**Replacement Algorithms**

A [**replacement**](javascript:void(0);) **(a replacement algorithm is simply a traversal in which, at some point, one or more items in the array or ArrayList are replaced with new items)** algorithm is simply a traversal in which, at some point, one or more elements in the array or ArrayList have their values are replaced with new ones.

You can compare it to going through a photo slideshow and replacing a few photos with newer or different ones.

The replacement algorithm is the second of the basic algorithms used for arrays and ArrayLists.

To use the replacement algorithm, you simply need a loop to traverse the array or ArrayList and then, at key moments, replace items in the array or ArrayList with new items. This could mean replacing one item or replacing several items that match certain criteria.

A replacement could happen various ways. For example, you replace based on index position by stating a specific index to perform the change. Or you could traverse the entire data structure of elements looking for any item whose name starts with the letter 'p' and update its quantity attribute by one.

### Part 1

Replacing items in a data structure starts with the fundamental traversal algorithm. From there, the key is locating where to perform the replacement and then setting the new values. As you work through the eIMACS labs and the demonstration programs, think about which data structure you prefer.

We now turn our attention to the process of replacement, in which one element of an array is replaced by some other object of the same data type. Before providing an example, we increase the usefulness of our Item class by adding a displayArray class method that traverses the array in order to print out a report of its contents. The following code creates an array of five elements, and then calls the displayArray method. Run the program to verify that it behaves as you expect:

public class Item   
{   
  private int myN;   
  
  public Item( int n )   
  {   
    myN = n;   
  }   
  
  public String toString()   
  {   
    return "Item: " + myN;   
  }   
  
  public int getN()    
  {   
    return myN;   
  }   
  
  public static Item[] makeItemArray( int len )   
  {   
    Item[] a = new Item[ len ];   
    int i;   
    for ( i = 0 ; i < len ; i++ )   
      a[ i ] = new Item( i );   
    return a;   
  }   
  
  public static void displayArray( Item[] array )   
  {   
    for ( Item item : array )   
      System.out.println( item );   
  }   
}   
    
  public static void main( String[] args )   
  {   
    // make the array   
    Item[] array = Item.makeItemArray( 5 );   
  
    // display the array   
    Item.displayArray( array );   
  }

Next, we modify the program slightly so that the second element of the array (that is, the element with index 1) is replaced by a new Item. Run the modified program to verify that it behaves as you expect:

  public static void main( String[] args )   
  {   
    Item[] array = Item.makeItemArray( 5 );   
  
    System.out.println( "Before:" );   
    Item.displayArray( array );   
  
    // replace element at index 1   
    array[ 1 ] = new Item( 99 );   
  
    // traverse the array   
    System.out.println( "\nAfter:" );   
    Item.displayArray( array );   
  }

In this case, once the replacement has been made, the original Item at index 1 can never be accessed again. In fact, the memory in which Java stored the instance variables of this object has probably been reclaimed. To use the terminology that you met earlier, the original element has been [overwritten](https://www.eimacs.com/eimacs/mainpage?epid=E2114552604&cid=162149#Exe041).

### Part 2

In principle, replacing values in an array or an ArrayList is identical (i.e. out with the old, in with the new). However, in practice, the coding involved is quite different. You may remember that arrays don't have any built-in methods, whereas ArrayLists do. Carefully consider the advantages of these two important data structures as you work through these demo programs.

* Open the [07.05 Virtual Lecture Notes](https://lti.flvsgl.com/flvs-cat-content/0hpq9ee8j3f6pj8a4mcu2emkd0/flvs-cat-session/apcomputersciencea_v20/module07/lesson05/pop/16_02b/16_02_virtual_lecture_notes.htm).
* Create a new project called 07.05 Replacements in the Mod07 Lessons folder.
* Download the following Java files to the newly-created folder:
  + [InventoryItem.java](https://lti.flvsgl.com/flvs-cat-content/0hpq9ee8j3f6pj8a4mcu2emkd0/flvs-cat-session/apcomputersciencea_v20/module07/lesson05/docs/16_02b/InventoryItem.java)
  + [TestInventory3.java](https://lti.flvsgl.com/flvs-cat-content/0hpq9ee8j3f6pj8a4mcu2emkd0/flvs-cat-session/apcomputersciencea_v20/module07/lesson05/docs/16_02b/TestInventory3.java)
  + [TestInventory4.java](https://lti.flvsgl.com/flvs-cat-content/0hpq9ee8j3f6pj8a4mcu2emkd0/flvs-cat-session/apcomputersciencea_v20/module07/lesson05/docs/16_02b/TestInventory4.java)
* Carefully read the discussion and carry out the instructions.