

ArrayList Methods in Java



All methods of ArrayList in Java



The `ArrayList` class in Java is a part of the `java.util` package and implements the `List` interface, which allows for dynamic arrays that can grow as needed. Here is a list of common methods available in the `ArrayList` class:

1. Basic Methods

- `boolean add(E e)`: Appends the specified element to the end of the list.
- `void add(int index, E element)`: Inserts the specified element at the specified position in this list.
- `boolean addAll(Collection<? extends E> c)`: Appends all elements in the specified collection to the end of the list.
- `boolean addAll(int index, Collection<? extends E> c)`: Inserts all elements in the specified collection at the specified position in this list.
- `E get(int index)`: Returns the element at the specified position in this list.
- `E set(int index, E element)`: Replaces the element at the specified position in this list with the specified element.

2. Size and Capacity

- `int size()`: Returns the number of elements in this list.
- `boolean isEmpty()`: Returns `true` if this list contains no elements.
- `void ensureCapacity(int minCapacity)`: Increases the capacity of this `ArrayList` instance, if necessary, to ensure that it can hold at least the number of elements specified by the `minCapacity`.

3. Removal Methods

- `E remove(int index)`: Removes the element at the specified position in this list.
- `boolean remove(Object o)`: Removes the first occurrence of the specified element from this list, if it is present.
- `boolean removeAll(Collection<?> c)`: Removes from this list all of its elements that are contained in the specified collection.
- `boolean retainAll(Collection<?> c)`: Retains only the elements in this list that are contained in the specified collection.
- `void clear()`: Removes all of the elements from this list.

4. Searching Methods

- `boolean contains(Object o)`: Returns `true` if this list contains the specified element.

- `int indexOf(Object o)`: Returns the index of the first occurrence of the specified element in this list.
- `int lastIndexOf(Object o)`: Returns the index of the last occurrence of the specified element in this list.

5. Iteration Methods

- `Iterator<E> iterator()`: Returns an iterator over the elements in this list.
- `ListIterator<E> listIterator()`: Returns a list iterator over the elements in this list.
- `ListIterator<E> listIterator(int index)`: Returns a list iterator over the elements in this list, starting at the specified position.
- `Splitter<E> splitter()`: Creates a late-binding and fail-fast splitter over the elements in this list.

6. Miscellaneous Methods

- `Object[] toArray()`: Returns an array containing all of the elements in this list in proper sequence.
- `<T> T[] toArray(T[] a)`: Returns an array containing all of the elements in this list in proper sequence, with the runtime type of the returned array being that of the specified array.
- `boolean equals(Object o)`: Compares the specified object with this list for equality.
- `int hashCode()`: Returns the hash code value for this list.
- `void trimToSize()`: Trims the capacity of this `ArrayList` instance to be the list's current size.
- `List<E> subList(int fromIndex, int toIndex)`: Returns a view of the portion of this list between the specified `fromIndex`, inclusive, and `toIndex`, exclusive.

These methods provide functionality for managing dynamic arrays in Java, such as adding, removing, accessing, and searching for elements.



Example for each method



Here's an example for each method of the `ArrayList` class in Java. I've grouped them by their functionality and provided simple explanations to help you understand how they work.

1. Basic Methods

- `add(E e)`

java

```
ArrayList<String> list = new ArrayList<>();
list.add("Apple");
System.out.println(list); // Output: [Apple]
```

- `add(int index, E element)`

```
java
```

```
list.add(0, "Banana");
System.out.println(list); // Output: [Banana, Apple]
```

- `addAll(Collection<? extends E> c)`

```
java
```

```
ArrayList<String> otherList = new ArrayList<>();
otherList.add("Orange");
otherList.add("Grapes");
list.addAll(otherList);
System.out.println(list); // Output: [Banana, Apple, Orange, Grapes]
```

- `addAll(int index, Collection<? extends E> c)`

```
java
```

```
ArrayList<String> fruits = new ArrayList<>();
fruits.add("Mango");
fruits.add("Pineapple");
list.addAll(1, fruits);
System.out.println(list); // Output: [Banana, Mango, Pineapple, Apple, Orange, Grapes]
```

