Lists

Chapter 12

Contents

- Specifications for the ADT List
- Using the ADT List
- Java Class Library: The Interface **`List`**
- Java Class Library: The Class **`ArrayList`**

Objectives

- Describe the ADT list
- Use the ADT list in a Java program

Lists

- A collection
  - Has order … which may or may not matter
  - Additions may come anywhere in list

Figure 12-1 A to-do list

Lists

- Typical actions with lists
  - Add item at end (although can add anywhere)
  - Remove an item (or all items)
  - Replace an item
  - Look at an item (or all items)
  - Search *for* an entry
  - Count how many items in the list
  - Check if list is empty

ADT List

- Data
  - A collection of objects in a specific order and having the same data type
  - The number of objects in the collection
- Operations
  - add(newEntry)
  - add(newPosition, newEntry)
  - remove(givenPosition)

    …

ADT List

- Operations (ctd.)
  - clear()
  - replace(givenPosition, newEntry)
  - getEntry(givenPosition)
  - contains(anEntry)
  - getLength()
  - isEmpty()
  - toArray()

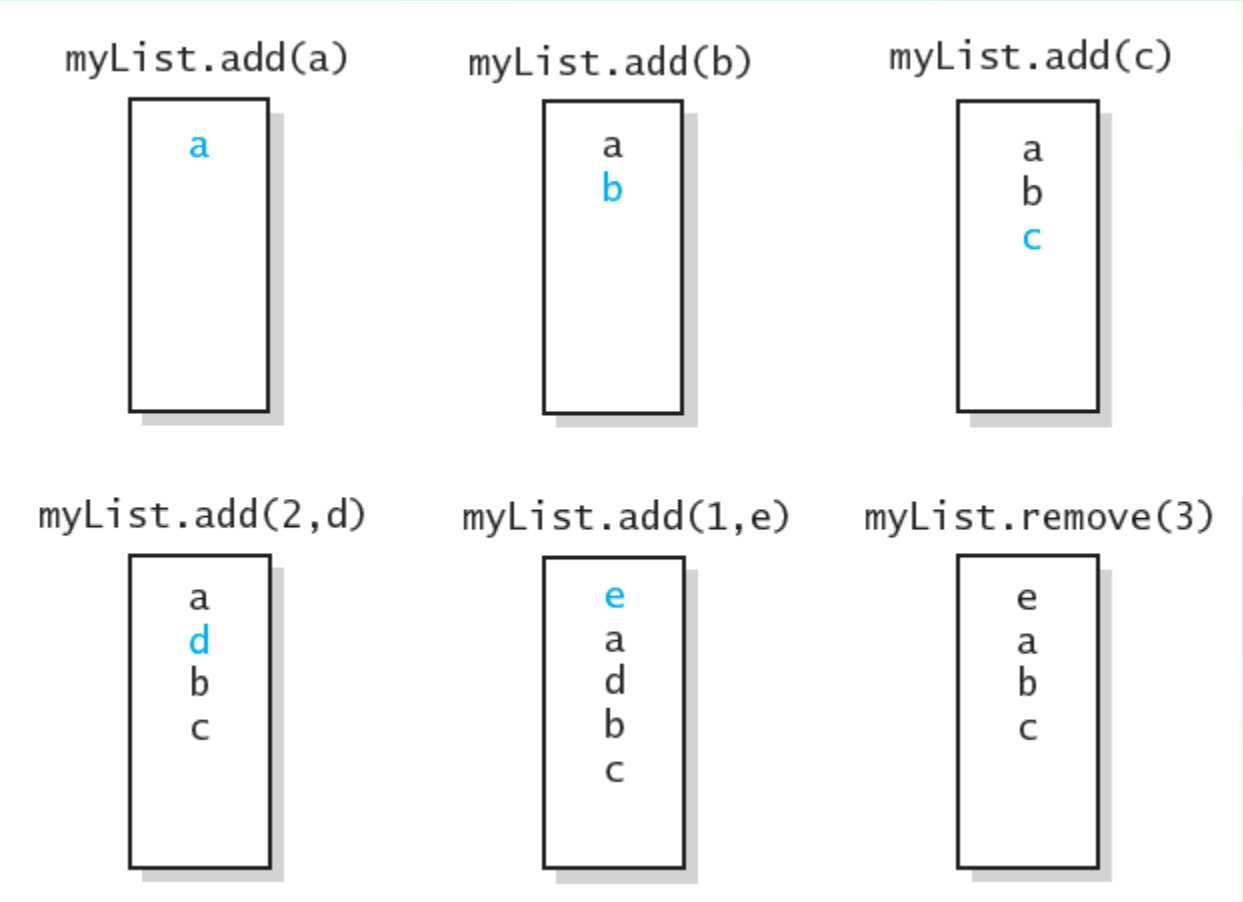

Figure 12-2 The effect of ADT list operations on an initially empty list

Question 1 Write pseudocode statements that add some objects to a list, as follows. First add c, then a, then b, and then d, such that the order of the objects in the list will be a, b, c, d.

