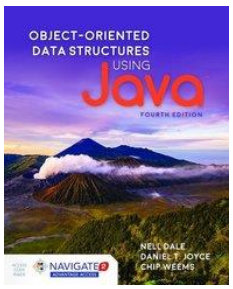
Chapter 6

# The List ADT




Section 5

## DOUBLY LINKED LISTS

## Objectives

- Implement a doubly-linked list data structure
- Explore pros and cons of the single-linked and doubly-linked implementations of lists

3

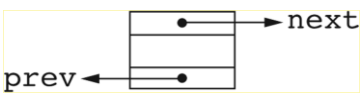
## Doubly Linked Lists

- Singly-linked lists allow one way traversal: one can move from a node to its successor
- Limitation: one cannot easily move from a node to its predecessor
- Doubly-linked lists allow easy transitions in both directions

4

## Nodes for Doubly Linked Lists

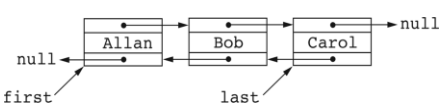
- A node for a doubly linked list must have both a successor link and a predecessor link:




5

## Doubly-Linked Lists

- Can be represented by a head (first) reference and a reference to the last node (for efficiency)




6

## Node Class for Doubly-Linked Lists

```
public class Node<T> {

    protected T data;
    protected Node<T> next;
    protected Node<T> prev;

    public Node( T data) {

        this.data = data;
        this.next = null;
        this.prev = null;
    }

}
```

7

## Operations on Doubly-Linked Lists

- Implementations of `size()` and `isEmpty()` are similar to those of singly-linked lists
- Addition and removals are also similar, but require care to manipulate the two links for successor and predecessors

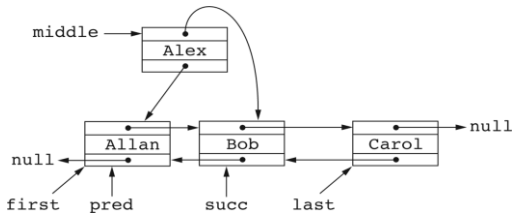
8

## Addition After the Frist Node

- Set `pred` to what will be predecessor of the new node
- Set `succ` to what will be the successor of the new node
- Put the new node in the middle (between) `pred` and `succ`

9

## Addition After the Frist Node




10