

```
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 {

    @Test
    public void testAddThenGet() {
        StringList slist = new ArrayStringList();
        slist.add("banana");
        slist.add("apple");

        assertEquals( "banana" , slist.get(0));
        assertEquals( "apple" , slist.get(1));
    }

    @Test
    public void testAddThenSize() {
        StringList slist = new ArrayStringList();
        slist.add("banana");
        slist.add("apple");

        assertEquals( 2 , slist.size());
    }
}
```

```
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)	


```

public class TestStringList {
    ... all code from other side ...
    @Test
    public void testAddMany() {
        StringList slist = new ArrayList();
        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 ArrayList {
    ... all code from other side ...
    private void expandCapacity() {
        int currentSize = this.elements.length;
        if(this.size < currentSize) { return; }

        String[] expanded = new String[currentSize * 2];

        for(int i = 0; i < this.size; i += 1) {
            expanded[i] = this.elements[i];
        }
        this.elements = expanded;
    }
}

```

@A	ArrayList size: 0 2 elements: @B @C
@B	[null, "a" null "b"
@C	[null, "a" null, "b" null, "c"
	[null

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.expandCapacity()	
this	@A
currentSize	2
expanded	@C
returns:	

@A.add("a")	@A.add("b")	@A.add("c")	.add()	.add()	.get()
this: @A	this: @A	this: @A	this:	this:	this:
s: "a"	s: "b"	s: "c"	s:	s:	s:
returns: void	returns: void	returns:	returns:	returns:	returns:

testAddMany()	
slist	@A
returns: nothing (void)	

Stack

Method calls and variables