Question 2 Write pseudocode statements that exchange the third and seventh entries in a list of 10 objects.

Question 1 Write pseudocode statements that add some objects to a list, as follows. First add c, then a, then b, and then d, such that the order of the objects in the list will be a, b, c, d.

```
myList.add(c)
myList.add(1, a)
myList.add(2, b)
myList.add(4, d)
```

Question 2 Write pseudocode statements that exchange the third and seventh entries in a list of 10 objects.

```
seven = myList.remove(7)
three = myList.remove(3)
myList.add(3, seven)
myList.add(7, three)
```

Another solution:

```
seven = myList.getEntry(7)
three = myList.getEntry(3)
myList.replace(3, seven)
myList.replace(7, three)
```

List

- View list interface, Listing 12-1
- Using the ADT List
  - Don't need to know *ho*
  - Only need to know *wha*

on

Note: Code listing files must be in same folder as PowerPoint files for links to work

- Consider keeping list of finishers of a running race
  - View client code, Listing 12-2
  - Output

Figure 12-3 A list of numbers that identify runners in the order in which they finished a race

Question 3 In the previous example, what changes to testList are necessary to represent the runner's numbers as Integer objects instead of strings?

```
ListInterface<String> runnerList = new AList<String>();
// runnerList has only methods in ListInterface
runnerList.add("16"); // winner
runnerList.add(" 4"); // second place
runnerList.add("33"); // third place
runnerList.add("27"); // fourth place
displayList(runnerList);
```

Question 3 In the previous example, what changes to testList are necessary to represent the runner's numbers as Integer objects instead of strings?

```
ListInterface<String> runnerList = new AList<String>();
// runnerList has only methods in ListInterface
runnerList.add("16"); // winner
runnerList.add(" 4"); // second place
runnerList.add("33"); // third place
runnerList.add("27"); // fourth place
displayList(runnerList);
```

```
ListInterface<Integer> rList = new AList<Integer>();
rList.add(16);
rList.add(4);
rList.add(33);
rList.add(27);
rList.displayList();
```

Java Class Library: The Interface `List`

- Method headers
  - `public T remove(int index)`
  - `public void clear()`
  - `public boolean isEmpty()`
  - `public boolean add(T newEntry)`
  - `public void add`
    `(int index, T newEntry)`
    `...`

Java Class Library: The Interface `List`

- Method headers (ctd.)
  - `public T set(int index, T anEntry)`
    `// like replace`
  - `public T get(int index)`
    `// like getEntry`
  - `public boolean contains`
    `(Object anEntry)`
  - `public int size()`
    `// like getLength`

Java Class Library: The Interface `ArrayList`

- Implementation of ADT list with resizable array
  - Implements java.util.list
- Constructors available
  - `public ArrayList()`
  - `public ArrayList (int initialCapacity)`

End

Chapter 12

List Implementations that Use Arrays

Chapter 13

Contents

- Using an Array to Implement the ADT List
  - An Analogy
  - The Java Implementation
  - The Efficiency of Using an Array to Implement the ADT List
- Using a Vector to Implement the ADT List

Objectives

- Implement ADT list by using either array that you can resize or instance of **`Vector`**
- Discuss advantages, disadvantages of implementations presented

Alternatives

- Use an array
  - When all space used, must move data to larger array
- Use Java class **`Vector`**
  - Like an array that can expand automatically
- Chain of linked nodes
  - Insertion/deletion anywhere is harder

