**CSE 143**

Computer Programming II

---

**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

---

**List Nodes**

---

**Memory**

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

---

**Mystery**

```java
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);
```

What does this code print?

**OUTPUT**

```
> A: [8, 3, 5, 2]
> B: [8, 3, 5, 2]
> C: [100]
```

---

**Mystery Explained**

```java
1. List<Integer> list1 = new ArrayList<Integer>(); //o1
2. list1.add(8);
3. list1.add(3);
o1: 8 3 0 1
4. List<Integer> list2 = new ArrayList<Integer>(); //o2
5. list2.add(100);
o2: 100 0
6. List<Integer> list3 = list2;
o1: 8 3 0 1
o2: 100 0
```

```java
list1
  o1: 8 3 ...
```

```java
list2
  o2: 100 ...
```

```java
list3
```

```java
```

--

What does this code print?

**OUTPUT**

```
> A: [8, 3, 5, 2]
> B: [8, 3, 5, 2]
> C: [100]
```
Mystery Explained (cont.)

7  list2 = list1;

8  list1.add(5);
9  list1.add(2);

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).

ListNode

1 public class ListNode {
2   int data;
3   ListNode next;
4 }

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.

Working With Linked Lists

```java
list.next.next.next = list.next;
```

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

Dereferencing

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

```
list
```

And we call the following code:

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

Or this code:

```java
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)));