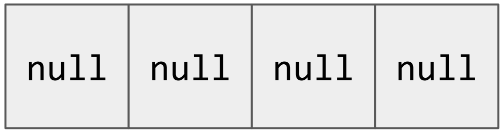
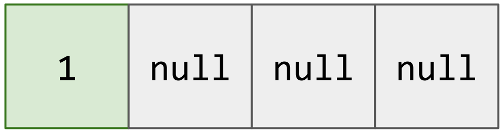
ICS-211 Lab
Assignment 2
List Arrays

Example



Space is initially allocated for some number of elements.

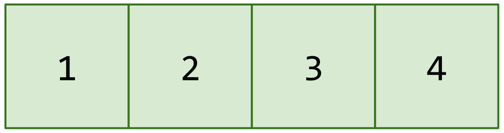
Example



1. add(1)

Size = 1

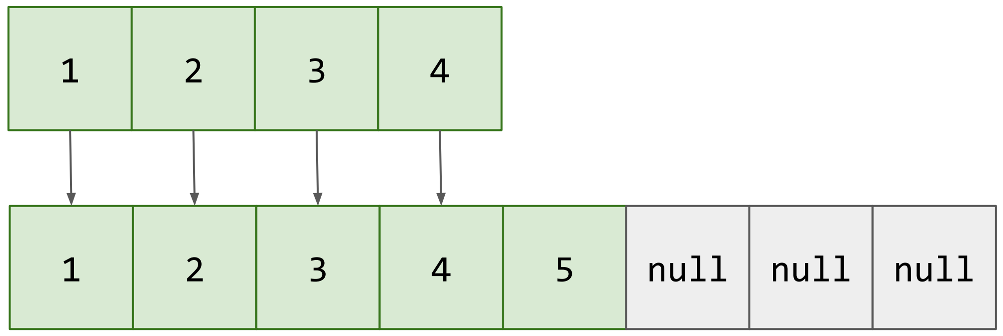
Example



Size = 4

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

Example

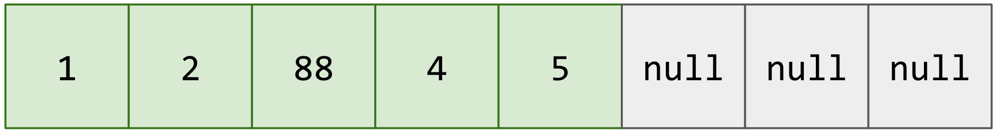


1. add(1)
2. add(2), add(3), add(4)
3. add(5)

Size = 5

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

Example

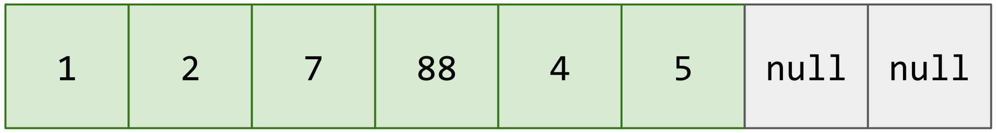


1. add(1)
2. add(2), add(3), add(4)
3. add(5)
4. set(2, 88) → 3

Size = 5

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

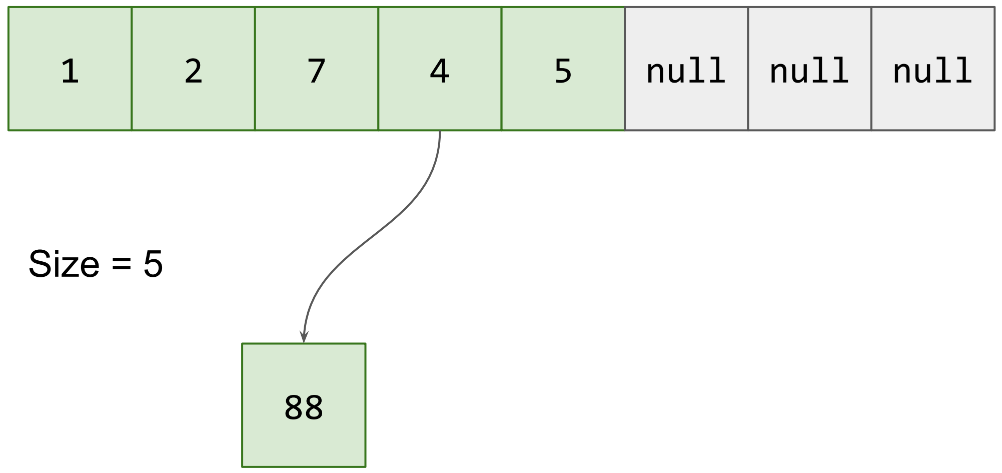
Example



Size = 6

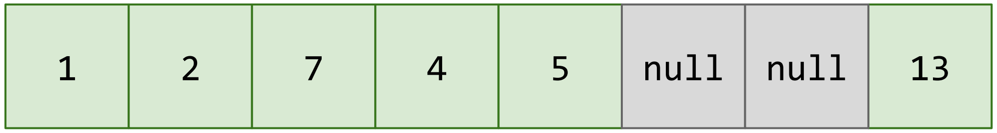
1. add(1)
2. add(2), add(3), add(4)
3. add(5)
4. set(2, 88) → 3
5. add(2, 7)

Example



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

Example



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”

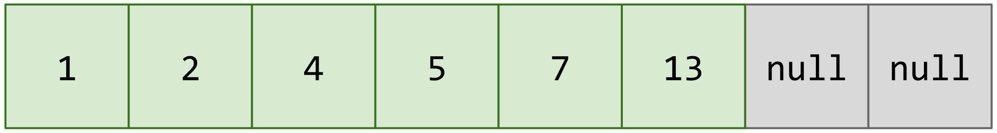
- An ArrayList is similar to a primitive Java array except:
  - Java arrays can contain only a fixed number of elements
  - An array list will “grow” to hold any number of elements
  - An array list uses a primitive Java array
- Allocating a generic array:

```
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



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`
- An array list may contain null elements (e.g., from a resize). Several different approaches:
    - Your sort methods must account for this
    - The comparator must account for this
    - You must only sort the portion of the backing array that contains valid elements
- 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)
    - Technically, this isn't really good programming
        - "Is-a" vs. "Has-a" relationship
        - `MyArrayList` has methods that don't make sense with a sorted list

Unit Tests

- MyArrayList
  - I have provided a lot of tests
  - You’ll need to implement tests for remove method
- ContactList
  - I have provided a basic test for this
  - You may wish to expand to ensure complete coverage
- Unit tests are in the “a2” directory