Java List<E> Interface Quick Reference Guide

What is List<E> Interface?

List<E> Interface is an ***ordered collection*** or a collection that ***maintains an insertion order***. Hence we can have control of the ***values based on index***. ***ArrayList<E>*** and ***LinkedList<E>*** are popular implementation of List<E> interface. ArrayList<E> uses array data structure to store elements where as LinkedList<E> uses Linked List (doubly Linked List) data structure to store elements.

|  |  |
| --- | --- |
| Method | Description |
|  |  |
| int size() | Returns the ***size*** of the Collection. |
| boolean isEmpty() | Returns true if the ***size*** (of List) ***== 0***  Else it returns false. |
| boolean contains(Object o) | Returns true if this List *contains* Object o such that  ***o == null ? e == null : o.equals(e)*** |
| Iterator<E> iterator() | Returns the ***iterator*** that will step over or iterate over the elements of List. It does ***guarantee the proper sequence***. |
| Object[] toArray() | Returns the ***Object[]*** of the elements of Collection. This method returns the ***deep copy*** *meaning* all the elements of Collections are copied to Object[] and Object[] does not maintain any references to Collection. |
| <T> T[] toArray(T[] a) | Returns an ***array*** containing elements of Collection. The returned array is of type runtime type of array to contain elements. |
| boolean add(E e) | This method adds the element to the ***end of the List*** or this method append the element to the List. |
| boolean remove(Object o) | This method removes the first occurrence of Object o from the Collection. This method removes single element from List such that  ***o == null ? e == null : o.equals(e)***. Returns true if the element is removed from Collection else returns false. |
| boolean  containsAll(Collection<?> c) | This method returns true if this ***List contains*** all the elements specified in Collection<?> c passed as parameter. |
| boolean  addAll(Collection<? extends E> c) | addAll method is used to ***add all* or append all** the elements of Collection<? extends E> c to this List. It returns true if specified List is changed because of this call. |
| boolean  addAll(int index, Collection<? extends E>) c | This overloaded version of addAll method is used to insert the Collection into the ***specified index*** of this List. |
| boolean  removeAll(Collection<?> c) | removeAll method is used to ***remove all*** the elements from List that are specified in Collection<?> c. Returns true if this List is changed because of this call. |
| default void  replaceAll(UnaryOperator<E> operator) | replaceAll method ***replaces*** each element in List with the ***result of applying operator***. This method is added in Java 8. |
| boolean  retainAll(Collection<?> c) | retainAll method ***retains*** *or holds* elements that are in this List specified by Collection<?> c in parameter. Returns true if the Collection is changed as a result of this call. |
| default void  sort(Comparator<? super E>) c | Sorts the List according to the order provided by the ***Comparator***. |
| void clear() | clear method is used to ***clear the List*** or remove all the elements from this Collection. |
| boolean equals(Object o) | equals method ***compares this List with Object o*** specified in parameter for equality. Returns true if *Object o* is equal to this Collection. |
| int hashCode() | Returns the hash code for this List. Remember ***list1.equals(list2)*** implies that  ***list1.hashCode() == list2.hashCode()***. |
| E get(int index) | get method returns the element which is at that ***index***. |
| E set(int index, E element) | set method is used to ***set the element E at the index*** specified in parameter. |
| void add(int index, E element) | This overloaded version of add method is used to ***insert the element E at index*** provided in parameter. |
| E remove(int index) | This method is used to ***remove*** element ***from*** the ***index***. |
| int indexOf(Object o) | indexOf method is used to ***search*** for the required Object in the List. If found it returns the index of the Object. Search is done as  ***o == null ? e == null : o.equals(e)***. This method scans the List from beginning. If element is not found it returns -1. |
| int lastIndexOf(Object o) | Returns the ***last index of the Object*** o in the List. lastIndexOf method works same as indexOf but it ***searches for the Object from end of the List***. |
| ListIterator<E> listIterator() | It returns a list iterator that provides you several operations like ***hasNext()***, **next()**, ***hasPrevious(), previous(), nextIndex(), previousIndex(), remove(), set(E)*** and ***add(E)***. We can traverse ***bi-directional*** through this iterator. |
| List<E>  subList(int fromIndex, int toIndex) | subList method returns a ***sub list*** from *fromIndex* and *toIndex exclusive.* |
| Spliterator<E> spliterator() | Creates a ***spliterator*** over the elements in this Collection. |