List Interface in Java with Examples
AbstractList in Java with Examples
ArrayList in Java
LinkedList in Java
Immutable List in Java
Custom ArrayList in Java
Java Collections | Difference between Synchronized ArrayList and CopyOnWriteArrayList
How to remove a SubList from a List in Java
Randomly select items from a List in Java
Get first and last elements from ArrayList in Java
How to Remove Duplicates from ArrayList in Java
How to get ArrayList from Stream in Java 8
How to convert LinkedList to Array in Java?
How to make an ArrayList read only in Java
Join two ArrayLists in Java
Find common elements in two ArrayLists in Java
Find first and last element of ArrayList in Java
Convert an Iterator to a List in Java
Polymorphism in Java
Maximum absolute difference in an array
What's the connection between Java and Blockchain?
Difference between static and non-static method in Java
Spring Dependency Injection with Example
Count occurrences of a given character using Regex in Java
File Handling in Java with CRUD operations
What is the Role of Java in the IT Industry?

CopyOnWriteArrayList in java

CopyOnWriteArrayList: CopyOnWriteArrayList class is introduced in JDK 1.5, which implements List interface. It is enhanced version of ArrayList in which all modifications (add, set, remove, etc) are implemented by making a fresh copy.

Fig: CopyOnWriteArrayList

Here are few points about CopyOnWriteArrayList:

- As the name indicates, CopyOnWriteArrayList creates a Cloned copy of underlying ArrayList, for every update operation at certain point both will be synchronized automatically ,which is taken care by JVM. Therefore there is no effect for threads which are performing read operation. CopyOnWriteArrayList is the best choice if our frequent operation is read operation.
- It extends object and implements Serializable, Cloneable, Iterable<E>, Collection<E>, List<E> and RandomAccess
- The underlined data structure is grow-able array.
- It is thread-safe version of ArrayList.
- Insertion is preserved, duplicates are allowed and heterogeneous Objects are allowed.
- The main important point about CopyOnWriteArrayList is Iterator of CopyOnWriteArrayList can not perform remove operation else we will get Run-time exception saying UnsupportedOperationException.

Constructors Summary:

- **CopyOnWriteArrayList c = new CopyOnWriteArrayList():** Creates an empty list.
- **CopyOnWriteArrayList c = new CopyOnWriteArrayList(Collection obj):** Creates a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.
- **CopyOnWriteArrayList c = new CopyOnWriteArrayList(Object[] obj):** Creates a list holding a copy of the given array.

Methods Summary:

