(We started with the review of cyclic arrays on page 3)

# Lec 13 handout

**Review**: Implementing wrap-around/cyclic arrays

Pink for ArrList position
green for Array index

```
public class ArrList { // like java ArrayList
   private String[] theArray; // where the list items are
   private int eltcount;      // how many elements are in the array
   private int start;         // index of first element
   private int end;           // index of the last element
   private int capacity;      // number of slots in the array
```

originally, assumed start = 0

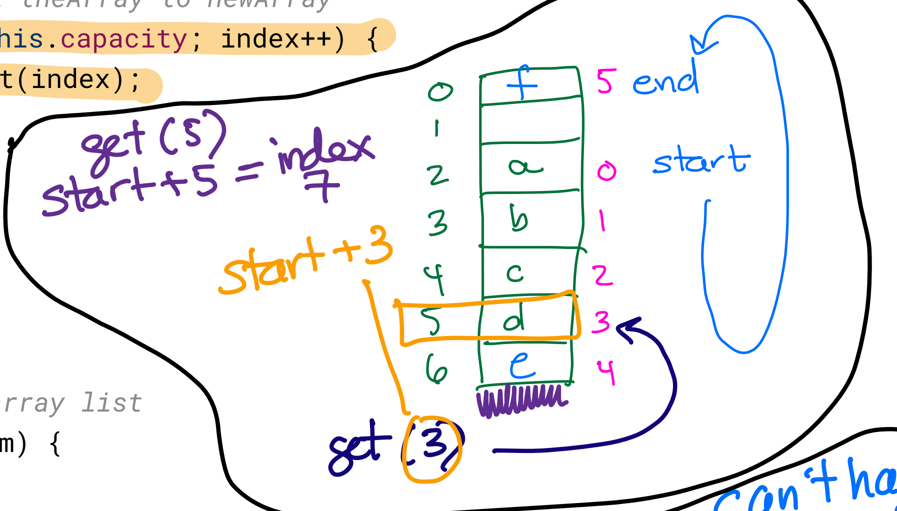
```
   // get the element at given index (0-based) from the ArrList
   public String get(int position) {
      if ((position >= 0) && (position < this.capacity)) {
         // return this.theArray[index];  // the original code
         return this.theArray[(position+start) % this.capacity];
      }
      throw new IllegalArgumentException("position " + position + " out of bounds");
   }
```

% capacity does wrap around

converting position in ArrList to array index

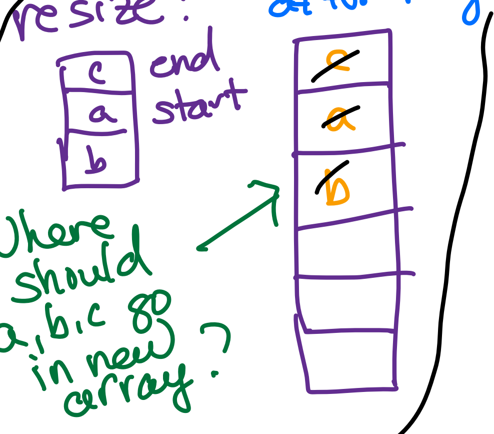
```
   // grow the array to the given size, copying over the existing elements.
   private void resize(int newSize) {
      String[] newArray = new String[newSize]; // make the new array
      // copy items from the current theArray to newArray
      for (int index = 0; index < this.capacity; index++) {
         newArray[index] = this.get(index);
      }
      this.theArray = newArray;
      this.start = 0;
      this.end = this.capacity - 1;
      this.capacity = newSize;
   }
```

set



```
   // add element to the end of the array list
   public void addLast(String newItem) {
      if (this.isFull()) {
         this.resize(this.capacity * 2); // add capacity to the array
         this.addLast(newItem);
      } else {
         if (!(this.isEmpty())) {
            this.end = (this.end + 1) % this.capacity;
         }
         this.eltcount = this.eltcount + 1;
         this.theArray[this.end] = newItem;
      }
   }
```

resize: can't have gaps in middle after resize



Where should a,b,c go in new array?

