Outline

1. Get more familiar with ListNode
2. Learn how to run through the values of a LinkedList
3. Learn how LinkedIntList is implemented
4. Learn about the different cases to deal with for LinkedLists

Another ListNode Example

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

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

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

Printing a LinkedList

Printing a LinkedList Manually

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

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

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

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

```
Printing a BIG LinkedList
while (list != null) {
  System.out.println(list.data);
  list = list.next;
}
```

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;
}
```

LinkedList vs. ArrayList

We can use for loops in a similar way to with ArrayLists to run through LinkedLists!

```
Traversing an ArrayList
for (int i = 0; i < arrayList.size(); i++) {
  System.out.println(arrayList.get(i));
}
```

```
Traversing an LinkedList
for (ListNode current = linkedList; current != null; current = current.next) {
  System.out.println(current.data);
}
```

LinkedIntList

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!

LinkedIntList Fields

What fields does our LinkedIntList need?

```
public class LinkedIntList {
  private ListNode front;
  ...
}
```

LinkedIntList toString()

Buggy toString()

```
public String toString() {
  String result = "[";
  ListNode current = this.front;
  while (current != null) {
    result += current.data + ", ";
    current = current.next;
  }
  return result + "]";
}
```

Our toString() puts a trailing comma. Fix it by stopping one early:

```
Fixed toString()
public String toString() {
  String result = "[";
  ListNode current = this.front;
  while (current != null && current.next != null) {
    result += current.data + ", ";
    current = current.next;
  }
  if (current != null) {
    result += current.data;
  }
  return result + "]";
}
```

Modifying LinkedLists

Writing a LinkedList Method

```
1 Identify cases to consider...
2 Front/Empty
3 Middle
4 End
5 Draw pictures for each case
6 Write each case separately
```

```
front middle end
```
Add To An Empty List
What does an empty list look like?

```
public void add(int value) {
    if (this.front == null) {
        this.front = new ListNode(value);
    } else {
        ListNode current = this.front;
        while (current != null) {
            current = current.next;
        }
        current = new ListNode(value);
    }
}
```

Add To A Non-Empty List (Fixed)
Consider a non-empty list:

```
public void add(int value) {  
  ListNode current = this.front;
  while (current != null) {
    current = current.next;
  }
  current = new ListNode(value);
}
```

Working with LinkedLists
There are only two ways to modify a LinkedList:

- **Change front**
  
  ![Diagram of changing the front of a LinkedList](image)

- **Change current.next** for some ListNode, current
  
  ![Diagram of changing current.next](image)

Settting “current” does NOTHING!

Some LinkedList Tips!
- Be able to deal with before-and-after ListNode pictures
- Know how to loop through a LinkedList
  - Use a while loop.
  - Don’t forget to create a ListNode current variable so we don’t destroy the original list.
  - Don’t forget to update the current variable.
- Understand differences and similarities between ArrayList and LinkedList
  - They both have the same functionality (add, remove, etc.)
  - But they’re implemented differently (array vs. LinkedLists)
- With LinkedLists, you often have to stop one node before the one you want.
- DO NOT start coding LinkedList problems without drawing pictures first.