CSE12 Discussion 2.2

Pooja Bhat

iClicker Participation

- There will be questions sprinkled over the 50 mins time slot.
- You must click at least 75% of the times to get any credit.
- You must attend the discussion the same week as that of the review quiz.

HW1 Grades will be out tonight

- There will be a pinned post on Piazza for regrades. If you have any VALID regrade request, make a follow-up post with the name of your Grader.
- Submission rules: If submission.txt says failed submission, then your submission is not graded.
  - Only for HW1, you can submit a regrade request and get a regrade with a 10 point penalty.
- Please triple-check submission report for HW2 and all future homeworks.

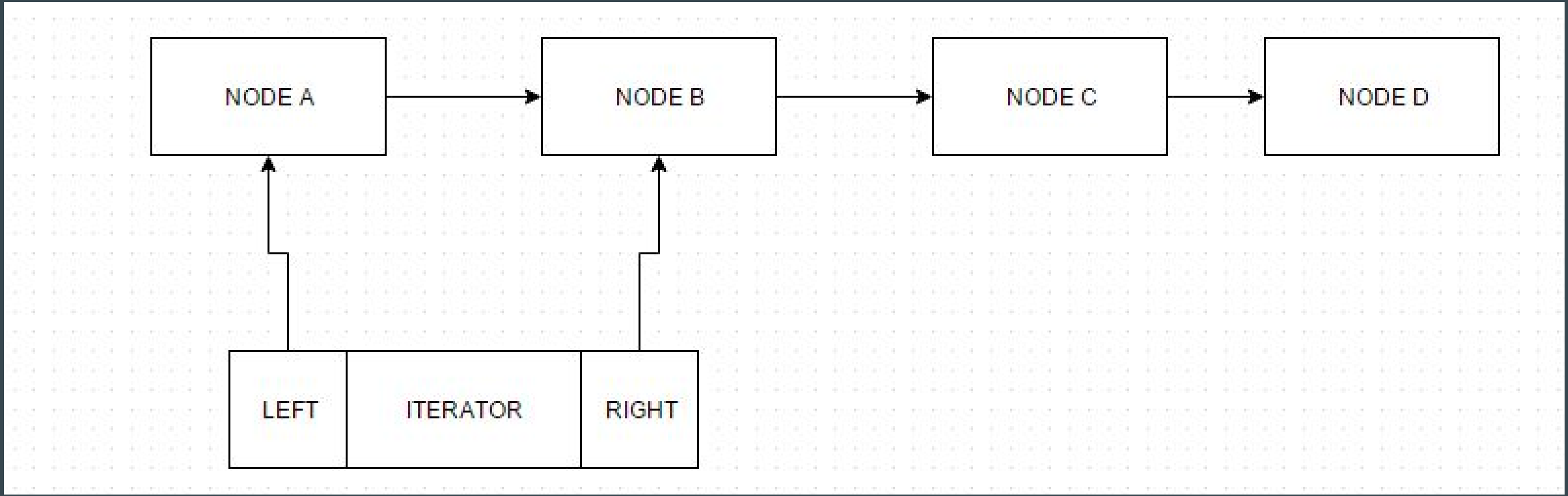
The key to HW2 is

TESTING!!

- Come up with really good tests for your DoublyLinkedList and Iterator classses.
- For example, what are some of the corner cases you can think of for the method add(index, data) in DoublyLinkedList?
  - add(1, 5) //List with 3 items
  - add(100, 5) //List with 3 items
  - add(1, null)
  - add(3, 5) //List with 3 items

Iterators

- Used to move across the LinkedList
- Exists in between nodes
- Points to nodes to its left and right



Where does the Iterator sit in an empty list?

A. Before the dummy head
B. Between the dummy head and tail
C. After the dummy tail
D. Iterator position is invalid for an empty list

Iterator add

Java API says:

Inserts the specified element into the list. The element is inserted immediately before the element that would be returned by next(), if any, and after the element that would be returned by previous(), if any. (If the list contains no elements, the new element becomes the sole element on the list.)

The new element is inserted before the implicit cursor: **a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element.** (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

Which flag is used to keep track of the Iterator direction?

A. forward
B. canRemove
C. canSet
D. ahead

Iterator remove

Removes from the list the last element that was returned by most recent call to next() or previous().

