

LEC 06

CSE 123

LinkedIntList

Questions during Class?

Raise hand or send here

sli.do #cse123



BEFORE WE START

*Talk to your neighbors:**What's your favorite form of
potato?*Music: [123 24su Lecture Tunes](#) **Instructor:** Joe Spaniac**TAs:** Andras Daniel Eric Nicole Sahej Trien Zach

Lecture Outline

- **Announcements/Reminders** 
- LinkedList
 - ListNodes cont.
- Why curr?
- Modifying LinkedLists
 - Special cases (MFEE)

Announcements

- R1 and P1 feedback releasing tonight sometime after lecture
- Creative Project 2 due tonight (7/9) at 11:59pm
 - Submit *something* so we can provide some feedback!
- Check-in 2 in section on Thursday (7/10)
 - Very, *very* similar problem to what you might see on a quiz
 - Guaranteed to get feedback before the quiz on Tuesday if you attend
- Programming Project 2 releases tomorrow (7/10)
 - One of the trickier assignments in the course
 - 2 weeks to complete this one! Feel free to take a breather if necessary but get started sooner than later
- Quiz 2 this upcoming Tuesday (7/15)
 - Topics: Abstract Classes, ArrayList, LinkedList

Lecture Outline

- Announcements/Reminders
- **LinkedList** 
 - ListNodes cont.
- Why curr?
- Modifying LinkedLists
 - Special cases (MFEE)

Reminder: Implementing Data Structures

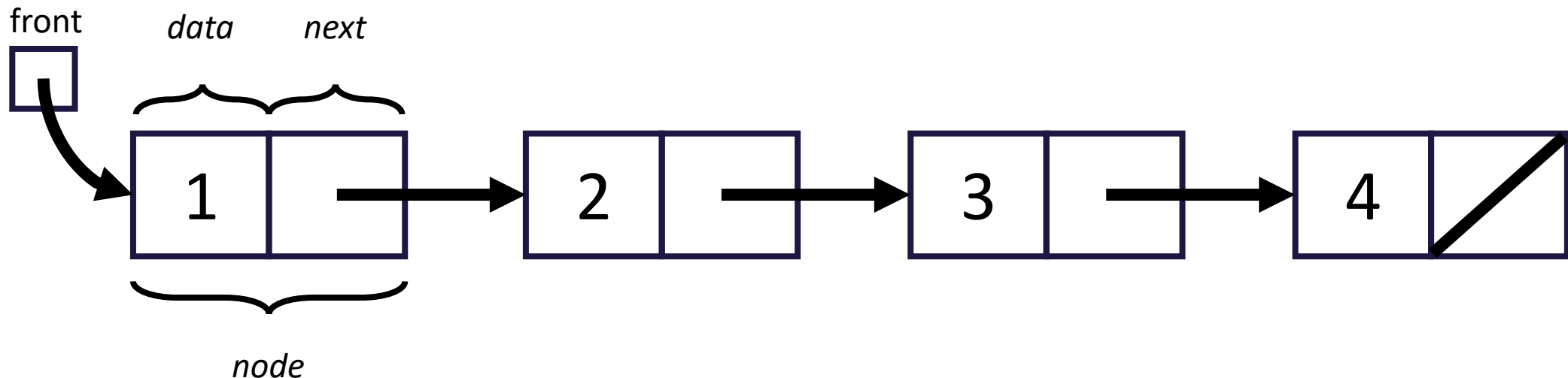
- No different from designing any other class!
 - Specified behavior (List interface):

Method	Description
<code>add(E value)</code>	Adds the given value to the end of the list
<code>add(int index, E value)</code>	Adds the given value at the given index
<code>remove(E value)</code>	Removes the given value if it exists
<code>remove(int index)</code>	Removes the value at the given index
<code>get(int index)</code>	Returns the value at the given index
<code>set(int index, int value)</code>	Updates the value at the given index to the one given
<code>size()</code>	Returns the number of elements in the list

- Choose appropriate fields based on behavior
- Just requires some thinking outside the box

LinkedList

- Goal: leverage non-contiguous memory usage
 - How? `LinkedListNodes`!
- What field(s) do we need to keep track of?
 - `ListNode front;` `// First node in the chain`
 - `int size;` `// Strictly necessary?`



ListNode cont.

- Now that we have `LinkedList`, will a client ever need to use a `ListNode`?
 - No! Not something they should have to worry about
- How can we abstract `ListNode` away from them?
 - Leaving them in a public file is pretty obvious...
- What if we made `ListNode` a private class inside `LinkedList`?
 - We can still access it (just like private fields)
 - Clients won't even know the class exists!
- Do fields need to be private if the entire class is private?

