**Replacements**

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 );   
  }

[Show program details »](https://www.eimacs.com/eimacs/mainpage?cid=162149&epid=E1942116296)

Item: 0   
Item: 1   
Item: 2   
Item: 3   
Item: 4

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 );   
  }

[Show program details »](https://www.eimacs.com/eimacs/mainpage?cid=162149&epid=E1942116296)

Before:   
Item: 0   
Item: 1   
Item: 2   
Item: 3   
Item: 4   
  
After:   
Item: 0   
Item: 99   
Item: 2   
Item: 3   
Item: 4

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=E2104715197&cid=162149#Exe041).