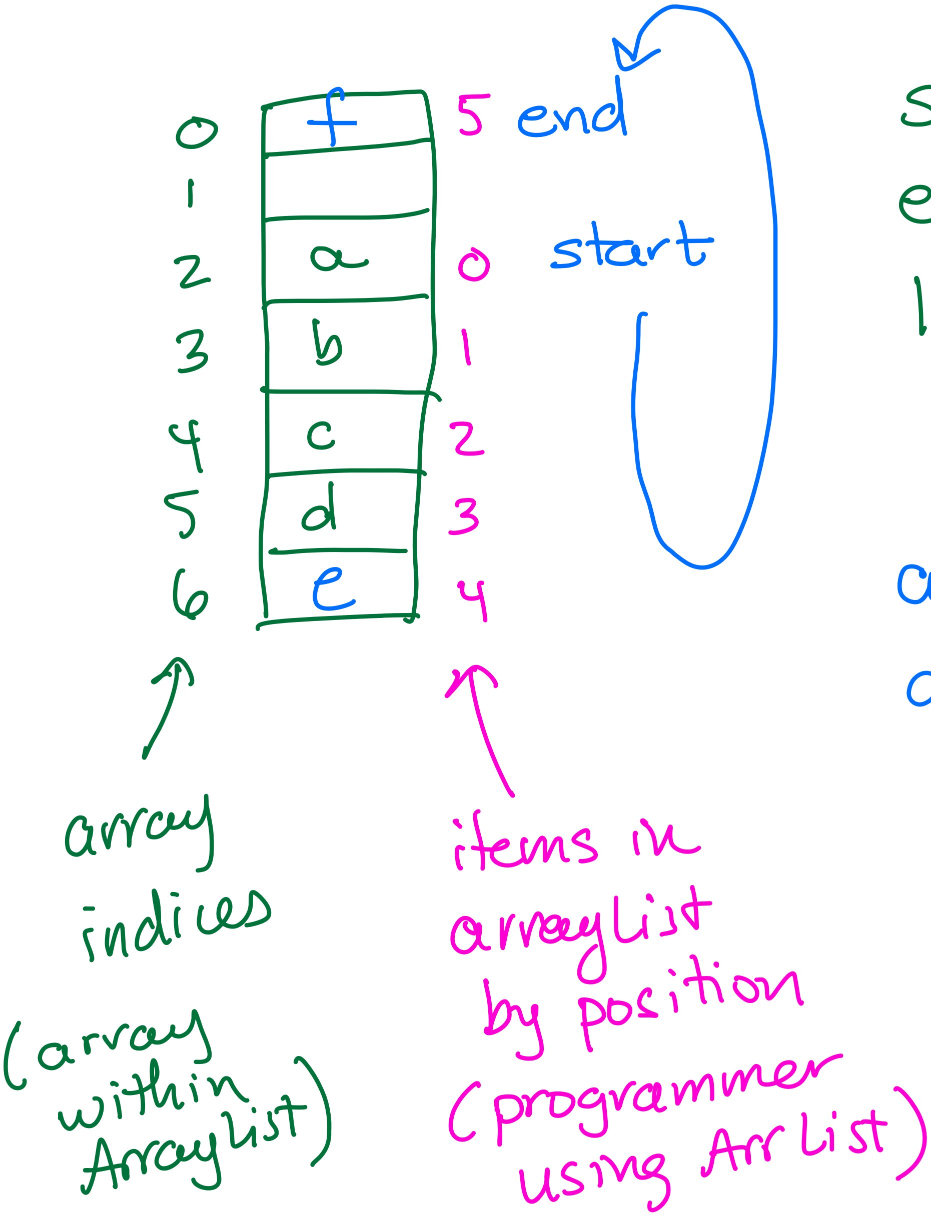
2 valid options (there are others)

