# Review of Arrays and the Need for `ArrayList`



## Review of Arrays

1. Consider the array labeled `arr` that is defined by the following statement.

```
double arr[] = { 1.1, 2.2, 3.3, 4.4, 5.5 };
```

a) Write the expression that gives the number of elements contained in the of the array.

`arr.length`

b) What does the expression `arr[1 + 3]` evaluate to?

`5.5`

c) What does the expression `arr[1] + arr[3]` evaluate to?

`6.6`

2. Consider the array labeled `arr` that is defined by the following statement.

```
int arr[] = new int[5];
```

a) What does the expression `arr[1 + 3]` evaluate to?

`0`

b) What is the result of the statement: `System.out.println( arr[5] );` ?

`ArrayIndexOutOfBoundsException`

3. Consider the following code that performs a linear search algorithm to find the first occurrence of the `String` parameter `value` in the parameter `arr`, which is an array of `String`.

```
1 public static int linearSearch(String[] arr, String value) {
2     for(int i = 0; i < arr.length; i++) {
3         if(arr[i].equals(value)) {
4             return i;
5         }
6     }
7     return -1;
8 }
```

a) Rewrite line 2 such that the algorithm will find the last occurrence of the `String` parameter `value` in the array parameter `arr`. Do not change any other line other than line 2.

```
for(int i = arr.length - 1; i >= 0; i--) {
```

b) In line 3, the `String` method `equals` is used to check if each `String` in array `arr` is equivalent to the `String` provided in the parameter `value`. Explain why this method is used rather than the equals operator ( `==` ).

The equals operator ( `==` ) checks if the reference to each `String` object is the same (the two strings are the exact same object), but we want to know if the contents of the two `String` objects are the same, even if they're not the same object.

4. Consider the following code.

```
1     public static void printArray(String[] arr) {
2     for(int i = 0; i < arr.length; i++) {
3         System.out.println(arr[i]);
4     }
5 }
```

a) Describe the output of this code in a full sentence using correct English.

This code prints out the `String` elements of the array `arr` with one element on each line.

b) Rewrite the method to use an enhanced for loop.

```
1 public static void printArray(String[] arr) {

2     for(String s : arr) {
3         System.out.println(s);
4     }


}
```

5. Write a method named `expandByOne` that takes as parameters an array of `String`, and a single `String`, and returns a new array that contains all the strings from the array parameter (in the same order), plus the `String` parameter appended to the end of the array. The algorithm should follow the following steps.
   - create a new array of `String` with a size one larger than the array parameter
   - copy all the strings from the array parameter into the new array using a loop
   - copy the parameter `value` to the last position of the new array.
   - let the new array be the return value

   The method header has been given for you. The solution will have about five lines of code.

```
1 public static String[] expandByOne(String[] arr, String value) {

2     String[] newArr = new String[arr.length + 1];
3     for(int i = 0; i < arr.length; i++) {
4         newArr[i] = arr[i];
5     }
6     newArr[arr.length] = value;
7     return newArr;


}
```

6. Write a method named `removeOne` that takes as parameters an array of `String`, and an index, and returns a new array that contains all the strings from the array parameter, except with the `String` at position index removed. The algorithm should follow the following steps.
   - create a new array of `String` with a size one smaller than the array parameter
   - copy all the strings from the array parameter up to but not including the `String` at `index` into the new array
   - copy all the strings from the array parameter after the string at index into the new array
   - let the new array be the return value

   The method header has been given for you. Your solution might use two loops (one to copy the strings before the index given by `index`, and another to copy the strings after that index) or a single loop that will copy all values from the first array into the new array (while skipping the value at the index given by `index`). Using two loops will likely result in cleaner and more understandable code.

```
1 public static String[] removeOne(String[] arr, int index) {

2     String[] newArr = new String[arr.length - 1];
3     for(int i = 0; i < index; i++) {
4         newArr[i] = arr[i];
5     }
6     for(int i = index; i < newArr.length; i++) {
7         newArr[i] = arr[i+1];
8     }
9     return newArr;




}
```

## The Need for `ArrayList`

7. Arrays in Java have a fixed size, which means once you create an array, you cannot change it's length.

   If you wish to add a new element to an existing array, you will need to create a totally new array that is larger than the original, copy the contents of the old array to the new array, and add the new element to the appropriate position in the new array. This requires extra code and execution time.

   Similarly if you wish to remove an element from an existing array, you need to again create a whole new array that is one smaller than the original array, and write code that will copy all the elements, except the element that is to be deleted, into the new array – again requiring extra code and execution time.

   Arrays are efficient and useful for relatively static data – when elements are not often added nor removed from the array. However, frequently we have a data set that requires frequent addition and removal of elements. In Java, one simple solution is the `ArrayList` class.

   The `ArrayList` class implements a dynamic array with code similar to what you've written above, but more clever and more optimized. It provides methods to insert, delete, and retrieve values from an ordered list of elements.

   Answer the following in complete sentences.

   a) In Java, if one wishes to add a new element to an existing array, what three steps must be done?

   One must create a totally new array, copy the contents of the old array to the new one, and add the new element.

   b) In Java, if one wishes to remove an element to an existing array, what steps must be done?

   One must create a totally new array, and copy the contents of the old array to the new one – except the one to be deleted.

   c) Give some examples of the types of methods that is provided by the ArrayList class for manipulating an ArrayList objects. (List at least three functions the methods provide).

   The `ArrayList` class provides methods to insert, delete, and retrieve values from an `ArrayList` object.