Data Representation Methods
array --- Chapter 5
linked --- Chapter 6
simulated pointer --- Chapter 7

Linear List Array Representation
use a one-dimensional array `element[]`

\[ L = (a, b, c, d, e) \]
Store element \( i \) of list in `element[i]`.

Right To Left Mapping

Mapping That Skips Every Other Position
Wrap Around Mapping

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
</table>

| d | e |

Representation Used In Text

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>size = 5</th>
</tr>
</thead>
</table>

put element i of list in element[i]

use a variable size to record current number of elements

Add/Remove An Element

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
</table>

| a | b | c | d | e |

Data Type Of Array element[]

Data type of list elements is unknown.

Define element[] to be of data type Object.

Cannot put elements of primitive data types (int, float, double, char, etc.) into our linear lists.
**Length of Array element[]**

Don’t know how many elements will be in list.

Must pick an initial length and dynamically increase as needed.

**Increasing Array Length**

Length of array `element[]` is 6.

```
[ a b c d e f ]
```

First create a new and larger array

```
newArray = new Object[15];
```

Finally, rename new array.

```
element = newArray;
element[0]
```

```
[ a b c d e f ]
element.length = 15
```
Altogether Now

// create a new array of proper length and data type
Object [] newArray = new Object [newLength];

// copy all elements from old array into new one
System.arraycopy(element, 0, newArray, 0, element.length);

// rename array
element = newArray;

public static Object [] changeLength(Object [] a, int newLength)
{
    Object [] newArray = new Object [newLength];
    System.arraycopy(a, 0, newArray, 0, a.length);
    return newArray;
}

Integer [] a = new Integer [10];
...
    a = (Integer []) changeLength(a, 100); // erroneous

How Big Should The New Array Be?

At least 1 more than current array length.

Cost of increasing array length is
Theta(new length)

Cost of n add operations done on an initially empty linear list increases by
Theta(n^2)

Space Complexity

element[6]
    a b c d e f

newArray = new char[7];

    abcdedef

space needed = 2 * newLength – 1
            = 2 * maxListSize – 1
Array Doubling

Double the array length.

\[ \text{newArray} = \text{new char}[12]; \]

Time for \( n \) adds goes up by \( \Theta(n) \).
Space needed = \( 1.5 \times \text{newLength} \).
Space needed \( \leq 3 \times \text{maxListSize} - 3 \)

How Big Should The New Array Be?

Resizing by any constant factor
\[ \text{new length} = c \times \text{old length} \]
increases the cost of \( n \) adds by \( \Theta(n) \).

Resizing by an additive constant increases the cost of \( n \) add operations by \( \Theta(n^2) \).

What Does Java Do?

java.util.Vector … array doubling

java.util.ArrayList … \( c = 1.5 \)

dataStructures.ArrayLinearList of text … \( c = 2 \)