CSE 143

Computer Programming II
Linked Lists I
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
Bring the beard back!!!

PIZZA!

Who’s in Evee’s linked list?
Can you have options in Java like you can in Pokémon?

What I picked up for class today.

NODE-CEPTION!

Even got, born not fami.

Actual Question © Next Page —

CSE 143
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

4. Learn about the different cases to deal with for LinkedLists
Another ListNode Example

Before:

```
list
<table>
<thead>
<tr>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
---

list2
<table>
<thead>
<tr>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
```

After:

```
list
<table>
<thead>
<tr>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
---

list2
<table>
<thead>
<tr>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</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:

```
list
  1 --> 2 --> 3 --> 4
  ^      ^      ^
  4      1      0
```

list2
  3 --> 4
  ^      ^
  5      3

After:

```
list
  1 --> 2 --> 4 --> 3
  ^      ^      ^
  4      1      2
```

list2
  3 --> 2
  ^      ^
  5      3

1. `list.next.next = list2.next`
2. `list2.next.next = list2;`
3. `list2.next = null;`
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
Printing a LinkedList

Printing a LinkedList Manually

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:

1. `System.out.println(list.data);`
2. `list = list.next;`
3. `System.out.println(list.data);`
4. `list = list.next;`
5. `System.out.println(list.data);`
6. `list = list.next;`
What if our list has 1000 nodes? That would be horrible to write.

```
list
1 2 ... 1000
```

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

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

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

```
ListNode current = list
while (current != null) {
    System.out.println(current.data);
    current = current.next;
}
```
We can use for loops in a similar way to with ArrayLists to run through LinkedLists!

**Traversing an ArrayList**

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

**Traversing an LinkedList**

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

<table>
<thead>
<tr>
<th>Description</th>
<th>ArrayList Code</th>
<th>LinkedList Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to front of list</td>
<td>int i = 0;</td>
<td>ListNode current = list;</td>
</tr>
<tr>
<td>Test for more elements</td>
<td>i &lt; list.size()</td>
<td>current != null</td>
</tr>
<tr>
<td>Current value</td>
<td>list.get(i)</td>
<td>current.data</td>
</tr>
<tr>
<td>Go to next element</td>
<td>i++;</td>
<td>current = current.next;</td>
</tr>
</tbody>
</table>
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
- 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`!
What fields does our LinkedIntList need?

A reference to the front of the list

```java
public class LinkedIntList {
    private ListNode front;

    public LinkedIntList() {
        front = null;
    }

    ...
}
```
LinkedIntList toString()
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
Writing a LinkedList Method

1. Identify cases to consider...
   - Front/Empty
   - Middle
   - End

2. Draw pictures for each case

3. Write each case separately
Cases to consider:
- Add to empty list
- Add to non-empty list

Add To An Empty List

What does an empty list look like?

```java
public void add(int value) {
    /* If the list is empty... */
    if (this.front == null) {
        this.front = new ListNode(value);
    }
    /* Other Cases ... */
}
```
Consider a non-empty list:

```
front

1 → 2 → 3 → ... → 100
```

/* Idea: We want to change the red arrow. 
   Loop until we’re at the last node. */

```java
ListNode current = this.front;

while (current != null) {
    current = current.next;
}

current = new ListNode(value);
```
Consider a non-empty list:

```
1 2 3 ... 100
```

```
/* Idea: We want to change the red arrow. 
   Loop until we’re at the node before the last node */
ListNode current = this.front;

while (current.next != null) {
    current = current.next;
}

current.next = new ListNode(value);
```
There are only two ways to modify a LinkedList:

- **Change front**

  ![Diagram showing changing the front of a LinkedList]

  \[
  \text{front} \quad 1 \rightarrow 2 \rightarrow 3 \quad \text{...changing front...} \quad \text{front} \quad 1 \rightarrow 2 \rightarrow 3
  \]

  \[
  (\quad [1,2,3]; \quad \text{...changing front...} \quad [2,3])
  \]

- **Change current.next for some ListNode, current**

  ![Diagram showing changing current.next of a LinkedList]

  \[
  \text{front} \quad 1 \rightarrow 2 \rightarrow 3 \quad \text{...changing .next...} \quad \text{front} \quad 1 \rightarrow 2 \rightarrow 3
  \]

  \[
  (\quad [1,2,3]; \quad \text{...changing .next...} \quad [1,3])
  \]

  **Setting “current” does NOTHING!**
```java
public int get(int index) {
    ListNode current = front;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    return current.data;
}
```
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. ListNode)

- With LinkedLists, you often have to stop **one node before the one you want**.

- DO NOT start coding LinkedList problems without drawing pictures first.