JAVA - THE COLLECTION ALGORITHMS

The collections framework defines several algorithms that can be applied to collections and maps.

These algorithms are defined as static methods within the Collections class. Several of the methods can throw a **ClassCastException**, which occurs when an attempt is made to compare incompatible types, or an **UnsupportedOperationException**, which occurs when an attempt is made to modify an unmodifiable collection.

The methods defined in collection framework's algorithm are summarized in the following table:

SN	Methods with Description
1	**static int binarySearch(List list, Object value, Comparator c)** Searches for value in list ordered according to c. Returns the position of value in list, or -1 if value is not found
2	**static int binarySearch(List list, Object value)** Searches for value in list. The list must be sorted. Returns the position of value in list, or -1 if value is not found.
3	**static void copy(List list1, List list2)** Copies the elements of list2 to list1.
4	**static Enumeration enumeration(Collection c)** Returns an enumeration over c.
5	**static void fill(List list, Object obj)** Assigns obj to each element of list.
6	**static int indexOfSubList(List list, List subList)** Searches list for the first occurrence of subList. Returns the index of the first match, or .1 if no match is found.
7	**static int lastIndexOfSubList(List list, List subList)** Searches list for the last occurrence of subList. Returns the index of the last match, or .1 if no match is found.
8	**static ArrayList list(Enumeration enum)** Returns an ArrayList that contains the elements of enum.
9	**static Object max(Collection c, Comparator comp)** Returns the maximum element in c as determined by comp.
10	**static Object max(Collection c)**

Returns the maximum element in c as determined by natural ordering. The collection need not be sorted.

11 **static Object min(Collection c, Comparator comp)**

Returns the minimum element in c as determined by comp. The collection need not be sorted.

12 **static Object min(Collection c)**

Returns the minimum element in c as determined by natural ordering.

13 **static List nCopies(int num, Object obj)**

Returns num copies of obj contained in an immutable list. num must be greater than or equal to zero.

14 **static boolean replaceAll(List list, Object old, Object new)**

Replaces all occurrences of old with new in list. Returns true if at least one replacement occurred. Returns false, otherwise.

15 **static void reverse(List list)**

Reverses the sequence in list.

16 **static Comparator reverseOrder( )**

Returns a reverse comparator

17 **static void rotate(List list, int n)**

Rotates list by n places to the right. To rotate left, use a negative value for n.

18 **static void shuffle(List list, Random r)**

Shuffles (i.e., randomizes) the elements in list by using r as a source of random numbers.

19 **static void shuffle(List list)**

Shuffles (i.e., randomizes) the elements in list.

20 **static Set singleton(Object obj)**

Returns obj as an immutable set. This is an easy way to convert a single object into a set.

21 **static List singletonList(Object obj)**

Returns obj as an immutable list. This is an easy way to convert a single object into a list.

22 **static Map singletonMap(Object k, Object v)**

Returns the key/value pair k/v as an immutable map. This is an easy way to convert a single key/value pair into a map.

23 **static void sort(List list, Comparator comp)**

Sorts the elements of list as determined by comp.

24 **static void sort(List list)**

Sorts the elements of list as determined by their natural ordering.

25 **static void swap(List list, int idx1, int idx2)**

Exchanges the elements in list at the indices specified by idx1 and idx2.

26 **static Collection synchronizedCollection(Collection c)**

Returns a thread-safe collection backed by c.

27 **static List synchronizedList(List list)**

Returns a thread-safe list backed by list.

28 **static Map synchronizedMap(Map m)**

Returns a thread-safe map backed by m.

29 **static Set synchronizedSet(Set s)**

Returns a thread-safe set backed by s.

30 **static SortedMap synchronizedSortedMap(SortedMap sm)**

Returns a thread-safe sorted set backed by sm.

31 **static SortedSet synchronizedSortedSet(SortedSet ss)**

Returns a thread-safe set backed by ss.

32 **static Collection unmodifiableCollection(Collection c)**

Returns an unmodifiable collection backed by c.

33 **static List unmodifiableList(List list)**

Returns an unmodifiable list backed by list.

34 **static Map unmodifiableMap(Map m)**

Returns an unmodifiable map backed by m.

35 **static Set unmodifiableSet(Set s)**

Returns an unmodifiable set backed by s.

36 **static SortedMap unmodifiableSortedMap(SortedMap sm)**

Returns an unmodifiable sorted map backed by sm.

37 **static SortedSet unmodifiableSortedSet(SortedSet ss)**

Returns an unmodifiable sorted set backed by ss.

Example:

Following is the example, which demonstrate various algorithms.

```
import java.util.*;

public class AlgorithmsDemo {

   public static void main(String args[]) {
      // Create and initialize linked list
      LinkedList ll = new LinkedList();
      ll.add(new Integer(-8));
      ll.add(new Integer(20));
      ll.add(new Integer(-20));
      ll.add(new Integer(8));

      // Create a reverse order comparator
      Comparator r = Collections.reverseOrder();
      // Sort list by using the comparator
      Collections.sort(ll, r);
      // Get iterator
      Iterator li = ll.iterator();
      System.out.print("List sorted in reverse: ");
      while(li.hasNext()){
         System.out.print(li.next() + " ");
      }
      System.out.println();
      Collections.shuffle(ll);
      // display randomized list
      li = ll.iterator();
      System.out.print("List shuffled: ");
      while(li.hasNext()){
         System.out.print(li.next() + " ");
      }
      System.out.println();
      System.out.println("Minimum: " + Collections.min(ll));
      System.out.println("Maximum: " + Collections.max(ll));
   }
}
```

This would produce the following result:

```
List sorted in reverse: 20 8 -8 -20
List shuffled: 20 -20 8 -8
Minimum: -20
Maximum: 20
```