**Delete and Remove**

**Deletion (is an algorithm in which element(s) are removed from the array or ArrayList)** of an array or ArrayList element is the last basic standard algorithm. It is similar to the replacement algorithm in that you are locating a particular element, but, once found, the element is deleted.

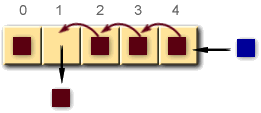
This is the same as flipping through a photo gallery to an image no longer wanted and then discarding it once you find it. You are not modifying the image or adding a new one; you are completely getting rid of it.

For an array, deletion involves removing an element from its position in the array and then shifting any elements after it by one position. The last element will now be duplicated in two positions! We do not want to accidentally have the last to an element twice. To solve this, original location set to null. An alternative option is to replace that element with an empty one. You can choose which method to use.

For an ArrayList, the process is even easier; when an item is deleted, the ArrayList size will automatically decrease by one and all the index positions following that item will shift by one.

### Part 1

The opposite of inserting an element into an array is the act of deleting an element from an array. This involves the removal of the element and the shifting of each subsequent element one place to the left so as to fill the spot formerly occupied by the deleted element. As a result of this shifting, the last array element becomes empty. The typical way to handle this situation is to fill the last array element with a null reference. Occasionally, however, a programmer may choose to fill it with a default element of some kind.



### Part 2

Some say an ArrayList is vastly superior to an array when it comes to insertions and deletions. However, an ArrayList requiring every element to be an object is a trade off every programmer must weigh carefully. Use the following demonstration programs to familiarize yourself with how to efficiently delete items from the two data structures:

* Open the [07.07 Virtual Lecture Notes](https://lti.flvsgl.com/flvs-cat-content/r6enaeb2ajq927me7e124iptme/flvs-cat-session/apcomputersciencea_v20/module07/lesson07/pop/16_04b/16_04a_pop01.htm).
* Create a new project called 07.07 Deletions in the Mod07 Lessons folder.
* Download the following Java files to the newly-created folder:
  + [InventoryItem.java](https://lti.flvsgl.com/flvs-cat-content/r6enaeb2ajq927me7e124iptme/flvs-cat-session/apcomputersciencea_v20/module07/lesson07/docs/16_04b/InventoryItem.java)
  + [TestInventory7.java](https://lti.flvsgl.com/flvs-cat-content/r6enaeb2ajq927me7e124iptme/flvs-cat-session/apcomputersciencea_v20/module07/lesson07/docs/16_04b/TestInventory7.java)
  + [TestInventory8.java](https://lti.flvsgl.com/flvs-cat-content/r6enaeb2ajq927me7e124iptme/flvs-cat-session/apcomputersciencea_v20/module07/lesson07/docs/16_04b/TestInventory8.java)

Don't forget! You cannot just remove an element from a data structure without leaving something to occupy its place. Forgetting this is often a source of error when dealing with the deletion algorithm. If you rushed through this section without recognizing the importance of null, you have missed an important concept that will surely increase the amount of time it takes you to debug programs.