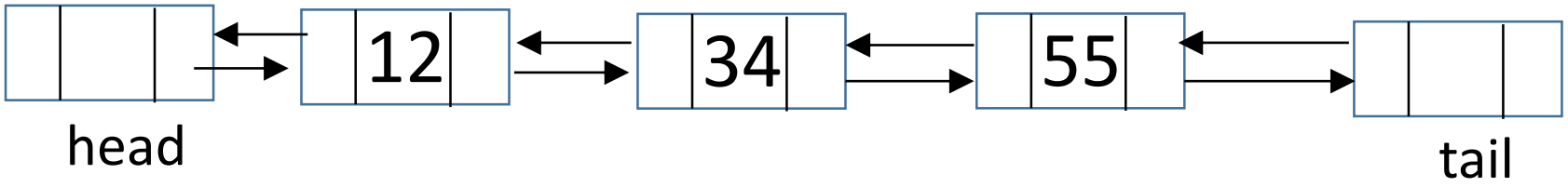
Notes About Lab 4

Lab 4 has 3 parts:

A. Create a doubly-linked list: every node has a link to both the *next* node and the *previous* node.
B. Create an *iterator* for the list
C. Run a test program an algorithm both with and without the iterators to see why they are used. This part involves no coding.

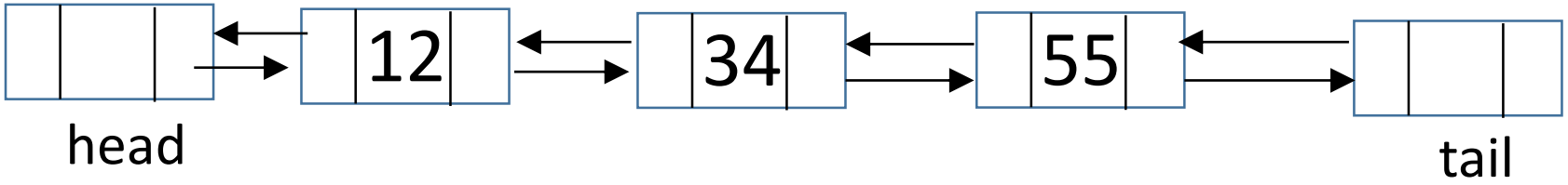


Here is a picture of a typical doubly-linked list containing values 12, 34, and 55. Note that we have *sentinel* nodes – empty boxes – at each end.

We will have a Node class:

```
class Node {
        T data;
        Node next, prev;
}
```

In my pictures *next* points at the box to the right and *prev* at the box to the left.



Most list operations refer to a specific index. To get to the node at index n, we start at the head and do n+1 *next*s:

```
Node p = head;
for( int i=0; i <= n; i++ )
        p = p.next;
```

The lab directions suggest you make a private method

```
Node getNth(int n)    // returns p, not p.data
```

out of this, and use it to implement get(), set(), add() and remove()

You need to read the lab directions carefully and draw pictures of your structures as you code, but if you do that your structures will work. I think this is fun programming.

The place students tend to find confusing in Lab 4 is the work with iterators, so we will talk about them next.