

Data Structures

Lists II

CS284

Structure of this week's classes

Implementing Lists as Double-Linked Lists

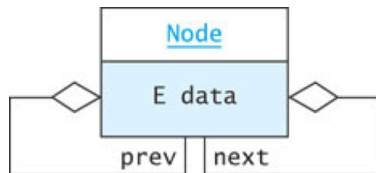
List

- ▶ Last class we introduced lists
- ▶ We studied an array based implementation
- ▶ We also studied a linked-list based implementation (Single Linked Lists)
- ▶ Next we present a double-linked list implementation (Double Linked Lists)
- ▶ Also, we present Iterators

Implementing Lists as Double-Linked Lists

Node Class

```
private static class Node<E> {  
    private E data;  
    private Node<E> next = null;  
    private Node<E> prev = null;  
    private Node(E dataItem) {  
        data = dataItem;  
    }  
    private Node(E dataItem, Node<E> p, Node<E> n ) {  
        data = dataItem;  
        prev = p;  
        next = n;  
    }  
}
```



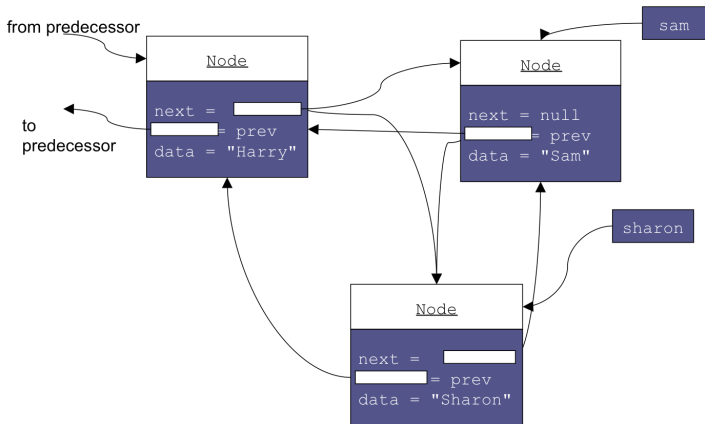
Inserting into a Double-Linked List

```
Node<String> sam = new Node<String>("Sam");  
Node<String> harry = new Node<String>("Harry");  
harry.next = sam;  
sam.prev = harry;
```

- ▶ Let's draw a diagram

Inserting into a Double-Linked List

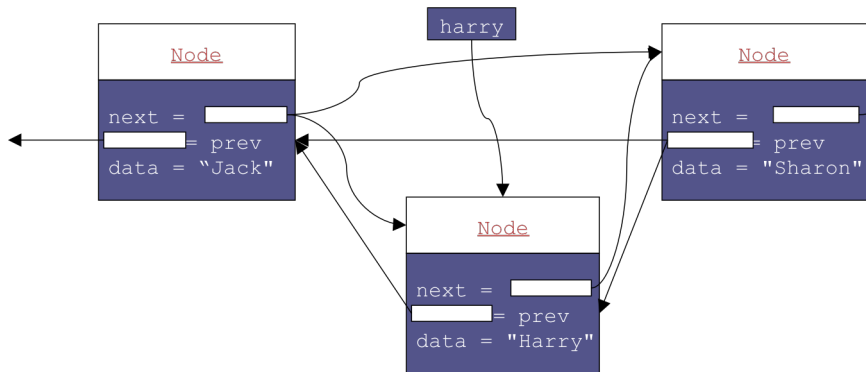
```
Node<String> sharon = new Node<String>("Sharon");  
sharon.next = sam;  
sharon.prev = sam.prev;  
sam.prev.next = sharon;  
sam.prev = sharon
```



How do we remove a node?

Consider the execution of the following additional lines

```
harry.prev.next = harry.next  
harry.next.prev = harry.prev
```



The class `DLList<E>`

```
public class DLList<E> {  
  
    private class Node<E> {  
        /* As defined above */  
        ...  
    }  
    /** The first element in the list */  
    private Node<E> head;  
    /** The last element in the list */  
    private Node<E> tail;  
    /** The size of the list */  
    private int size = 0;  
  
    // Operations should follow  
}
```

Implement `public void add(E item)`

- ▶ This operation should add the item in a new node at the beginning of the list

Double-Linked List

- ▶ So far we have worked only with internal nodes
- ▶ As with the single-linked class, it is best to access the internal nodes with a double-linked list object
- ▶ A double-linked list object has data fields:
 - ▶ head (a reference to the first list Node)
 - ▶ tail (a reference to the last list Node)
 - ▶ size
- ▶ Insertion at either end is $\mathcal{O}(1)$; insertion elsewhere is still $\mathcal{O}(n)$
- ▶ For the second assignment you will be asked to implement an **indexed** double-linked list.

Circular lists

- ▶ Circular double-linked list:
 - ▶ Link last node to the first node, and
 - ▶ Link first node to the last node
- ▶ We can also build singly-linked circular lists:
 - ▶ Traverse in forward direction only
- ▶ Advantages:
 - ▶ Continue to traverse even after passing the first or last node
 - ▶ Visit all elements from any starting point
 - ▶ Never fall off the end of a list
- ▶ Disadvantage: Code must avoid an infinite loop!