- `get(int index)`

```
java
```

```
String fruit = list.get(2);
System.out.println(fruit); // Output: Pineapple
```

- `set(int index, E element)`

```
java
```

```
list.set(2, "Kiwi");
System.out.println(list); // Output: [Banana, Mango, Kiwi, Apple, Orange, Grapes]
```

2. Size and Capacity

- `size()`

```
java
```

```
int size = list.size();
```

```
System.out.println(size); // Output: 6
```

- `isEmpty()`

```
java
```

```
System.out.println(list.isEmpty()); // Output: false
```

- `ensureCapacity(int minCapacity)`

```
java
```

```
list.ensureCapacity(10); // Ensures the capacity of the list is at least 10
```

3. Removal Methods

- `remove(int index)`

```
java
```

```
list.remove(1);
System.out.println(list); // Output: [Banana, Kiwi, Apple, Orange, Grapes]
```

- `remove(Object o)`

```
java
```

```
list.remove("Apple");
System.out.println(list); // Output: [Banana, Kiwi, Orange, Grapes]
```

- `removeAll(Collection<?> c)`

```
java
```

```
ArrayList<String> toRemove = new ArrayList<>();
toRemove.add("Orange");
toRemove.add("Kiwi");
list.removeAll(toRemove);
System.out.println(list); // Output: [Banana, Grapes]
```

- `retainAll(Collection<?> c)`

```
java
```

```
ArrayList<String> toRetain = new ArrayList<>();
toRetain.add("Banana");
```

```
list.retainAll(toRetain);
System.out.println(list); // Output: [Banana]
```

- `clear()`

```
java
```

```
list.clear();
System.out.println(list); // Output: []
```

4. Searching Methods

- `contains(Object o)`

```
java
```

```
list.add("Apple");
System.out.println(list.contains("Apple")); // Output: true
```

- `indexOf(Object o)`

```
java
```

```
int index = list.indexOf("Apple");
System.out.println(index); // Output: 0
```

- `lastIndexOf(Object o)`

```
java
```

```
list.add("Banana");
int lastIndex = list.lastIndexOf("Banana");
System.out.println(lastIndex); // Output: 1
```

5. Iteration Methods

- `iterator()`

```
java
```

```
Iterator<String> it = list.iterator();
while(it.hasNext()) {
    System.out.println(it.next());
}
// Output: Apple, Banana
```

- `listIterator()`

```
java
```

```
ListIterator<String> listIt = list.listIterator();
while(listIt.hasNext()) {
    System.out.println(listIt.next());
}
// Output: Apple, Banana
```

- `listIterator(int index)`

```
java
```

```
ListIterator<String> listItFromIndex = list.listIterator(1);
while(listItFromIndex.hasNext()) {
    System.out.println(listItFromIndex.next());
}
// Output: Banana
```

6. Miscellaneous Methods

- `toArray()`

```
java
```

```
Object[] array = list.toArray();
System.out.println(Arrays.toString(array)); // Output: [Apple, Banana]
```

- `toArray(T[] a)`

```
java
```

```
String[] strArray = list.toArray(new String[0]);
System.out.println(Arrays.toString(strArray)); // Output: [Apple, Banana]
```

- `equals(Object o)`

```
java
```

```
ArrayList<String> anotherList = new ArrayList<>();
anotherList.add("Apple");
anotherList.add("Banana");
System.out.println(list.equals(anotherList)); // Output: true
```

- `hashCode()`

```
java
```

```
int hashCode = list.hashCode();  
System.out.println(hashCode);
```

- `subList(int fromIndex, int toIndex)`

```
java
```

```
List<String> subList = list.subList(0, 1);  
System.out.println(subList); // Output: [Apple]
```

- `trimToSize()`

```
java
```

```
list.trimToSize(); // Reduces storage capacity to current size
```

This covers the common methods of the `ArrayList` class with examples! Let me know if you'd like further clarification on any method.

ChatGPT can make mistakes. Check important info.