RemoveAll elements operation on ArrayList.

[List<E>](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-5list-interface.html) has a removeAll(Collection<?> c) method that is used to remove all the elements from this collection that are in specified collection. This operation is a bulk operation as it takes entire collection as input and then performs operations. [ArrayList<E>](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-5list-interface.html) is concrete implementation of List<E> interface.

So in this operation each element is taken from specified collection and searched in this collection. It also removes all the occurrences too.

What removeAll(Collection<?> c) does is it searches for element in this collection using contains(Object o) and if found then it removes it. As we learnt [contains(Object o)](http://data-structure-learning.blogspot.com/2015/05/search-element-in-arraylist.html), time complexity is O(n). And this operation of removeAll(Collection<?> c) is done n times hence removeAll(Collection<?> c)time complexity is O(n).

Below is the program that demonstrate removeAll(Collection<?> c)

**package** org.collections;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ArrayListRemoveAll {

**public** **static** **void** main(String[] args) {

List<Integer> listA = **new** ArrayList<Integer>();

listA.add(17);

listA.add(25);

listA.add(50);

listA.add(17);

listA.add(92);

listA.add(67);

listA.add(89);

listA.add(25);

listA.add(17);

System.***out***.println("ListA " + listA);

List<Integer> listB = **new** ArrayList<Integer>();

listB.add(17);

listB.add(25);

System.***out***.println("ListB " + listB);

listA.removeAll(listB);

System.***out***.println(listA);

}

}

Below is output

ListA [17, 25, 50, 17, 92, 67, 89, 25, 17]

ListB [17, 25]

After removeAll(..) operation [50, 92, 67, 89]