

Lists

Chapter 8

Specifying the ADT List

- Things you make lists of
 - Chores
 - Addresses
 - Groceries
- Lists contain items of the same type
- Operations
 - Count items
 - Add, remove items
 - Retrieve

Specifying the ADT List

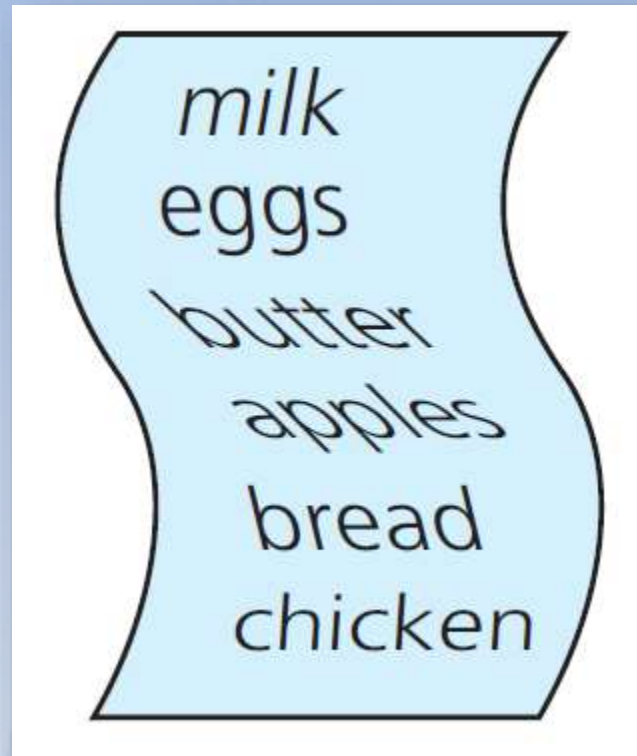


Figure 8-1 A grocery list

Specifying the ADT List

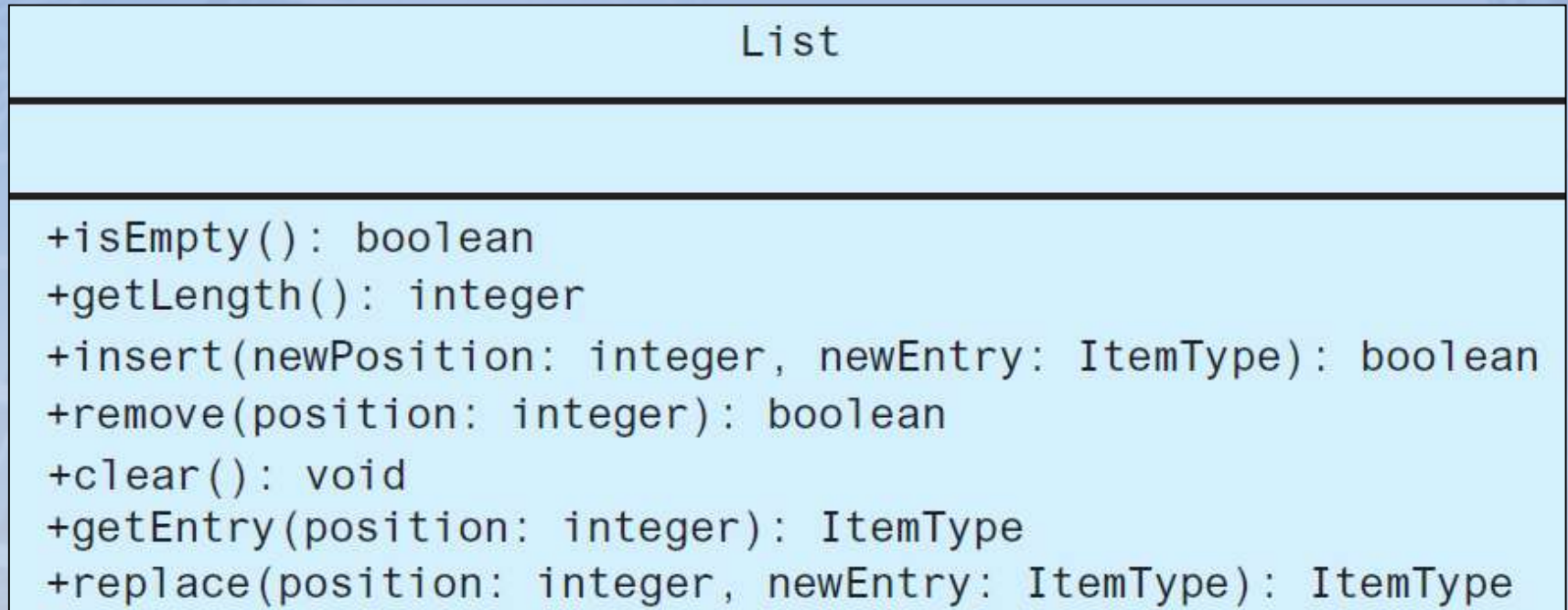


FIGURE 8-2 UML diagram for the ADT list

Specifying the ADT List

- Definition: ADT List
 - Finite number of objects
 - Not necessarily distinct
 - Same data type
 - Ordered by position as determined by client

Axioms for ADT List

1. `(List()).isEmpty() = true`
2. `(List()).getLength() = 0`
3. `aList.getLength() = (aList.insert(i, item)).getLength() - 1`
4. `aList.getLength() = (aList.remove(i)).getLength() + 1`
5. `(aList.insert(i, item)).isEmpty() = false`
6. `(List()).remove(i) = false`
7. `(aList.insert(i, item)).remove(i) = true`
8. `(aList.insert(i, item)).remove(i) = aList`
9. `(List()).getEntry(i) => error`
10. `(aList.insert(i, item)).getEntry(i) = item`
11. `aList.getEntry(i) = (aList.insert(i, item)).getEntry(i + 1)`
12. `aList.getEntry(i + 1) = (aList.remove(i)).getEntry(i)`
13. `(List()).replace(i, item) => error`
14. `(aList.replace(i, item)).getEntry(i) = item`

Using the List Operations

```
// Displays the items on the list aList.  
displayList(aList)  
{  
    for (position = 1 through aList.getLength())  
    {  
        dataItem = aList.getEntry(position)  
        Display dataItem  
    }  
}
```

Displaying the items on a list.

Using the List Operations

```
// Replaces the ith entry in the list aList with newEntry.  
// Returns true if the replacement was successful; otherwise return false.  
replace(aList, i, newEntry)  
{  
    success = aList.remove(i)  
    if (success)  
        success = aList.insert(i, newEntry)  
  
