//I actually didn't really get to use their functions for simplicity gahhh  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
  
**public class** Main {  
 **public static void** main(String[] args) {  
  
 //------------------------------------------------------------------------------//  
 // // ARRAY NUMBER 1 // //  
  
 //it can be done with inserting numbers or string alreaedy into the array to create a new array  
 String[] strings = {"I", "like", "to", "read", "write", "draw", "watch", "anime", "and"};  
  
 // to check the original  
 System.*out*.println("\n" + Arrays.*toString*(strings));  
  
 //------------------------------------------------------------------------------//  
 // // ARRAY NUMBER 2 // //  
  
 // the array is created with the use of, especially, the new Keyword.  
 // I have declared of its contents to be integers (aka numbers)  
 // There will be 10 items (integers) in this array  
 **int**[] nums = **new int**[10];  
  
 // to check the original  
 **for** (**int** num: nums) {  
 System.*out*.print(num + " ");  
 }  
  
  
 //------------------------------------------------------------------------------//  
 System.*out*.println("\n -----------------------------------------");  
 //------------------------------------------------------------------------------//  
  
 // // APPLYING STRING BUILDER // //  
  
 String newStr = *buildString*("more");  
  
 String[] newString = *add*(strings, newStr);  
  
 //printing the new String array  
 System.*out*.println(Arrays.*toString*(newString));  
  
 //------------------------------------------------------------------------------//  
 System.*out*.println("\n -----------------------------------------");  
 //------------------------------------------------------------------------------//  
  
 // // Applying the "pop" // //  
  
 // In this case, it would be 9 because the index will represent the 10th since it starts with 1=0  
 // since I did not assign specific integers to the int array, they are automatically 0s  
 // in this case the 10th 0 is actually the index of 9  
 **int**[] poppedNums = *pop*(nums, 9);  
  
 **for** (**int** newInt : poppedNums) {  
 System.*out*.println(newInt + " ");  
 }  
  
 //------------------------------------------------------------------------------//  
 System.*out*.println("\n -----------------------------------------");  
 //------------------------------------------------------------------------------//  
  
 // // Insert values into array, I need a new list because of the newly imported Array// //  
  
 ArrayList<String> anotherList = **new** ArrayList<>(Arrays.*asList*("sushi", "meatball", "rice", "watermelon"));  
  
 // to check the original  
 System.*out*.println("\n" + anotherList);  
  
 // from the elementtoinsert in the public static void bellow, it now equates to Sashimi  
 String elementToInsert = "sashimi";  
 // and it will be entered after the first word at position 1  
 **int** insertionPosition = 1;  
  
  
 // here, it uses the method to add another element to a specific "index"  
 *insertInMiddle*(anotherList, elementToInsert, insertionPosition);  
 System.*out*.println(anotherList);  
  
 }  
  
//------------------------------------------------------------------------------//  
 //ADD VALUES TO THE END OF AN ARRAY  
//------------------------------------------------------------------------------//  
  
 // This is step 1 of adding the new value for strings  
 **public static** String[] add(String[] strings, String newValue) {  
 // this will mean for the array strings to increase by a length by 1 (when used)  
 **int** popOn = strings.length + 1;  
  
 // creating the new array to copy the old array into (after increasing the length)  
 // basically copy everything except the modifications made  
 String[] newString = **new** String[popOn];  
  
 // this is for copying the int from the old array to the new one  
 **for** (**int** i=0; i<strings.length; i++) {  
 newString[i] = strings[i];  
 }  
  
  
 // add the new value to the last index  
 newString[popOn -1] = newValue;  
  
 **return** newString;  
 }  
  
 //------------------------------------------------------------------------------//  
  
 // Step two use this methode here to add what I need to add more flexibly  
 **public static** String buildString(String str1) {  
 // very useful function to add, delete, alter, insert strings  
 // using sb instead of typing out stringBuilder everytime is much more efficient  
 StringBuilder sb = **new** StringBuilder();  
  
 //append is to add stuff, in this case I only need to add one thing  
 sb.append(str1);  
 **return** sb.toString();  
 }  
  
//------------------------------------------------------------------------------//  
 // DELETE VALUES FROM THE END OF AN ARRAY (OFTEN REFERRED TO AS POP)  
 // Array size should shrink by one.  
//------------------------------------------------------------------------------//  
 **public static int**[] pop(**int**[] nums, **int** index) {  
 // it first checks if the array is not meeting the necessary expectations in order for anything to be removed  
 // if everything is fine, it will just pass and return to nums  
 **if** (nums.length==0 || index < 0 || index >= nums.length) {  
 **return** nums;  
 }  
  
 // this will mean for the array nums to decrease a length by 1 (when used)  
 **int** popOff = nums.length - 1;  
  
 // creating the new array to copy the old array into (after decreasing the length)  
 // basically copy everything except the modifications made  
 **int**[] newSize = **new int**[popOff];  
  
 // this is for copying the int from the old array to the new one  
 // j representing the i for the new array to copy things into  
 // and i in this case have to be shorter than the new num's length  
 **for** (**int** i=0, j=0; i<nums.length; i++) {  
 **if** (i== index) **continue**;  
 newSize[j++] = nums[i];  
 }  
 // after completed copying, the function will return to the newly created array size  
 // this is to 1: ensure that the original data is not altered (to maintain data integrity)  
 // 2: since java arrays are fixed in size, new array is created for any modifications required, and the new array needs to return to the caller of the function  
 **return** newSize;  
 }  
  
//------------------------------------------------------------------------------//  
 //INSERT VALUES INTO AN ARRAY AT CHOSEN INDEX POSITION  
 // Increases the size of the array by 1  
// ------------------------------------------------------------------------------//  
  
 **public static void** insertInMiddle(ArrayList<String> list, String element, **int** position) {  
 // this sets a parameter for the function so that the insertion of the new value for the list should remain in the list not outside  
 **if** (position < 0 || position > list.size()) {  
 System.*out*.println("Invalid position for insertion.");  
 **return**;  
 }  
  
 // method 1...  
// list.add(position, element);  
  
 // method 2  
  
 // to create a new array list  
 ArrayList<String> resultList = **new** ArrayList<>();  
  
 // the for loop is to look through the original array that will be added  
 **for** (**int** i = 0; i<list.size(); i++) {  
 **if** (i == position) {  
 resultList.add(element);  
 }  
 resultList.add(list.get(i));  
 }  
  
 list.clear ();  
 list.addAll ( resultList );  
  
 }  
  
 }