Finishing Linked Lists

10-7-2011

Opening Discussion

- Minute essays
  - Throws and exceptions.
  - Monday assignments.

Implementing a Singly Linked List

- Let's finish our implementation of the singly linked list.

Sentinels

- A sentinel is an extra node in the list that represents the “end” of the list and doesn't store data.
- The purpose of the sentinel is to remove special cases. The next of the sentinel is what we have called head.
- They are most useful in a doubly linked list where the previous of the sentinel is tail.

Implementing a Doubly Linked List

- Now let's implement Buffer 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.

Stack and Queue with LL

- Remember that the 'A' in ADT means abstract. So you can build multiple implementations.
- Let's redo our implementation of the stack and the queue using a linked list to store the data.

Minute Essay

- Questions?