CS62: Spring 2025 | Lecture #6 (ArrayLists) worksheet | Jingyi Li

```java
public class ArrayList<E> implements List<E> {
    private E[] data; // underlying array of Es
    private int size; // number of Es in arraylist.

    /**
     * Constructs an ArrayList with an initial capacity of 2.
     */
    public ArrayList() {
        data = (E[]) new Object[2];
        size = 0;
    }

    /**
     * Constructs an ArrayList with the specified capacity.
     */
    public ArrayList(int capacity) {
        data = (E[]) new Object[capacity];
        size = 0;
     }

     /**
      * TODO: Resizes the ArrayList's capacity to the specified capacity.
      */
      private void resize(int capacity) {
          //reserve a new temporary array of Es with the provided capacity


          //copy all elements from old array (data) to temp array




          //point data to the new temp array

      }

    /**
      * TODO: Inserts the element at the specified index. Shifts existing elements to the right and
      * doubles its capacity if necessary.
      *
      * @param index the index to insert the element
      * @param element the element to be inserted
      * @pre: 0 <= index <= size
      */
      public void add(int index, E element) {
          //check whether index is in range



          //if full double in size



          // shift elements to the right



          //increase number of elements


          //put the element at the right index in data

        }
```

```java
/**
 * TODO: Replaces the element at the specified index with the specified E.
 *
 * @param index the index of the element to replace
 * @param element element to be stored at specified index
 * @return the old element that was replaced
 * @pre: 0<=index< size
 */
 public E set(int index, E element) {
     //check if index is in range


     //retrieve old element at index


     //update index with new element


     //return old element


 }

/**
 * TODO: Removes and returns the element at the specified index.
 *
 * @param index the index of the element to be removed
 * @return the removed element
 * @pre: 0<=index<size
 */
 public E remove(int index) {
     //check if index is in range


     //retrieve element at index


     //reduce number of elements by 1


     //shift all elements from the index until the end left 1




     // set last element to null (since they've been shifted)


     // shrink to save space if necessary



     //return removed element


 }
```