Lecture 12 – ArrayLists, Part 3 (and debugging diagrams)

Summarize Worst-Case Runtimes (in terms of number of elements in the list)

	`LinkList`	`MutableList`	`ArrList`
`size`			
`addFirst`			
`addLast`			
`get(index)`			

What about addFirst?

ArrList theArray: locXXXX end: 2 eltcount: 3
“hello”
“there”
“brown”

```
public class ArrList {
   String[] theArray;  // the underlying array that stores the elements
   int eltcount;       // how many elements are in the array
   int end;            // the last USED slot in the array


   public ArrList(int initialSlots) {
       this.theArray = new String[initialSlots];
       this.eltcount = 0;
       this.end = 0;

   }

   // get an element by its position in the list
   public String get(int index) {

       if ((index >= 0) && (index < this.eltcount)) {
           return theArray[index];
       }
       throw new IllegalArgumentException("array index " + index + " out of
bounds");

   }

   private void resize(int newSize) {
       // make the new array
       String[] newArray = new String[newSize];
       // copy items from the current theArray to newArray
       for (int index = 0; index < theArray.length; index++) {
           newArray[index] = this.theArray[index];
       }
       // change this.theArray to refer to the new, larger array
       this.theArray = newArray;
   }

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

Tests for get

```
@Test
public void testGet() {
    ArrList L1 = new ArrList(5);
    L1.addLast("a");
    L1.addLast("b");
    L1.addLast("c");

    // L1: a, b, c
    assertEquals("b", L1.get(1));

    ArrList L2 = new ArrList(5);
    L2.addFirst("b");
    L2.addFirst("a");
    L2.addLast("c");
    // L2: a b c
    assertEquals("b", L2.get(1));

    ArrList L3 = new ArrList(5);
    L3.addLast("c");
    L3.addFirst("b");
    L3.addFirst("a");
    // L3: a b c
    assertEquals("b", L3.get(1));
}
```

Debugging – different cases `for addLast/addFirst`

ArrList theArray: locXXXX end: start: eltcount:	**ArrList** theArray: locXXXX end: start: eltcount:	**ArrList** theArray: locXXXX end: start: eltcount:

ArrList theArray: locXXXX end: start: eltcount:	**ArrList** theArray: locXXXX end: start: eltcount:	**ArrList** theArray: locXXXX end: start: eltcount:

Full ArrList code (from Friday)

```
public class ArrList {
   String[] theArray;  // the underlying array that stores the elements
   int eltcount;       // how many elements are in the array
   int end;            // the last USED slot in the array


   public ArrList(int initialSlots) {
       this.theArray = new String[initialSlots];
       this.eltcount = 0;
       this.end = 0;
   }

   // get an element by its position in the list
   public String get(int index) {

       if ((index >= 0) && (index < this.eltcount)) {
           return theArray[index];
       }
       throw new IllegalArgumentException("array index " + index + " out of bounds");

   }

   private void resize(int newSize) {
       // make the new array
       String[] newArray = new String[newSize];
       // copy items from the current theArray to newArray
       for (int index = 0; index < theArray.length; index++) {
           newArray[index] = this.theArray[index];
       }
       // change this.theArray to refer to the new, larger array
       this.theArray = newArray;
   }

   private boolean isFull() {
       return this.eltcount == this.theArray.length;
   }

   public boolean isEmpty() {
       return this.eltcount == 0;
   }

   public int size() {
       return this.eltcount;
   }

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