1. **add(int index, E element):** This method Inserts the specified element at the specified position in this list.
2. **add(E e):** This method Appends the specified element to the end of this list.
3. **addAll(int index, Collection<E> c):** This method Inserts all of the elements in the specified collection into this list, starting at the specified position.
4. **addAll(Collection<E> c):** This method Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.
5. **addAllAbsent(Collection<E> c):** This method Appends all of the elements in the specified collection that are not already contained in this list, to the end of this list, in the order that they are returned by the specified collection's Iterator.
6. **addIfAbsent(E e):** This method Appends the element, if not present.
7. **clear():** This method Removes all of the elements from this list.
8. **clone():** This method Returns a shallow copy of this list.
9. **contains(Object o):** This method Returns true if this list contains the specified element.
10. **containsAll(Collection<E> c):** This method Returns true if this list contains all of the elements of the specified collection.
11. **equals(Object o):** This method Compares the specified object with this list for equality.
12. **forEach(Consumer<E> action):** This method Performs the given action for each element of the Iterable until all elements have been processed or the action throws an exception.
13. **get(int index):** This method Returns the element at the specified position in this list.
14. **hashCode():** This method Returns the hash code value for this list.
15. **indexOf(E e, int index):** This method Returns the index of the first occurrence of the specified element in this list, searching forwards from index, or returns -1 if the element is not found.
16. **indexOf(Object o):** This method Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
17. **isEmpty():** This method Returns true if this list contains no elements.
18. **iterator():** This method Returns an iterator over the elements in this list in proper sequence.
19. **lastIndexOf(E e, int index):** This method Returns the index of the last occurrence of the specified element in this list, searching backwards from index, or returns -1 if the element is not found.
20. **lastIndexOf(Object o):** This method Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
21. **listIterator():** This method Returns a list iterator over the elements in this list (in proper sequence).
22. **listIterator(int index):** This method Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list.
23. **remove(int index):** This method Removes the element at the specified position in this list.
24. **remove(Object o):** This method Removes the first occurrence of the specified element from this list, if it is present.
25. **removeAll(Collection<E> c):** This method Removes from this list all of its elements that are contained in the specified collection.
26. **removeIf(Predicate<E> filter):** This method Removes all of the elements of this collection that satisfy the given predicate.
27. **replaceAll(UnaryOperator<E> operator):** This method Replaces each element of this list with the result of applying the operator to that element.
28. **retainAll(Collection<E> c):** This method Retains only the elements in this list that are contained in the specified collection.
29. **set(int index, E element):** This method Replaces the element at the specified position in this list with the specified element.
30. **size():** This method Returns the number of elements in this list.
31. **sort(Comparator<E> c):** This method Sorts this list according to the order induced by the specified Comparator.
32. **spliterator():** This method Returns a Spliterator over the elements in this list.
33. **subList(int fromIndex, int toIndex):** This method Returns a view of the portion of this list between fromIndex, inclusive, and toIndex, exclusive.
34. **toArray():** This method Returns an array containing all of the elements in this list in proper sequence (from first to last element).
35. **toArray(T[] a):** This method Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array.
36. **toString():** This method Returns a string representation of this list.

```
// Java program to illustrate
// CopyOnWriteArrayList class
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.*;

class ConcurrentDemo extends Thread {

    static CopyOnWriteArrayList l = new CopyOnWriteArrayList();

    public void run()
    {
        // Child thread trying to
        // add new element in the
        // Collection object
        l.add("D");
    }

    public static void main(String[] args)
        throws InterruptedException
    {
        l.add("A");
        l.add("B");
        l.add("C");

        // We create a child thread
        // that is going to modify
        // ArrayList l.
        ConcurrentDemo t = new ConcurrentDemo();
        t.run();

        Thread.sleep(1000);

        // Now we iterate through
        // the ArrayList and get
        // exception.
        Iterator itr = l.iterator();
        while (itr.hasNext()) {
            String s = (String)itr.next();
            System.out.println(s);
            Thread.sleep(1000);
        }
        System.out.println(l);
    }
}
```

Output:

```
A
B
C
D
[A, B, C, D]
```


Recommended Posts:

CopyOnWriteArrayList get() method in Java
CopyOnWriteArrayList add() method in Java
CopyOnWriteArrayList contains() method in Java
CopyOnWriteArrayList isEmpty() method in Java
CopyOnWriteArrayList spliterator() method in Java
CopyOnWriteArrayList containsAll() method in Java
CopyOnWriteArrayList set() method in Java with Examples
CopyOnWriteArrayList toArray() method in Java
CopyOnWriteArrayList listIterator() method in Java
CopyOnWriteArrayList iterator() method in Java
CopyOnWriteArrayList lastIndexOf() method in Java
CopyOnWriteArrayList toString() method in Java
CopyOnWriteArrayList size() method in Java
CopyOnWriteArrayList hashCode() method in Java
CopyOnWriteArrayList addIfAbsent() method in Java

Bishal Kumar Dubey
Hello everyone, I am Bishal KUMAR Dubey and most importantly an idea creator I just love Programming languages and love to know new concepts everyday,every minute,every second Here to help Other GEEKS

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Improved By : vinaykumar54, randhish79, rahuicoolude05

Article Tags: Java Java - util package Java-ArrayList Java-Collections Java-CopyOnWriteArrayList java-list
Practice Tags: Java Java-Collections

Previous
How to measure time taken by a function in java ?
Next
Difference between ArrayList and CopyOnWriteArrayList

Most popular in Java

Java program to check whether a string is a Palindrome
Recursion in Java
Differences between Interface and Class in Java
Difference between concat() and + operator in Java
ClassLoader in Java

More related articles in Java

Difference between String and Character array in Java
Integer.parseInt(): Method in Java with Examples
Integer.valueOf(): Method in Java with Examples
Count the pairs of vowels in the given string
How to overload and override main method in Java