Linked Lists and Iterators

3/17/2009

Opening Discussion

- Midterm results.
- Let's look at some solutions to the interclass problem.
- Do you have any questions about the reading?
- Do you have any questions about the assignment?

Implementing a Doubly Linked List

- Now let's implement java.util.List with a doubly linked list with a sentinel. The list will also be circular.
- You should notice that this implementation never has to check for null because no references in the list should ever be null. This simplifies the code significantly. We also implicitly get a head and a tail with no extra work. If you don't have a sentinel you will write a lot of extra checks for nulls and even more to include a tail.

Iterators

- Direct access on linked lists is very inefficient. How then should we walk through a list with outside code? Remember that the outside code doesn't have access to the nodes so it can't use the style of loop we have been doing internally.
- The concept of an Iterator is something that abstracts the process of walking through all of the elements in a container. Iterators can not only be efficient, they also make code more flexible because they don't depend on the implementation details of the containers.

Iterating Lists

- An iterator basically needs to encapsulate the information and functionality we would put into a standard method of going through a container.
- With this in mind, what do we need to put in an iterator for an array based list?
- What would we put in an iterator for a linked list?

Minute Essay

- How do you think you would do a linked list in C? How is it different from the Java code?
- Interclass problem – Write an iterator that goes through our linked list backwards. Try another one that skips every other element.