines some of its own, which are summarized in the following table.
* Several of the list methods will throw an UnsupportedOperationException if the collection cannot be modified, and a ClassCastException is generated when one object is incompatible with another.

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **void add(int index, Object obj)**  Inserts obj into the invoking list at the index passed in the index. Any pre-existing elements at or beyond the point of insertion are shifted up. Thus, no elements are overwritten. |
| 2 | **boolean addAll(int index, Collection c)**  Inserts all elements of **c** into the invoking list at the index passed in the index. Any pre-existing elements at or beyond the point of insertion are shifted up. Thus, no elements are overwritten. Returns true if the invoking list changes and returns false otherwise. |
| 3 | **Object get(int index)**  Returns the object stored at the specified index within the invoking collection. |
| 4 | **int indexOf(Object obj)**  Returns the index of the first instance of obj in the invoking list. If obj is not an element of the list, .1 is returned. |
| 5 | **int lastIndexOf(Object obj)**  Returns the index of the last instance of obj in the invoking list. If obj is not an element of the list, .1 is returned. |
| 6 | **ListIterator listIterator( )**  Returns an iterator to the start of the invoking list. |
| 7 | **ListIterator listIterator(int index)**  Returns an iterator to the invoking list that begins at the specified index. |
| 8 | **Object remove(int index)**  Removes the element at position index from the invoking list and returns the deleted element. The resulting list is compacted. That is, the indexes of subsequent elements are decremented by one. |
| 9 | **Object set(int index, Object obj)**  Assigns obj to the location specified by index within the invoking list. |
| 10 | **List subList(int start, int end)**  Returns a list that includes elements from start to end.1 in the invoking list. Elements in the returned list are also referenced by the invoking object. |

Add()

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  List a1 = **new** ArrayList();  a1.add(1);  a1.add(2);  a1.add(3);  System.***out***.println(**" ArrayList Elements"**);  System.***out***.println(**"\t"** + a1);   a1.add(1, 10);  System.***out***.println(**" new ArrayList Elements"**);  System.***out***.println(**"\t"** + a1);   *//addall method parameters : <index, collection c>* Collections.*addAll*(a1, **"cow"**, **"dog"**, **"goat"**);  System.***out***.println(**"got agter using addall"**);  System.***out***.println(**"\t"** +a1);   *//get(<index>)* System.***out***.println(**"using get method"**);  System.***out***.println(**"\t"** +a1.get(5));   *//get() -- another* System.***out***.println(**"using get method another example"**);  System.***out***.println(**"\t"** +a1.get(2));   *//indexOf(Object obj)* **int** have = a1.indexOf(**"cow"**);  **int** donthave = a1.indexOf(**"rat"**);  System.***out***.println(**"have \n\t"** + have);  System.***out***.println(**"don't have \n\t"** + donthave);   *//lastIndexof(<OBJECT>)* **int** v = a1.lastIndexOf(**"dog"**);  System.***out***.println(**"last index of \t\n"** + v);   *//remove(object) & removeall(object)* a1.remove(**"dog"**);  System.***out***.println(**"using remove method"**);  System.***out***.println(**"\t"** +a1);   *//set(int Index, Object obj)  // UPDATES THE ELEMENT -- DELETS THE VALUE AND OVE RRIDES IT   //a1.set(5, "GAS"); //[1, 10, 2, 3, cow, GAS]* a1.set(4, **"GAS"**);  System.***out***.println(**"using set method"**);  System.***out***.println(**"\t"** +a1);   *//WHERE AS add() METHOD ADDS THE VALUES AT A PARTICULAR PLACE  //if an index is found, and that index has a value then,  // add<index, Object> METHOD PUSHES THE ELEMENT RIGHT.  //NEVER OVERDIDES ANY, like .set<index, Object> method* a1.add(1, **"CATS"**);  System.***out***.println(**"using ADD<INDEX, OBJECT> method"**);  System.***out***.println(**"\t"** +a1);   *//sublist(<int fromIndex>, <int toIndex>)   // returntupe - List  //cuts a particular part of a list by the index* List x = a1.subList(1, 4);  System.***out***.println(**"Sublist of the list is here"**);  System.***out***.println(**"\t "** + x);  } } | ArrayList Elements  [1, 2, 3]  new ArrayList Elements  [1, 10, 2, 3]  got agter using addall  [1, 10, 2, 3, cow, dog, goat]  using get method  dog  using get method another example  2  have  4  don't have  -1  last index of  5  using remove method  [1, 10, 2, 3, cow, goat]  using set method  [1, 10, 2, 3, GAS, goat]  using ADD<INDEX, OBJECT> method  [1, CATS, 10, 2, 3, GAS, goat]  Sublist of the list is here  [CATS, 10, 2] |

listIterator()

|  |
| --- |
| CHECK 36\_COLLECTION INTERFACE |