ResizeCopyArray

You can't resize Array directly. You have 2 options:

1. Create new Array Object with more Array Length and then copy the content to the new Array.

   However, this method involve `for-loop` process which can be expensive depending on the Array Length.

   ```
     // This will return new reference instead of resizing.
     static String[] copyResizeArray(String[] strArray, int newLength) {
       String[] newArray = new String[newLength];
       for (int i = 0; i < strArray.length; i++) {
         // prevent updating array if it's null.
         if (strArray[i] != null) {
           newArray[i] = strArray[i];
         }
       }
       return newArray;
     }
   ```

2. If you are not sure with the length of Array that you're going to use, and constantly adding new item to the Array. You can use `ArrayList`, which `ArrayList` is not shackled with the Array length, typical behavior of Primitive Data Types.

   However, ArrayList wraps primitive data types with object. In theory, this is slower if compared to Primitive Data Types which does not wraps primitive data types. This is because to read or write a primitive data types that is wrapped with object, you add 2 unnecessary process:

   a. Wrapping `process of converting primitive data type to object`

      ```
      char chExample = ‘a’;
      Character ChWrapperExample = chExample;
      ```

b. Unwrapping `process of converting object to primitive data type`

    ```
    char unboxExample = ChWrapperExample;
    ```

This is what happens to primitive data types with ArrayList.

```
List<Character> arrayListExample = new ArrayList<Character>();
arrayListExample(‘a’); // wrapping: convert primitive data type to object.

char chExample = arrayListExample.get(0); // unwrapping: convert object to primitive data type.
```