The Collection Algorithms

The java collection framework defines several algorithms as static methods that can be used directly without knowledge of their implementation.

All the collection algorithms in java are defined in a class called Collections which is defined in java.util package.

The collection framework has the following methods as algorithms.

`Method`	Description
`void sort(List list)`	Sorts the elements of the list as determined by their natural ordering.
`void sort(List list, Comparator comp)`	Sorts the elements of the list as determined by Comparator comp.
`void reverse(List list)`	Reverses all the elements sequence in the list.
`void rotate(List list, int n)`	Rotates list by n places to the right. To rotate left, use a negative value for n.
`void shuffle(List list)`	Shuffles the elements in the list.

`void shuffle(List list, Random r)`	Shuffles the elements in the list by using r as a source of random numbers.
`void copy(List list1, List list2)`	Copies the elements of list2 to list1.
`List nCopies(int num, Object obj)`	Returns num copies of obj contained in an immutable list. num can not be zero or negative.
`void swap(List list, int idx1, int idx2)`	Exchanges the elements in the list at the indices specified by idx1 and idx2.
`int binarySearch(List list, Object value)`	Returns the position of value in the list (must be in the sorted order), or -1 if the value is not found.
`int binarySearch(List list, Object value, Comparator c)`	Returns the position of value in the list ordered according to c, or -1 if the value is not found.
`int indexOfSubList(List list, List subList)`	Returns the index of the first match of subList in the list, or -1 if no match is found.
`int lastIndexOfSubList(List list, List subList)`	Returns the index of the last match of subList in the list, or -1 if no match is found.
`Object max(Collection c)`	Returns the largest element from the collection c as

	determined by natural ordering.
`Object max(Collection c, Comparator comp)`	Returns the largest element from the collection c as determined by Comparator comp.
`Object min(Collection c)`	Returns the smallest element from the collection c as determined by natural ordering.
`Object min(Collection c, Comparator comp)`	Returns the smallest element from the collection c as determined by Comparator comp.
`void fill(List list, Object obj)`	Assigns obj to each element of the list.
`boolean replaceAll(List list, Object old, Object new)`	Replaces all occurrences of old with new in the list.

Example using few of above methods

```
import java.util.*;
public class Algorithm {

   public static void main(String args[]) {

      // Create and initialize Array list
      ArrayList<Integer> list = new ArrayList<Integer>();
      list.add(-4);
      list.add(21);
      list.add(-21);
      list.add(4);
      System.out.println("Original list:");
      System.out.println(list);

      // Create a reverse order comparator
      Comparator r = Collections.reverseOrder();

      // Sort list by using the comparator
      Collections.sort(list, r);

      // Get iterator
      Iterator li = list.iterator();
      System.out.print("List sorted in reverse: ");

      while(li.hasNext()) {
         System.out.print(li.next() + " ");
      }
      System.out.println();

      Collections.shuffle(list);

      // display randomized list
      li = list.iterator();
      System.out.print("List shuffled: ");

      while(li.hasNext()) {
         System.out.print(li.next() + " ");
      }

      System.out.println();
      System.out.println("Minimum: " + Collections.min(list));
      System.out.println("Maximum: " + Collections.max(list));
   }
}
```

Output:

```
Original list:
[-4, 21, -21, 4]
List sorted in reverse: 21 4 -4 -21
List shuffled: -4 21 -21 4
Minimum: -21
Maximum: 21
```