Advanced Programming

List Implementations That Use Arrays
Chapter Contents

- Using a Fixed-Size Array to Implement the ADT List
  - An Analogy
  - The Java Implementation
- Using Dynamic Array Expansion to Implement the ADT List
  - Expanding an Array
  - A News Implementation of a List
- Using a Vector to Implement the ADT List
  - A Summary of Methods in the Class Vector
- The Pros and Cons of Using an Array to Implement the ADT List
  - The Class `ArrayList`
  - The `Interface Serializable`
An Analogy

- Consider a classroom with 40 desks in fixed position
  - Desks are wasted if less than 40 students
  - Not enough desks if more than 40 students

- An array is like the classroom
  - Each desk an array location
An Analogy

Fig. 5-1 A classroom that contains desks in a fixed position.
An Analogy

- Suppose we have some students in classroom in order alphabetically
- We add a new student
  - We desire to maintain the alphabetic order
  - We must shift some students
- We remove a student in the middle of the sequence
  - Again, we must shift some students
Adding a Student

Fig. 5-2 Seating a new student between two existing students: at least one other student must move
The Java Implementation

- Private data fields for implementation of AList
  - Implements interface ListInterface of Chapter 4

```java
private Object entry[]; // array of list entries
private int length; // current number of entries in list
private static final int MAX_SIZE = 50; // max length of list
```

- Note the full specification, pgs 103-105
**AList add() Methods**

- First `add` method adds a new item at the end of the list
  - Assign new value at end
  - Increment `length` of list
- Second `add` method adds item in mid-list
  - Requires a utility method, `makeRoom()`
  - This shifts elements ahead
Adding Items in Mid-list

Fig. 5-3 Making room to insert Carla as third entry in an array.
The `remove()` Method

- Must shift existing entries to avoid gap in the array
  - Except when removing last entry
- Method must also handle error situation
  - When position specified in the remove is invalid
  - When `remove()` is called and the list is empty
  - Invalid call returns null value
Removing a List Entry

Fig. 5-4 Removing Bob by shifting array entries.
Dynamic Array Expansion

- An array has a fixed size
  - If we need a larger list, we are in trouble
- When array becomes full
  - Move its contents to a larger array (dynamic expansion)
  - Copy data from original to new location
  - Manipulate names so new location keeps name of original array
Dynamic Array Expansion

Fig. 5-5 The dynamic expansion of an array copies the array's contents to a larger second array.
Dynamic Array Expansion

Fig. 5-6  (a) an array;
(b) the same array with two references;
(c) the two arrays, reference to original array now
referencing a new, larger array
A New Implementation of a List

- Change the **isFull** to always return false
  - We will expand the array when it becomes full
  - We keep this function so that the original interface does not change
- The **add()** methods will double the size of the array when it becomes full
- Now declare a private method **isArrayFull**
  - Called by the **add()** methods
Using a Vector to Implement the ADT List

- Java's **vector** class provides capabilities of an array
  - Able to expand dynamically
  - Hides the details of the process

- Vector
  - Found in package **java.util**
  - Has methods for manipulating entries
  - Enables implementing the ADT List
Using a Vector

Fig. 5-7 A client uses the methods given in ListInterface, but the implementation of the list uses Vector methods to perform its operations.
Using a Vector

Elements of the class

```java
import java.util.Vector;
public class VectorList implements ListInterface {
    private Vector entry; // entries in list
}
```

- Class `Vector` comes from package `java.util`
- Data field `entry` is an instance of a `Vector`
Using a Vector

- The \texttt{add()} methods
  - The first uses the \texttt{addElement} method from the Vector class
  - The other uses the \texttt{insertElementAt} method
- The \texttt{remove()} method
  - Uses the \texttt{removeElementAt} method
Pros and Cons of Array Use for the ADT List

- When using an array or vector …
  - Retrieving an entry is fast
  - Adding an entry at the end of the list is fast
  - Adding or removing an entry that is between other entries requires shifting elements in the array
  - Increasing the size of the array or vector requires copying elements
Java Class Library

- Has a class similar to class `AList` defined in this chapter
  - Class `ArrayList`
  - Uses dynamic array expansion
- Interface `Serializable`
  - Represent an object as a sequence of bytes to be written to a file
  - Add the words `implements Serializable` to class definition

```java
public class AList implements ListInterface, Serializable {
    ...}
```
Material Covered

- Chapter 5