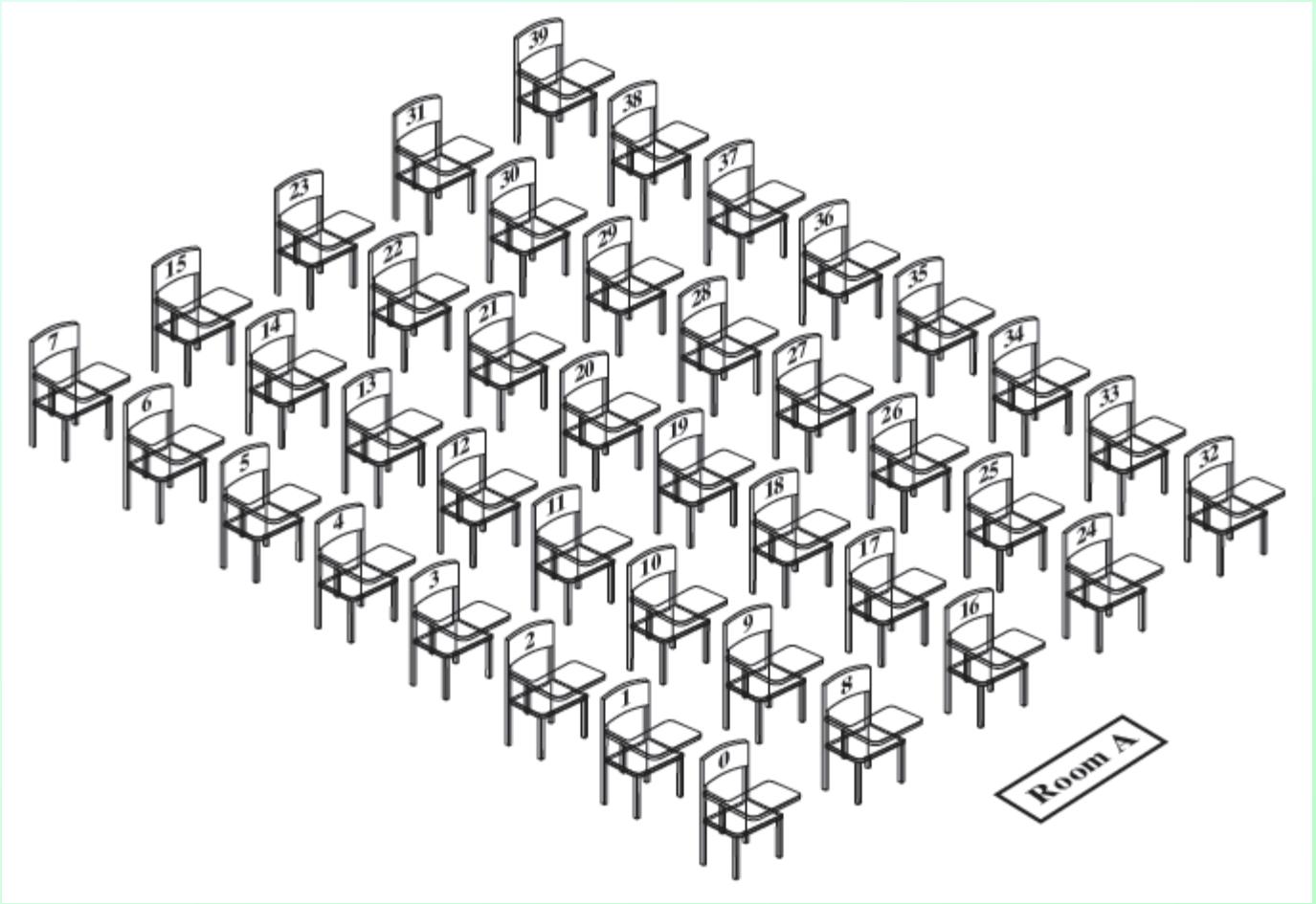

Figure 13-1 A classroom that contains desks in fixed positions

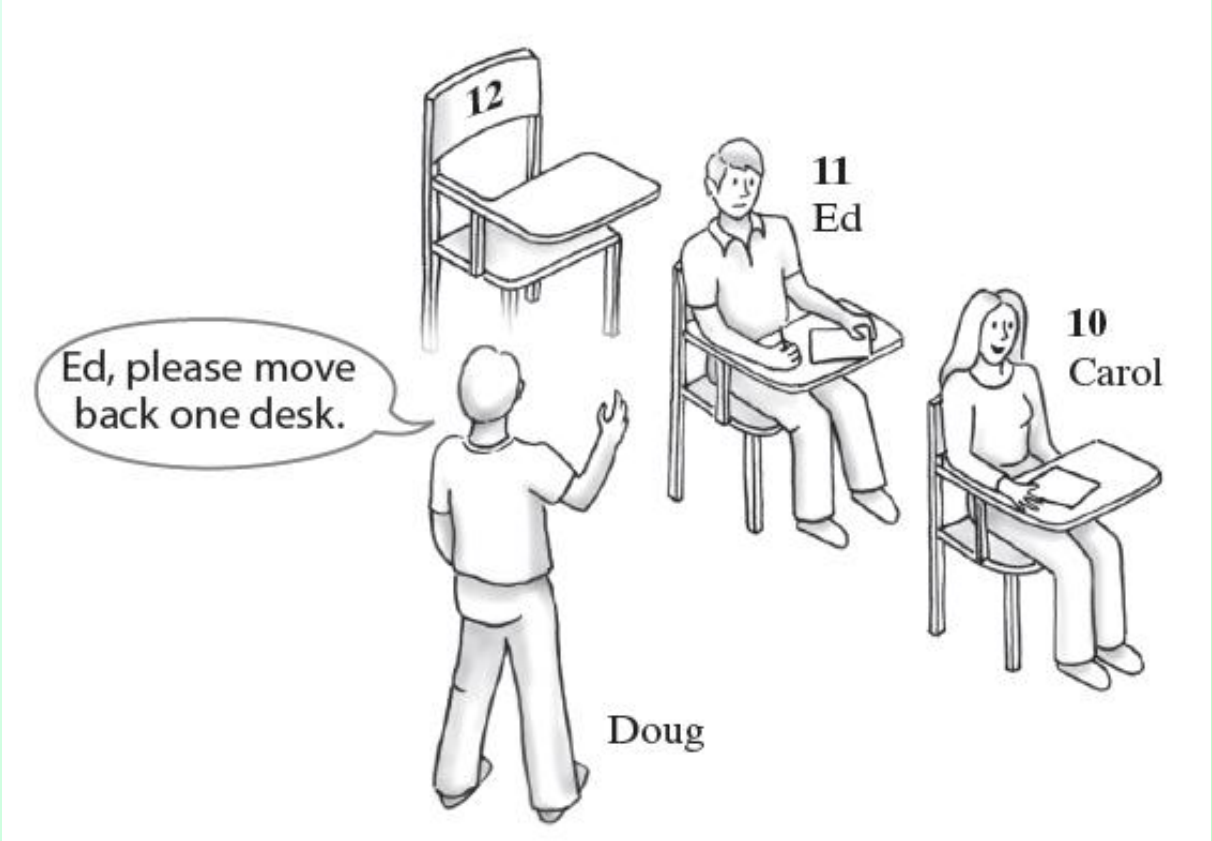

Figure 13-2 Seating a new student between two existing students: At least one other student must move

Question 1 In the previous example, under what circumstance could you add a new student alphabetically by name without moving any other student?

When the name comes after the name of the student in the last occupied desk; the new student then sits at the desk after the last one that is currently occupied.

