

CSIS 3103

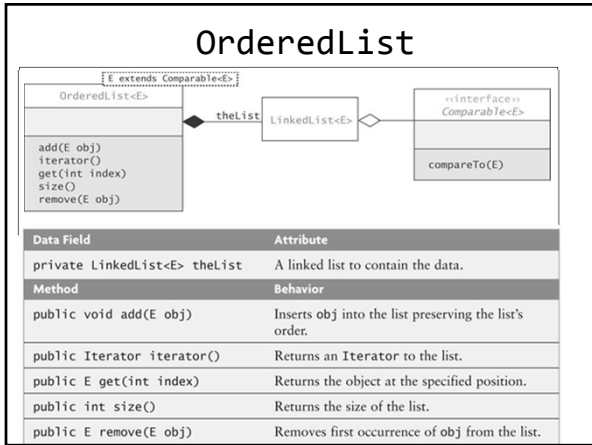
Ch 2: Linked List Applications

Ordered Lists

We want to maintain a list of items in ascending order at all times

Approach

- Develop an `OrderedList` class which implements the `Comparable`
- Use a `LinkedList` class as a component of the `OrderedList`
- If `OrderedList` *extended* `LinkedList`, it would expose `LinkedList`'s add methods to add an element out of order



Inserting into an OrderedList

Strategy for inserting new element `e`:

Find first item `> e`

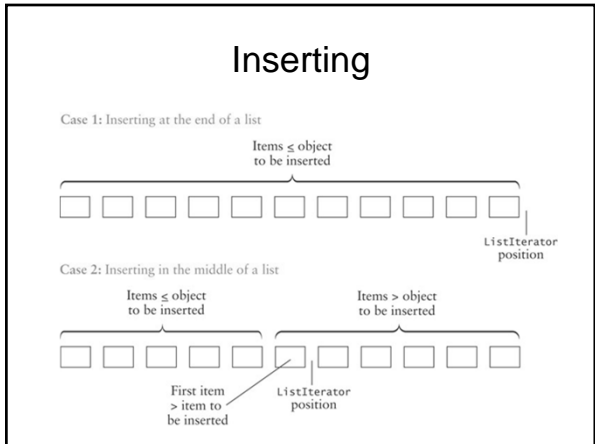
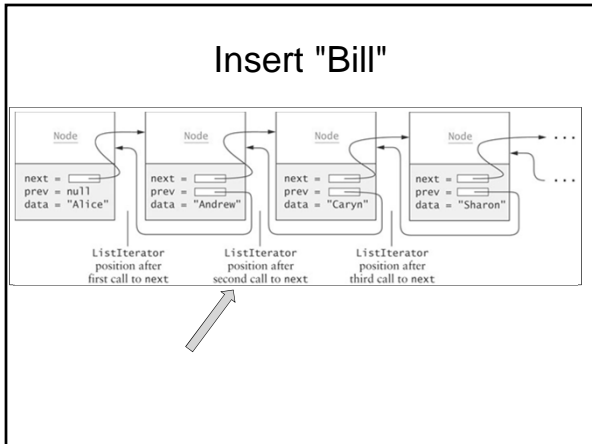
Insert `e` before that item:

Create `ListIterator` that starts at the beginning of the list

While the `ListIterator` is not at the end of the list and `e >=` the next item

Advance the `ListIterator`

Insert `e` before the current `ListIterator` position



Implement Methods Using Delegation

```
public E get (int index) {
    return theList.get(index);
}
public int size () {
    return theList.size();
}
public E remove (E e) {
    return theList.remove(e);
}
public Iterator iterator() {
    return theList.iterator();
}
```

Chapter Review

- The List is a generalization of the array concept
- The Java API ArrayList class uses an array as the underlying structure to implement the List
- The Java API LinkedList class uses a double-linked list to implement the List interface

Chapter Review (continued)

- To find an item at a position indicated by an index in a linked list requires traversing the list from the beginning until the item at the index is found
- An iterator provides access to the items in a List sequentially
- The ListIterator interface extends the capabilities of the Iterator interface

Testing OrderedList

- Store a collection of randomly generated integers in an OrderedList
- Test insertion at beginning of list: insert a negative integer
- Test insertion at end of list: insert an integer larger than any integer in the list
- Create an iterator and iterate through the list, reporting an error if any element is larger than the previous element
- Remove the first element, the last element, and a middle element, then traverse to show that order is maintained