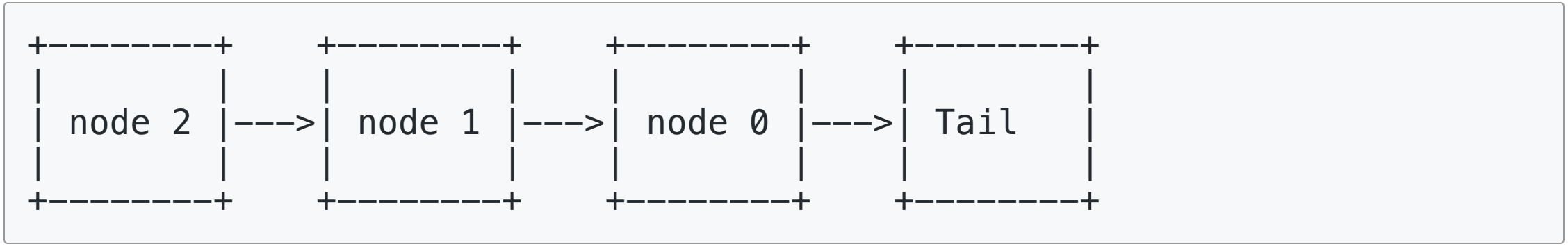
LinkedList

(Singly) LinkedList

A linked list is a linear data structure where each element is a separate object.



LinkedList (Cont'd)

- Each element (we will call it a node) of a list is comprising of two items:
  - the data
  - a reference to the next node.
- The last node has a reference to null. The entry point into a linked list is called the ***head*** of the list.
- *head* is not a separate node, but the reference to the first node.
  - If the list is empty then the head is a `nil` reference.

Time Complexity: LinkedList vs. Array

	LinkedList	Dynamic Array
Indexing	O(n)	O(1)
Insert/delete at beginning	O(1)	O(n)
Insert/delete at end	O(1)	O(1)
Insert/delete in middle	O(1)	O(n)
Wasted space (average)	O(n)	O(n)

LinkedList (Cont'd)

- A linked list is a dynamic data structure.
    - The number of nodes in a list is not fixed and can grow and shrink on demand.
- One disadvantage of a linked list against an array is that it does not allow direct access to the individual elements.
    - If you want to access a particular item then you have to start at the head and follow the references until you get to that item.
- Another disadvantage is that a linked list uses more memory compare with an array - we store extra 4 bytes (on 32-bit CPU) as reference to the next node.

LinkedList Design

Simple `Node` implementation

```
public class Node<T> {
    private T data;
    private Node<T> next;

    public Node(T data, Node<T> next) {
        this.data = data;
        this.next = next;
    }
}
```

LinkedList Design (Cont'd)

`Node` implementation which inherits `T` from enclosing class

```
public class LinkedList<T> {
    class Node {
        private T data;
        private Node<T> next;

        public Node(T data, Node next) {
            this.data = data;
            this.next = next;
        }
    }

    public LinkedList() {
        // ...
    }
}
```

`static` nested class in Java vs. C#

- In Java, the code in the outer class has access to private members in the containing class. In C#, it's the other way around.
- In Java, only nested classes can be static. In C#, this constraint is not required.
- In Java, we can instantiate nested static class but in C# we cannot.

Code Design Overview

```
public class ADT<T> {

    private static class Node {
        private Node<T> _next;
        private T _data;

        Node(T data, Node<T> next) {
            _data = data;
            _next = next;
        }
    }

    public ADT() {
        Node<String> head = new Node<String>("Hello world!", null);

        // Notice how we can access private fields of Node
        Node<String> nextNodeRef = node._next;
        String data = next._data;
    }
}
```

Add to LinkedList (TODO)

```
public class ADT<T> {

    private Node<T> _head;

    // O(1) time complexity
    public void add(T value) {

        // TODO:
    }
}
```

Add to LinkedList (Solution)

```
public class ADT<T> {

    private Node<T> _head;

    // O(1) time complexity
    public void add(T value) {
        // Create a new node
        Node<T> node = new Node<T>(value, _head);

        // Re-assign the head ref
        _head = node;
    }
}
```

Remove a value from LinkedList (TODO)

```
public class ADT<T> {

    private Node<T> _head;

    public void remove(T value) {
        Node<T> precursor = null;
        Node<T> cursor = _head;

        // TODO:
    }
}
```

Remove a value from LinkedList (Solution)

```
public class ADT<T> {

    private Node<T> _head;

    public void remove(T value) {
        Node<T> precursor = null;
        Node<T> cursor = _head;

        // Loop through the LinkedList
        while (cursor != null) {
            // Test if we found the target node
            if (cursor._data == value) {

                // Target is the first element in the list
                // Remove the first node from the chin
                if (precursor == null) {
                    _head = _head.next;
                    break;
                }
                // Target is the last element in the list
                // Remove the last element from the chain
                else if (cursor.next == null) {
                    precursor.next = null;
                    break;
                }

                // Remove the ref
                else {
                    precursor.next = cursor.next;
                    break;
                }
            }

            precursor = cursor;
            cursor = cursor.next;
        }
    }
}
```

Lab assignment #4: