CSE 143
Computer Programming II
List Nodes
Today’s Goals

- Get familiar with the idea of “references” (things that point to objects)

- Define and explore `ListNode`

- Learn about `null`

- Practice modifying linked lists

- Get familiar with matching up code and pictures of linked lists
Consider the following two documents in a text editor:

- A normal book
- A “choose your own adventure” book

Which tasks are easy/hard in each type of book?

- Find the last page
- Add a new page in the middle of the book
- Add a new page at the end of the book

Books as Data Structures

- Arrays are stored in memory like a normal book; it’s contiguous, and random-access
- For the next three lectures, we’ll discuss the data structure equivalent to a “choose your own adventure” book
What does this code print?

```
1 List<Integer> list1 = new ArrayList<Integer>();
2 list1.add(8);
3 list1.add(3);
4 List<Integer> list2 = new ArrayList<Integer>();
5 list2.add(100);
6 List<Integer> list3 = list2;
7 list2 = list1;
8 list2.add(5);
9 list1.add(2);
10 System.out.println("A: " + list1);
11 System.out.println("B: " + list2);
12 System.out.println("C: " + list3);
```

OUTPUT

```
>> A: [8, 3, 5, 2]
>> B: [8, 3, 5, 2]
>> C: [100]
```
1. List<Integer> list1 = new ArrayList<Integer>(); //o1
2. list1.add(8);
3. list1.add(3);

4. List<Integer> list2 = new ArrayList<Integer>(); //o2
5. list2.add(100);

6. List<Integer> list3 = list2;
7  list2 = list1;

What's Going On?
- The keyword `new` creates an actual new object to point to (o1, o2).
- All the other variables just point to objects that were created with `new` (list1, list2, list3).
A ListNode is:

The box represents data, and the arrow represents next.

Since next is of ListNode type, the arrow can either point to nothing (null) or another ListNode.
ListNode Class
1 public class ListNode {
2     int data;
3     ListNode next;
4 }

How can we use code to make this list?

```
1 ListNode list = new ListNode();
2 list.data = 5;
3 list.next = new ListNode();
4 list.next.data = 10;
5 list.next.next = new ListNode();
6 list.next.next.data = 15;
```
```
public class ListNode {
    int data;
    ListNode next;
}
```

How can we use code to make this list?

```
ListNode node = list.next;
list.next = list.next.next;
list.next.next = node;
```

What does this code do to our list?

1. `ListNode node = list.next;`
   - This assigns the next node of `list` to `node`.
2. `list.next = list.next.next;`
   - This updates the next node of `list` to be the next node of the current next node.
3. `list.next.next = node;`
   - This assigns the previous next node of the current next node to `node`.

This isn't quite... What's wrong?
list.next.next.next = list.next;

list
↓
a b c d e

The code sets **the arrow** coming out of c to **the node** d.

list
↓
a b c d e

The **left side** of the assignment is **an arrow**.

The **right side** of the assignment is **a node**.
Dereferencing

When we call `.next`, we follow an arrow in the list. What happens if we have this list:

```
list
↓
5 → 15 → 10 →
```

And we call the following code:

```
1 System.out.println(list.next.next.next);
```

Or this code:

```
1 System.out.println(list.next.next.next.data);
```

The first one prints `null`. The second throws a `NullPointerException`.

`null` means “end of the list”!
public class ListNode {
    int data;
    ListNode next;

    public ListNode(int data) {
        this(data, null);
    }

    public ListNode(int data, ListNode next) {
        this.data = data;
        this.next = next;
    }
}

What list does this code make?

ListNode list = new ListNode(1, null);
list.next = new ListNode(2, null);
list.next.next = new ListNode(3, null);

Can we do this without ever using .next?

ListNode list = new ListNode(1, new ListNode(2, new ListNode(3, null)));