    return success  
}
```

Replacing an item.

Interface Template for ADT List

```
1  /** Interface for the ADT list
2   * @file ListInterface.h */
3
4  #ifndef LIST_INTERFACE_
5  #define LIST_INTERFACE_
6
7  template<class ItemType>
8  class ListInterface
9
10 {
11 public:
12     /** Sees whether this list is empty.
13      * @return True if the list is empty; otherwise returns false. */
14     virtual bool isEmpty() const = 0;
15
16     /** Gets the current number of entries in this list.
17      * @return The integer number of entries currently in the list. */
18     virtual int getLength() const = 0;
```

LISTING 8-1 A C++ interface for lists

Interface Template for ADT List

```
19
20     /** Inserts an entry into this list at a given position.
21     @pre  None.
22     @post If 1 <= position <= getLength() + 1 and the insertion is
23           successful, newEntry is at the given position in the list,
24           other entries are renumbered accordingly, and the returned
25           value is true.
26     @param newPosition The list position at which to insert newEntry.
27     @param newEntry The entry to insert into the list.
28     @return True if the insertion is successful, or false if not. */
29     virtual bool insert(int newPosition, const ItemType& newEntry) = 0;
30
31     /** Removes the entry at a given position from this list.
32     @pre  None.
33     @post If 1 <= position <= getLength() and the removal is successful,
34           the entry at the given position in the list is removed, other
35           items are renumbered accordingly, and the returned value is true.
36     @param position The list position of the entry to remove.
37     @return True if the removal is successful, or false if not. */
38     virtual bool remove(int position) = 0;
39
```

