So far, the only “real data structure” we’ve seen is *arrays*.

What are some limitations of arrays?

- You need to know the size before declaration
- Adding/removing can be annoying
- They have no methods

This is where the idea of a *list* comes in.
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

What happens to the page numbers when we . . .

- 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?

```
int[] a1 = new int[2];
a1.x = 8;
a1.y = 3;
int[] a2 = new int[2];
a2.x = 100;

int[] a3 = a2;
a2 = a1;
a2.x = 5;
a1.y = 2;
System.out.println("A: " + a1.x + ", " + a1.y);
System.out.println("B: " + a2.x + ", " + a2.y);
System.out.println("C: " + a3.x + ", " + a3.y);
```

OUTPUT

```
>> A: 5, 2
>> B: 5, 2
>> C: 100, 0
```
1 int[] a1 = new int[2]; //o1
2 a1.x = 8;
3 a1.y = 3;

4 int[] a2 = new int[2]; //o2
5 a2.x = 100;

6 int[] a3 = a2;
Mystery Explained (cont.)

7  a2 = a1;

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` (`a1`, `a2`, `a3`).

8  a2.x = 5;
9  a1.y = 2;
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.
```java
public class ListNode {
    int data;
    ListNode next;
}

How can we use code to make this list?

```

```java
ListNode list = new ListNode();
list.data = 5;
list.next = new ListNode();
list.next.data = 10;
list.next.next = new ListNode();
list.next.next.data = 15;
```
**ListNode Class**

```java
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?

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

What's wrong?

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

This isn’t quite

What’s wrong?
list.next.next.next = list.next;

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

The left side of the assignment is an arrow.
The right side of the assignment is a node.
When we call `.next`, we follow an **arrow** in the list. What happens if we have this list:

```
5  ▼
   ▶ 15 ▶ 10 ▶
```

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)));
Linked Lists I
1. Get more familiar with ListNodes
2. Learn how to run through the values of a LinkedList
3. Learn how LinkedIntList is implemented
Quick Note: When I say “does that make sense?”.

- If it does make sense, yell “yes”

- Otherwise, say nothing.
1. Get more familiar with ListNode

2. Learn how to run through the values of a LinkedList

3. Learn how LinkedIntList is implemented
Another ListNode Example

Before:

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>list2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

After:

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>list2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

How many ListNode are there in the before picture?

**There are FOUR. Each box is a ListNode.**

How many references to ListNode are there?

**There are SIX. Every arrow is a reference to a ListNode.**
Another ListNode Example (Solution)

Before:
```
1 2 3 4
0 1 2 3
list
4
list2
5
```

After:
```
1 2 4 3
0 1 2 3
list
4
list2
5
```

1. list.next.next = list2.next
2. list2.next.next = list2;
3. list2.next = null;
Outline

1. Get more familiar with ListNode
2. Learn how to run through the values of a LinkedList
3. Learn how LinkedIntList is implemented
Printing a LinkedList

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

Now, note that we can use a variable to keep track of where we are:

```java
1 System.out.println(list.data);  
2 list = list.next; 
1 2 3 
3 System.out.println(list.data);  
4 list = list.next; 
1 2 3 
5 System.out.println(list.data);  
6 list = list.next; 
1 2 3 
```
Printing a LinkedList: Better Version

What if our list has 1000 nodes? That would be horrible to write.

```
list
\downarrow
1 \rightarrow 2 \rightarrow \cdots \rightarrow 1000
```

But that destroys the list; so, use a temporary variable instead:

Printing a BIG LinkedList Correctly

```
ListNode current = list
while (current != null) {
    System.out.println(current.data);
    current = current.next;
}
```
Outline

1. Get more familiar with ListNode
2. Learn how to run through the values of a LinkedList
3. Learn how LinkedIntList is implemented
- No generics (only stores ints)

- Fewer methods: `add(value), add(index, value), get(index),
  set(index, value), size(), isEmpty(), remove(index),
  indexOf(value), contains(value), toString()`

- This is the same idea as when we implemented `ArrayIntList`!