



## The Class ArrayLinearList



- General purpose implementation of linear lists.
- Unknown number of lists.

## Create An Empty List

```
ArrayLinearList a = new ArrayLinearList(100),  
    b = new ArrayLinearList(),  
    c;  
LinearList d = new ArrayLinearList(1000),  
    e = new ArrayLinearList(),  
    f;
```

## Using A Linear List

```
System.out.println(a.size());  
a.add(0, new Integer(2));  
b.add(0, new Integer(4));  
System.out.println(a);  
b.remove(0);  
if (a.isEmpty())  
    a.add(0, new Integer(5));
```

## Array Of Linear Lists

```
LinearList [] x = new LinearList [4];  
x[0] = new ArrayLinearList(20);  
x[1] = new Chain();  
x[2] = new Chain();  
x[3] = new ArrayLinearList();  
for (int i = 0; i < 4; i++)  
    x[i].add(0, new Integer(i));
```

## The Class ArrayLinearList

```
/** array implementation of LinearList */
package dataStructures;
import java.util.*; // has Iterator interface
import utilities.*; // has array resizing class

public class ArrayLinearList implements LinearList
{
    // data members
    protected Object [] element; // array of elements
    protected int size; // number of elements in array
    // constructors and other methods come here
}
```

## A Constructor



```
/** create a list with initial capacity initialCapacity
 * @throws IllegalArgumentException when
 * initialCapacity < 1 */
public ArrayLinearList(int initialCapacity)
{
    if (initialCapacity < 1)
        throw new IllegalArgumentException
            ("initialCapacity must be >= 1");
    // size has the default initial value of 0
    element = new Object [initialCapacity];
}
```

## Another Constructor



```
/** create a list with initial capacity 10 */
public ArrayLinearList()
{ // use default capacity of 10
    this(10);
}
```

## The Method isEmpty



```
/** @return true iff list is empty */
public boolean isEmpty()
{ return size == 0; }
```

### The Method size()

```
/** @return current number of elements in list */  
public int size()  
{return size;}
```

### The Method checkIndex

```
/** @throws IndexOutOfBoundsException when  
 * index is not between 0 and size - 1 */  
void checkIndex(int index)  
{  
    if (index < 0 || index >= size)  
        throw new IndexOutOfBoundsException  
            ("index = " + index + " size = " + size);  
}
```

### The Method get

```
/** @return element with specified index  
 * @throws IndexOutOfBoundsException when  
 * index is not between 0 and size - 1 */  
public Object get(int index)  
{  
    checkIndex(index);  
    return element[index];  
}
```

### The Method indexOf

```
/** @return index of first occurrence of theElement,  
 * return -1 if theElement not in list */  
public int indexOf(Object theElement)  
{  
    // search element[] for theElement  
    for (int i = 0; i < size; i++)  
        if (element[i].equals(theElement))  
            return i;  
  
    // theElement not found  
    return -1;  
}
```

### The Method remove

```
public Object remove(int index)
{
    checkIndex(index);

    // valid index, shift elements with higher index
    Object removedElement = element[index];
    for (int i = index + 1; i < size; i++)
        element[i-1] = element[i];

    element[--size] = null; // enable garbage collection
    return removedElement;
}
```

### The Method add

```
public void add(int index, Object theElement)
{
    if (index < 0 || index > size)
        // invalid list position
        throw new IndexOutOfBoundsException
            ("index = " + index + " size = " + size);

    // valid index, make sure we have space
    if (size == element.length)
        // no space, double capacity
        element = ChangeArrayLength.changeLength1D(element, 2 *
            size);
}
```

### The Method add

```
// shift elements right one position
for (int i = size - 1; i >= index; i--)
    element[i + 1] = element[i];

element[index] = theElement;

size++;
}
```

### Faster Way To Shift Elements 1 Right

```
System.arraycopy(element, index, element,
    index + 1, size - index);
```

### Convert To A String

```
public String toString()
{
    StringBuffer s = new StringBuffer("");
    // put elements into the buffer
    for (int i = 0; i < size; i++)
        if (element[i] == null) s.append("null, ");
        else s.append(element[i].toString() + ", ");
    if (size > 0) s.delete(s.length() - 2, s.length());
    // remove last ", "
    s.append("");
    // create equivalent String
    return new String(s);
}
```