The Java Implementation

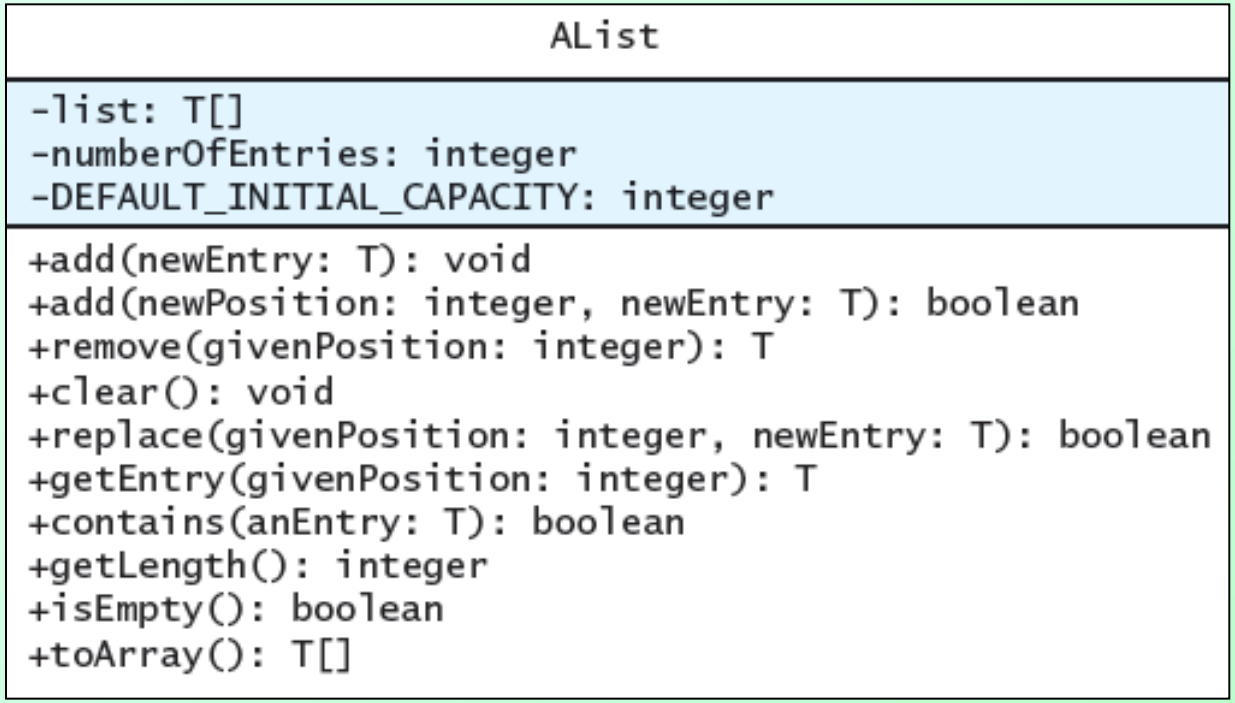

Figure 13-3 UML notation for the class **`AList`**

The Java Implementation

- Note **`Alist`** code, Listing 13-1

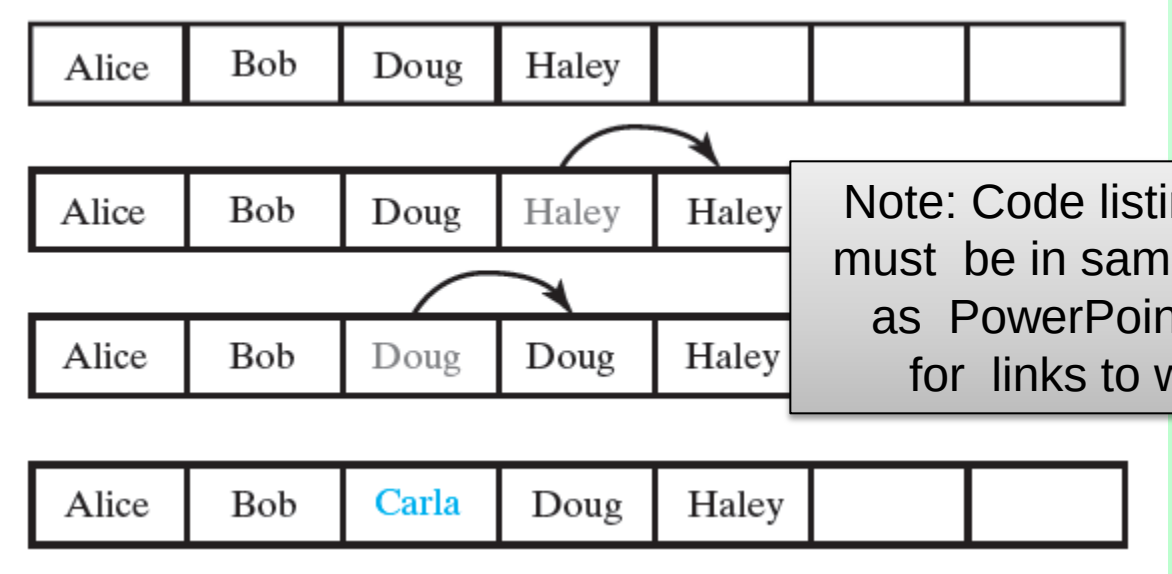

Figure 13-4 Making room to insert Carla as the third entry in an array

Question 4 You could implement the first add method, which adds an entry to the end of the list, by invoking the second add method, as follows:

```
public void add(T newEntry)
{   add(numberOfEntries + 1, newEntry);
}
```

Discuss the pros and cons of this revised approach.

Advantage: It is easier to implement this add method. Your code will more likely be correct if the other add method is correct.

Disadvantage: Invoking another method uses more execution time. Additionally, the second add method invokes makeRoom needlessly.