keeps old
a
b
a
b
c
relocates start to

```
// add element to the end of the array list
public void addFirst(String newItem) {
    if (this.isFull()) {
        // add capacity to the array
        this.resize(this.capacity * 2);
        // now that the array has room, add the item
        this.addFirst(newItem);
    } else {
        if (!(this.isEmpty())) {
            this.start = ((this.start - 1) + capacity) % this.capacity;
        }
        this.eltcount = this.eltcount + 1;
        this.theArray[this.start] = newItem;
    }
}
```

we do this because Java's % does remainder, not full modulo. It gives the wrong answer on negative numbers (eg, when start=0). adding capacity retains the result of modulo capacity, while avoiding the negative number problem

# Cyclic Array



start = 2
end = 5
list contents are
a, b, c, d

addLast e
addLast f

```
class ArrList {
    String[] theArray;

}
```

```
class ... {
  AL = new ArrList(...)
```

**Activity: Three Design Exercises**

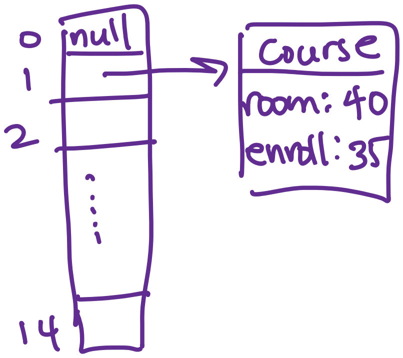
Propose data organizations for these

**Design problem #1**: A professor is trying to manage enrollments for several lab sections (numbered 01 through 14). For each lab, the professor needs to store the capacity of the room and the number of students in the lab. Propose specific data structures to organize this information.

**Design problem #2**: A department is trying to manage enrollments several courses (numbered 1000 through 1999). For each course, the department needs to store the capacity of the room and the number of students in the course. Propose specific data structures to organize this information.

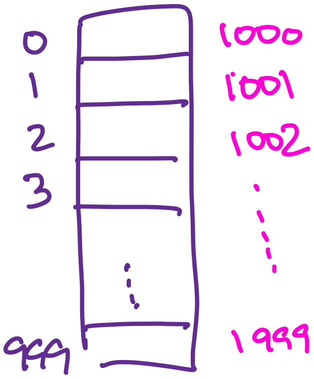
**Design problem #3**: A professor is trying to manage enrollments for multiple lab sections, each labeled with the day of week and start time (such as Mon 8-10, Tues 4-6, etc). For each lab, the professor needs to store the room where the lab is meeting.

#1



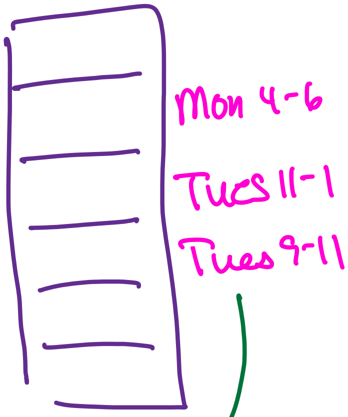
array (list) of objects

#2



infoForCourse(1562)
convert 1562 into array index 562

#3



if can convert strings to #s, can still use array

## Working with HashMaps (Java) or Dictionaries (Python)

```
// Map lab times to room numbers
HashMap<String, String> labRooms = new HashMap<String, String>();

// Associate this key with this value
labRooms.put("Mon 4-6", "CIT219");
labRooms.put("Tue 6-8", "CIT501");

labRooms.get("Mon 4-6"); // Returns "CIT219"

// Changes the value mapped to this key
labRooms.put("Mon 4-6", "CIT444");
labRooms.get("Mon 4-6");

labRooms.get("Wed 8-10");

if(labRooms.containsKey("Mon 4-6")) {
   // . . .
}
```

pink highlights are labels that I want to access my data.