LISTING 8-1 A C++ interface for lists

Interface Template for ADT List

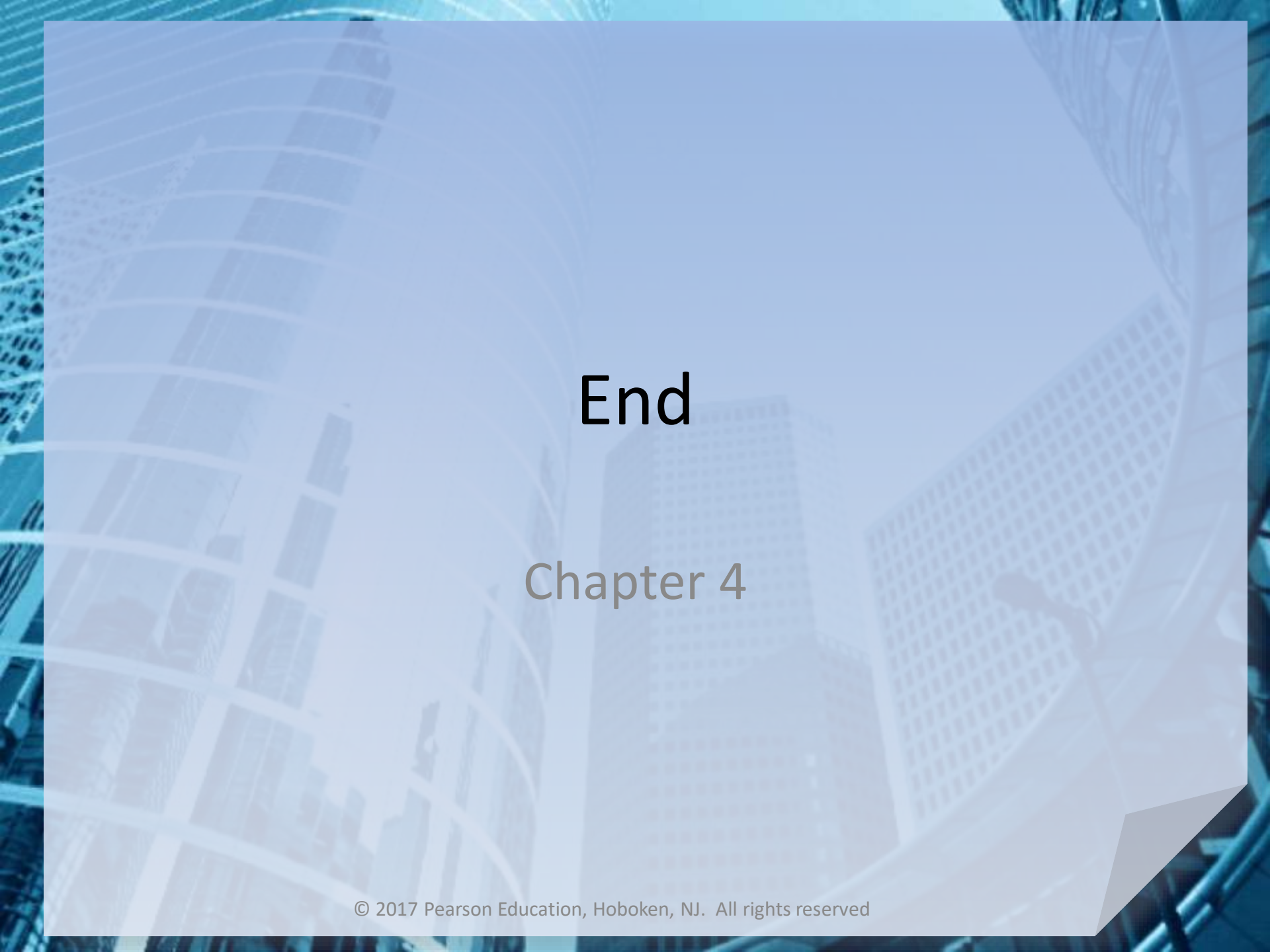
```
39
40     /** Removes all entries from this list.
41         @post  The list contains no entries and the count of items is 0. */
42     virtual void clear() = 0;
43
44     /** Gets the entry at the given position in this list.
45         @pre   1 <= position <= getLength().
46         @post  The desired entry has been returned.
47         @param position  The list position of the desired entry.
48         @return The entry at the given position. */
49     virtual ItemType getEntry(int position) const = 0;
50
```

LISTING 8-1 A C++ interface for lists

Interface Template for ADT List

```
50
51     /** Replaces the entry at the given position in this list.
52         @pre  1 <= position <= getLength().
53         @post  The entry at the given position is newEntry.
54         @param position  The list position of the entry to replace.
55         @param newEntry  The replacement entry.
56         @return  The replaced entry. */
57     virtual ItemType replace(int position, const ItemType& newEntry) = 0;
58
59     /** Destroys this list and frees its assigned memory. */
60     virtual ~ListInterface() { }
61 }; // end ListInterface
62 #endif
```

LISTING 8-1 A C++ interface for lists



End

Chapter 8