```
public interface StringList {
  /* Add an element at the end of the list */
  void add(String s);
  /* Get the element at the given index */
  String get(int index);
  /* Get the number of elements in the list */
  int size();
  /* Add an element at the specified index */
  void insert(int index, String s);
  /* Remove the element at the specified index */
  void remove(int index);
import static org.junit.Assert.assertEquals;
import org.junit.Test;
public class TestStringList {
  public void testAddThenGet() {
    StringList slist = new ArrayStringList();
    slist.add("banana");
    slist.add("apple");
    assertEquals( "banana", slist.get(0));
    assertEquals(
                                     , slist.get(1));
  @Test
  public void testAddThenSize() {
    StringList slist = new ArrayStringList();
    slist.add("banana");
    slist.add("apple");
                                     , slist.size());
    assertEquals(
}
```

```
public class ArrayStringList implements StringList {
 String[] elements;
  // How will we construct it?
  // How will we implement the methods?
}
```

.add()	.add()	.get()	.get()	
this	this	this	this	
s	s	index	index	
returns:	returns:	returns:	returns:	
testAddThenGet()				
slist				
returns: nothing (void)				

Stack Method calls and variables Heap Objects, arrays

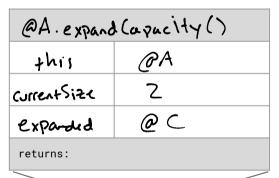
```
public class TestStringList {
  ... all code from other side ...
  @Test
  public void testAddMany() {
    StringList slist = new ArrayStringList();
    slist.add("a");
    slist.add("b");
    slist.add("c");
    slist.add("d");
    slist.add("e");
    assertEquals("e", slist.get(4));
assertEquals("d", slist.get(3));
assertEquals("c", slist.get(2));
assertEquals("b", slist.get(1));
    assertEquals("a", slist.get(0));
public class ArrayStringList {
  ... all code from other side ...
  private void expandCapacity() {
    int currentSize = this.elements.length; 
    if(this.size < currentSize) { return; }
    String[] expanded = new String [concentsize
    for(int i = 0; i < this.5; &C; i += 1) {
    expanded[i] = this.elements[i];
this.elements = expanded;
```

@ A	Arraysting List size: & X Z elements: @B @C
@B	Thut, "a" nott "b"
@ (Thot, "a", aut, "b" aut, "c" I Null
. ¥2}	

Heap Objects, arrays

Key ArrayList idea:

when storage runs out in the array stored in elements, make a new array with more capacity and copy elements over.



@ A.add(`` \^ ')			
this	@A		
S	"a"		
returns: Void			

@ A.add	("b")
this	@A
s	"b"
return	s: void

$Q\!\!A$, add	(" c ")	
this	@A	
S	``c "	
returns:		

. add	()
this		
S		
return	s:	

.add	()
this		
S		
return	s:	

.get	()
this		
S		
return	s:	

testAddMany()

slist

CA

returns: nothing (void)