



SuperArray

Arrays - Disadvantage in Java

- An array is a container object that holds a fixed number of values of a single type.
- The length of an array is established when the array is created.
- **After creation, its length is fixed.**

How we resize an array and add more elements?



SuperArray Class

It will create a dynamic array.

It will behave as a list in Python.



Instance variables

data => Array of String

size => Number of actual elements in the array

Important: size and length are not the same

data.length: the total capacity

size: elements that have been added to the array



Constructors

You may create a SuperArray object like this:

```
SuperArray supArr = new SuperArray()
```

How should you define your constructor?


```
public SuperArray(){  
    // initialize instance variables  
    // Indicate the array capacity  
}
```

You may create a SuperArray object indicating the initial capacity

```
SuperArray supArr = new SuperArray(10)
```

How should you define your constructor?

```
public SuperArray(int initialCapacity){  
    // initialize instance variables  
    // Assign the array capacity  
    // using the parameter  
}
```



Methods

Add: `supArr.add("First")`

Print: It will print `["First"]`. It does not print null elements.

Get size: Return the size

`data => ["First", ?, ?, ?, ?, ?,`

`size => 1`


Adding more:

`supArr.add("Second")`

`supArr.add("Third")`

`data => ["First", "Second", "Third", ?, ?, ?, ?,`

`size => 3`



Methods

Get a single value by index: `supArr.get(0)` => "First"

Set a single value by index: `supArr.set(0, "Primero")` => Will replace it with "Primero"

Remove by index: `supArr.remove(0)` => Removes element at index 0

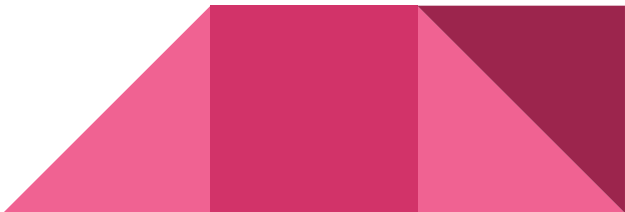
```
data => ["Second", "Third", ?, ?, ?, ?, .....]
```

```
size => 2
```

Remove by value: `supArr.remove("Third")` => Removes the leftmost element that matches the string received as parameter

```
data => ["Second", ?, ?, ?, ?, ?, .....]
```

```
size => 1
```



SuperArray and null values

```
index:  0   1       2       3   4   5   6   7
data : [null, null, "Halloween", null, null, null, null, null]
size : 3
```

Even though `data[0]` and `data[1]` are null, they are still part of the list because they are within the first `size` elements (indexes 0–2).

Why “null” can’t be used to check for valid elements?

If you tried to tell which elements are in the list by checking whether they are null, it wouldn’t work because null could be a **valid value** that was intentionally added by the user.

That’s why we need the **size field**:

- It keeps track of how many elements the user has added.
 - Anything at an index **less than size** is a valid element (even if it’s null).
 - Anything **at or beyond size** is just unused capacity.
- 