Question 5 Suppose that myList is a list that contains the five entries a b c d e.

a. What does myList contain after executing myList.add(5, w) ?

a b c d w e

b. Starting with the original five entries, what does myList contain after executing myList.add(6, w) ?

a b c d e w

c. Which of the operations in Parts a and b of this question require entries in the array to shift?

The operation in Part a

Question 6 If myList is a list of five entries, each of the following statements adds a new entry to the end of the list:

```
myList.add(newEntry);
myList.add(6, newEntry);
```

Which way requires fewer operations?

myList.add(newEntry). The other add method validates the position 6 and then needlessly invokes makeRoom.

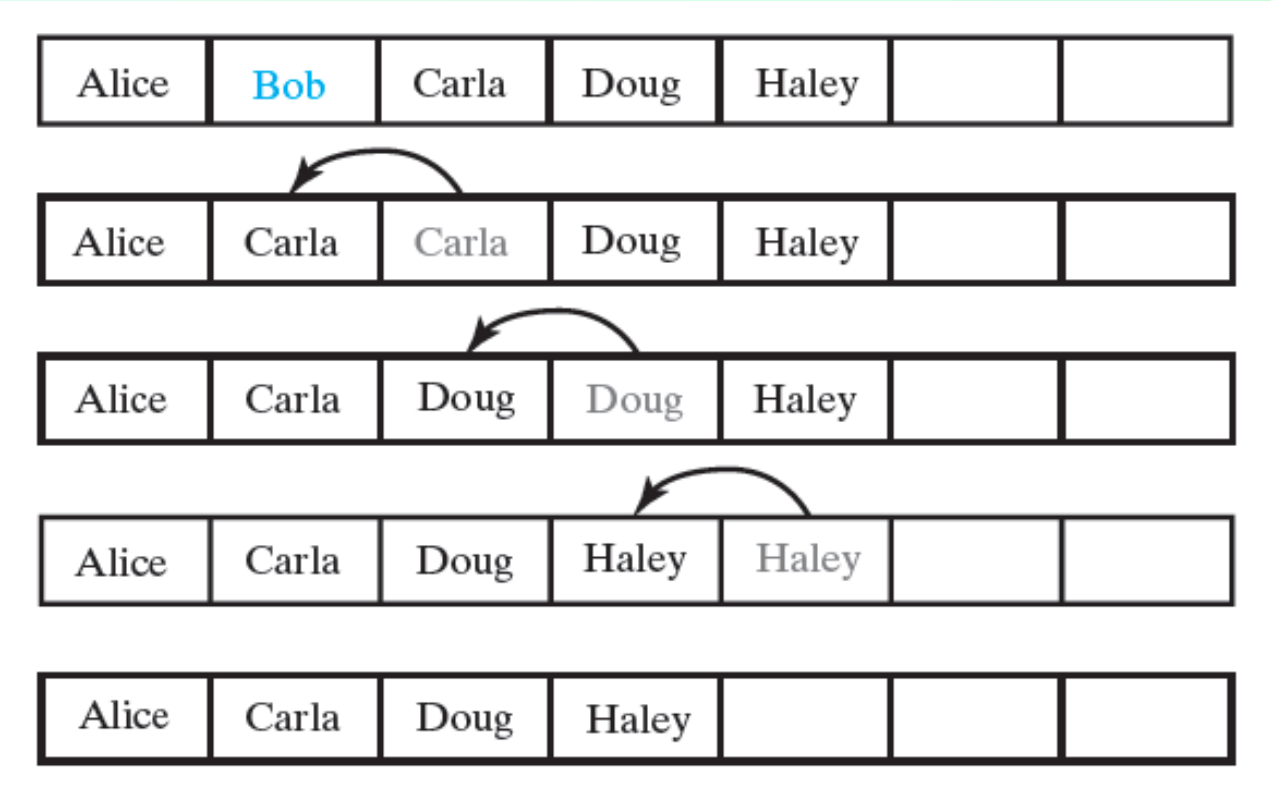

Figure 13-5 Removing Bob by shifting array entries

Using a Vector to Implement the ADT List

- View class **`VectorList`**, Listing 13-A
- Note
  - Example of an adaptor class
  - Writing code for the class simple
  - Execution may be slow due to background invocation of **`Vector`** methods
  - Adding at end of list, retrieving specific entry are fast
  - Adding, removing in middle of list slower

End

Chapter 13