

# CS 213 : Software Methodology

Spring 2019

*Sesh Venugopal*

Lecture 7: Feb 12

Inner Classes

# Inner Classes

```
public class LinkedList<T> {
```

```
    public static class Node<E> { // inner class
```

```
        E data;
```

```
        Node<E> next;
```

```
        public Node(E data,  
                    Node<E> next) {...}
```

```
    }
```

```
    Node<T> front;
```

```
    int size;
```

```
    ...
```

```
    public void addFront(T item) {
```

```
        front = new Node<T>(item, front);  
        size++;
```

```
    }
```

```
    ...
```

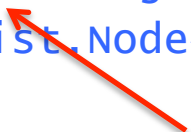
```
}
```

Since nodes are the building blocks of linked lists, a `Node` class can be defined inside a linked list to emphasize this (will get to the static thing in a bit....)

Inside the `LinkedList` class, references to the `Node` type are no different than if `Node` had been defined outside `LinkedList`

# Inner Classes

```
public class LinkedList<T> {  
  
    public static class Node<E> { // inner class  
        E data;  
        Node<E> next;  
        public Node(E data,  
                     Node<E> next) {...}  
    }  
    ...  
}  
  
// in some application code outside of LinkedList class  
LinkedList.Node<Integer> temp =  
    new LinkedList.Node<Integer>(10, null);
```



Reference to Node needs to be  
qualified with LinkedList prefix

# Non Static Inner Class

```
public class LinkedList<T> {  
  
    public static class Node<E> { // inner class  
        E data;  
        Node<E> next;  
        public Node(E data,  
                    Node<E> next) {...}  
    }  
    ...  
}  
  
// in some application code outside of LinkedList class  
LinkedList<Integer>.Node<Integer> temp =  
    new LinkedList<Integer>().new Node<Integer>(10, null);
```



Can only create a Node instance off  
a LinkedList instance