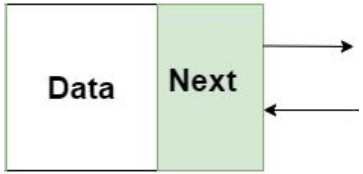
COSC 121: Computer Programming II

Today's Key Concepts

- Implementing data structures from scratch
  - An ArrayList using an array
  - A LinkedList using a Node structure
- Handling nuances
- Developing your own test cases

Implementing Linked List

- List now contains nodes
- Singly linked list:




```
public class Node<E> {
    E element;
    Node<E> next;
    public Node( E elem ) {
        element = elem;
        next    = null;
    }
}
```

- Doubly linked list:


```
public class Node<E> {
    Node<E> prev;
    E element;
    Node<E> next;
    public Node( E elem ) {
        prev    = null;
        element = elem;
        next    = null;
    }
}
```

Review linked list operations and check out MyLinkedList.java

Exercise: Implement LinkedList Methods

- Try implementing:
  - **public void** add(E elem)
  - **public** E set(**int** index, E elem)
  - **public** E get(**int** index)
  - **public boolean** remove(E elem)
  - **public int** indexOf(E elem)
  - **public boolean** contains(E elem)
  - **public int** size()

Things to consider:

- What is being added or deleted from the list? (Is it elem?)
- How to find end an item or the end of the list? (How to traverse the list?)
- Do we need to check for valid indices anymore?
- Does the list ever get full?
- Does addition/removal look different when list is empty or almost empty?

Exercise: Implement Doubly Linked List Methods

- Try implementing:
  - **public void** add(E elem)
  - **public** E set(**int** index, E elem)
  - **public** E get(**int** index)
  - **public boolean** remove(E elem)
  - **public int** indexOf(E elem)
  - **public boolean** contains(E elem)
  - **public int** size()

Things to consider:

- Keeping track of a tail node (in addition to the head node from before)
- How to find end an item or the end of the list? (How to traverse the list?)
- Do we need to check for valid indices anymore?
- Does the list ever get full?
- Does addition/removal look different when list is empty or almost empty?