ArrayList implementation

implemented with a regular 1D array (internalArr) with initial size (e.g., 16).

when internalArr gets full:
  - new array (tmpInternalArr) created, with double the size (32)
  - elements from internalArr copied over to tmpInternalArr

when internalArr gets half empty:
  - new array tmpInternalArr with half the size created
  - elements from internalArr copied over to tmpInternalArr

```
[abc] => [0 1 ... 8]
```

internalArray          tmpInternalArr
### Running Time: ArrayList vs (sufficiently large) array:

#### Worst-case:

<table>
<thead>
<tr>
<th>Operation</th>
<th>ArrayList</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>remove (from front)</td>
<td>$O(n)$</td>
<td>$O(1)$ (set to -1, null)</td>
</tr>
<tr>
<td>get</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>indexOf</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>contains</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>size()</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

Maintain variable `size` to keep track of number of elements:
- if add: do `size++`
- remove: `size--`

ArrayLists also waste space!

Suppose internal array reaches capacity:

| 0 | 1 | 2 | 3 | 14 |

For next insert, internal array doubled:

| 0 | 1 | 2 | 3 | 14 | 15 |

If no more inserts:

| 0 | 1 | 2 | 3 | 15 | 16 | 31 |

Wasted space: (Objects vs. primitive types)
Problem with arrays + ArrayLists?
- elements need to be shifted for add(), remove()

Reason: Both are stored adjacently in memory.

Ideas to make insertions + deletions not require shift:

Linked list - series of nodes, not always adjacent in memory.

LinkedList <String> list = new LinkedList<String>();

Requires import java.util.LinkedList;
Node class: (already built into Java!)

```java
public class Node {
    Object element;
    Node next;
}
```

Something else?

```java
public Node (Object e, Node n) {
    element = e;
    next = n;
}
```

```java
public Node (Object e) {
    element = e;
}
```

What other info should we keep about the list?

Where it starts/ends:

```java
public class LinkedList {
    Node first;
    Node last;
    int size;
}
```
Linked List operations

Assume some elements are already in the list (so don't have to check if list is empty)

Assume LinkedList of Strings (applies to any)

PseudoCode

```
["a"]  →  ["b"] →  null →  first
↑ (next)  ↑  last

1. addToEnd (Object e)

Create new Node: Node newNode = new Node ("new")
Add to end: last.next = newNode
```

```
["a"]  →  ["b"]  →  ["new"] →  null →  first
↑ (next)  ↑  last
```

Time: O(1)

2. remove (Later)
(3) contains (Object e) (Scan and most n nodes)

Node current = first
while (current != null)
    if (current.element == e) // pseudocode for equals
        return true;
    current = current.next

return false

Time: O(n)

(4) indexof (Object e) (Scan and keep counter)

count = 0
Node current = first
while (current.element != null)
    if (current.element == e)
        return count;
    else
        current = current.next
        count++

return null

Time: O(n)