This call can only be made once per call to next or previous. It can be made only if add(E) has not been called after the last call to next or previous.

canRemove Flag

If the user has made a call to next() or previous(), then a method like remove() would remove the element that was recently returned by next() or previous(). However, if the user has not recently called next() or previous() on the iterator, then the iterator would not know which element to remove from the LinkedList.

The canRemove flag keeps track of whether the user has recently called next() or previous() on the iterator.

What methods affect canRemove?
1. Add, next, and previous will change the status of canRemove
2. If remove succeeds depends on the status of canRemove, and it will also change the status of canRemove if the call is valid.
The set( ) method depends on the status of canRemove, but does not modify it.

What node will be deleted after the following code executes?

$$\begin{tikzcd}
20 \arrow[r] & 85 \arrow[r] & 75 \arrow[r] & 50 \arrow[r] & 45 \\
& \text{Iterator} \arrow[u] \arrow[ur] & & &
\end{tikzcd}$$

A. 45
B. 85
C. 50
D. 75

```
next();
next();
remove();
```

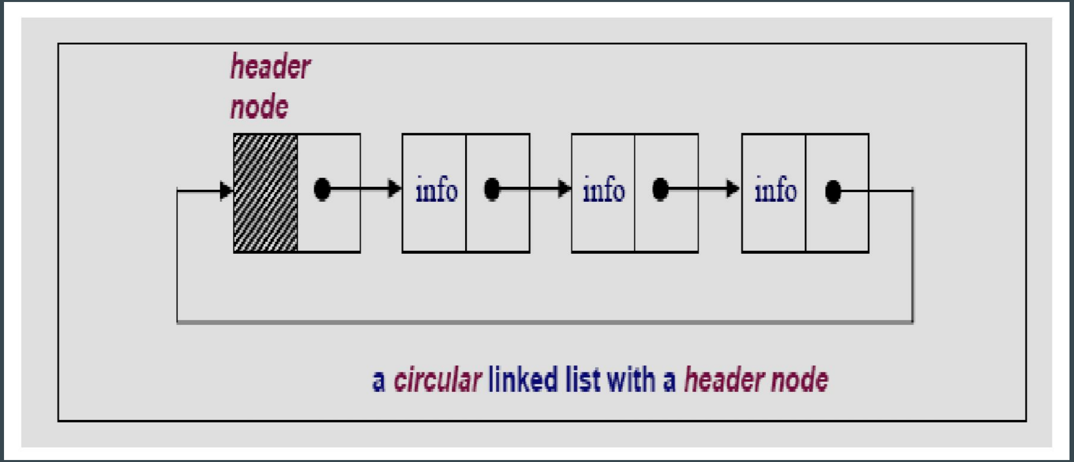
A few tips

- Note that methods like hasNext( ), hasPrevious( ), nextIndex( ) and previousIndex( ) do not move the iterator. The iterator positions remains unchanged after these methods are called.
- The dummy nodes sit at index -1 and index nelems respectively. In other words they do not have any index associated with them. These nodes must not be visible outside the class.
- Read the javadoc carefully on expected behavior, exceptions to throw etc.

A variant of a LinkedList where the last node points back to the first node is

A. DoublyLinkedList
B. Rounded LinkedList
C. Circular LinkedList
D. Hybrid LinkedList

Circular Linked Lists



a *circular* linked list with a *header node*

A linked list where the last node of the list points back to the head. Useful for traversing the list again and again (e.g. a multiplayer game?)

Find out if a given linked list is a singly-linked list or has a cycle in it? (In a list with cycle, the last node may point to ANY other node in the list)

- Pretty useful technique - often called as the ‘hare’ and ‘tortoise’ algorithm for cycle-detection
- Use 2 pointers: a fast pointer and a slow pointer. In each iteration, move the slow pointer by a node and the fast pointer by 2 nodes. If there is no cycle, the fast pointer will eventually reach the end (will be either null or point to null). If there is a cycle, the fast pointer and the slow pointer will eventually meet.

Develop an algorithm called FindMid that returns the value of the node at the middle of a singly linked list without using any form of counter. Assume that you are given the Node and SLList structures (no iterators) and their associated methods.

Also, assume that the total number of nodes in the list is odd.