An Application: Ordered List

We want to maintain a list of names in alphabetical order at all times

Approach:

* Develop an OrderedList class (which can be used for other applications)
* Implement a Comparable interface by providing a compareTo(E) method
* Use a LinkedList class as a component of the OrderedList
  + if OrderedList extended LinkedList, the user could use LinkedList’s add methods to add an element out of order

Diagram

Description automatically generated

Filled diamond 🡪 if you distract the OrderedList, LinkedList will be distracted as well. LinkedList is component of OrderedList. Since it is component, then when you distract the OrderedList, LinkedList will be distracted as well.

Empty diamond 🡪 LinkedList contains Comparable objects, but distraction of the LinkedList doesn’t require distraction of the objects that are comparable.

Table

Description automatically generated

Inserting into an OrderedList

Strategy for inserting new element e:

* Find first item > e
* Insert e before that item

Refined with an iterator:

* 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

Diagram

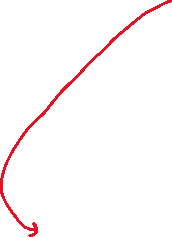
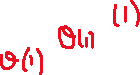
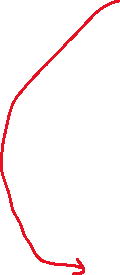
Description automatically generated

Text, timeline

Description automatically generated

Text

Description automatically generated



We don’t know how compareTo is implemented in E. We assumed it is constant time.

Tb(n) = θ(1) 🡪 first place

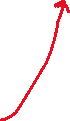
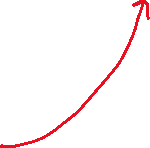
Tw(n) = θ(n)

T(n) = O(n)

Overall, add method takes linear time.

Graphical user interface, text, application, email

Description automatically generated



We delegate methods through list.