

# ICS-211 Lab

## Assignment 2

### List Arrays

# Example

null	null	null	null
------	------	------	------

Space is initially allocated for some number of elements.

# Example

1. add(1)

1	null	null	null
---	------	------	------

Size = 1

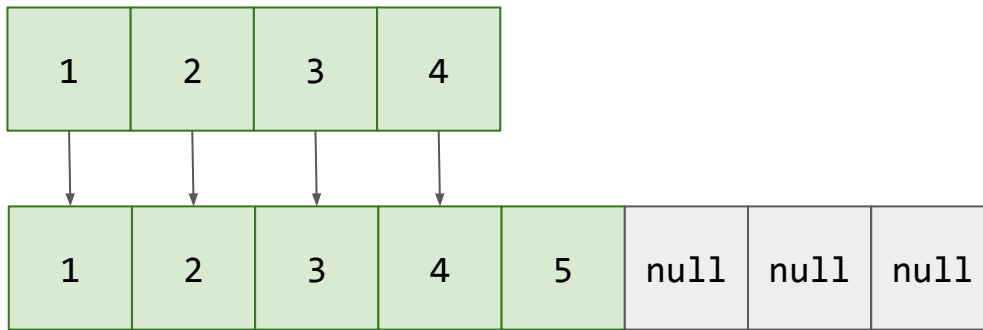
# Example

1	2	3	4
---	---	---	---

Size = 4

1. `add(1)`
2. `add(2), add(3), add(4)`

# Example



1. add(1)
2. add(2), add(3), add(4)
3. add(5)

Size = 5

If there is insufficient space, a new, bigger array is allocated and the old one is copied to the new one.

# Example

1	2	88	4	5	null	null	null
---	---	----	---	---	------	------	------

1. `add(1)`
2. `add(2)`, `add(3)`, `add(4)`
3. `add(5)`
4. `set(2, 88) → 3`

Size = 5

Element 2 is replaced by "88" and "3" is returned to caller.

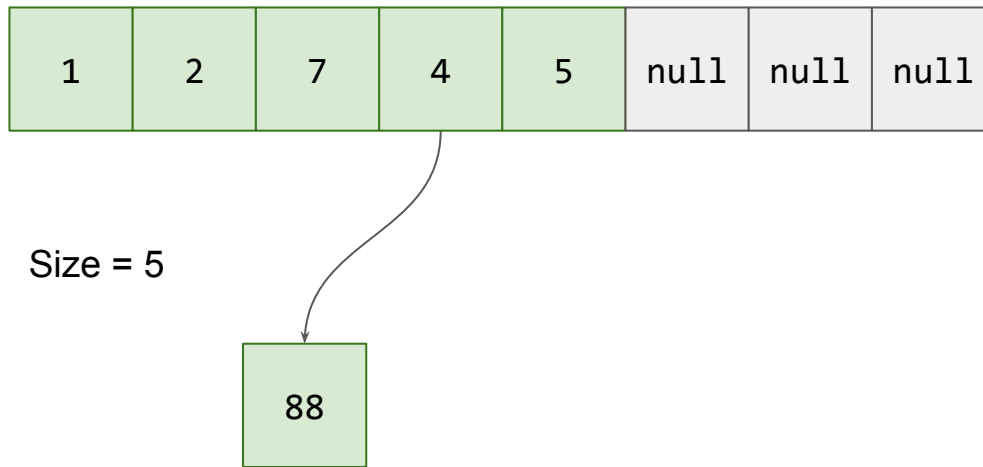
# Example

1	2	7	88	4	5	null	null
---	---	---	----	---	---	------	------

Size = 6

1. `add(1)`
2. `add(2), add(3), add(4)`
3. `add(5)`
4. `set(2, 88) → 3`
5. `add(2, 7)`

# Example



1. add(1)
2. add(2), add(3), add(4)
3. add(5)
4. set(2, 88)  $\rightarrow$  3
5. add(2, 7)
6. remove(3)  $\rightarrow$  88



# Example

1	2	7	4	5	null	null	13
---	---	---	---	---	------	------	----

Size = 5

1. `add(1)`
2. `add(2), add(3), add(4)`
3. `add(5)`
4. `set(2, 88) → 3`
5. `add(2, 7)`
6. `remove(3) → 88`
7. `add(7, 13)`

# Part 1 - Implementing an “Array List”

- An ArrayList is similar to a primitive Java array except:
  - Java arrays can contain only a fixed number of elements
  - An array list will “grow” to hold any number of elements
  - An array list uses a primitive Java array
- Allocating a generic array:

```
data = (T[]) new Object[newSize];
```

→ Eclipse will generate a warning about this (you can safely ignore)

- You are implementing the List211 interface (**not** the Java List interface)
- The interface (with comments) is available on the class GitHub repo <https://github.com/psoulier/ics211-fall16>
- Adding element beyond “size+1” will create a “gap” of null elements.
  - Sorting must take this into account
  - Can be handled in the “Comparator” or sort methods themselves
  - The null elements should be at the end of the sorted list

# Example

1	2	4	5	7	13	null	null
---	---	---	---	---	----	------	------

Size = 5

After sort, null elements must be at the end of array.

1. add(1)
2. add(2), add(3), add(4)
3. add(5)
4. set(2, 88) → 3
5. add(2, 7)
6. remove(3) → 88
7. add(7, 13)
8. sort(cmp)

## Part 2 - Contact List

- For the sorting methods, you should use your ArraySort class
  - There's no need to re-implement this code or copy-paste it (although you can)
  - Create an instance of ArraySort and use the data member from MyArrayList
- The easiest way to implement ContactList is to inherit from MyArrayList
  - Override the appropriate methods
  - Pick one of the sorting methods (doesn't really matter which)