Lecture Outline

- Announcements/Reminders
- LinkedList
 - ListNodes cont.
- **Why curr?** 
- Modifying LinkedLists
 - Special cases (MFEE)

Reminder: Iterating over ListNodes

- General pattern iteration code will follow:

```
ListNode curr = front;
while (curr != null) {
    // Do something

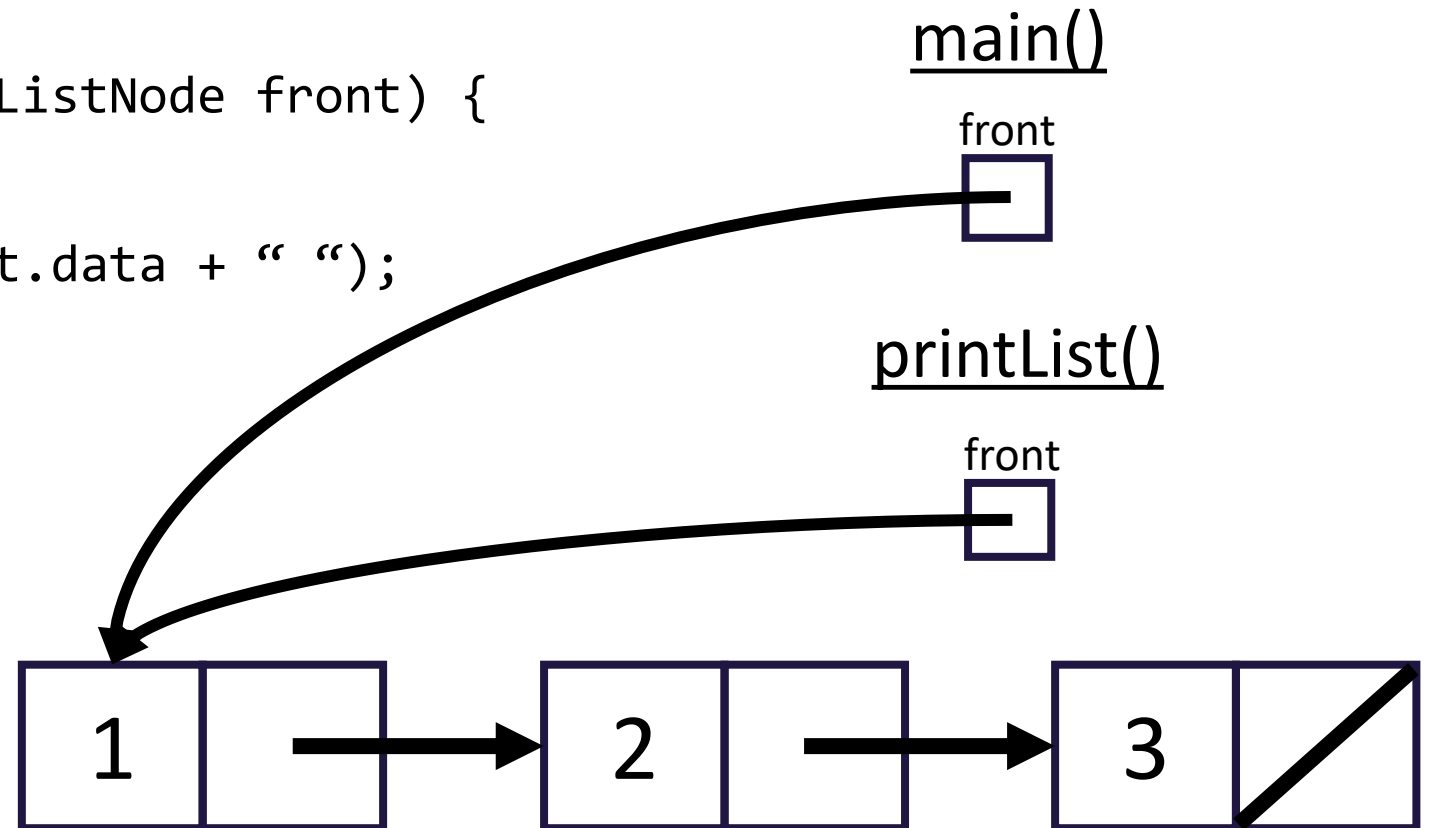
    curr = curr.next;
}
```

Why do we need a ListNode curr?

Why curr?

```
public static void main(String[] args) {  
    ListNode front = new ListNode(1, new ListNode(2, new ListNode(3)));  
}
```

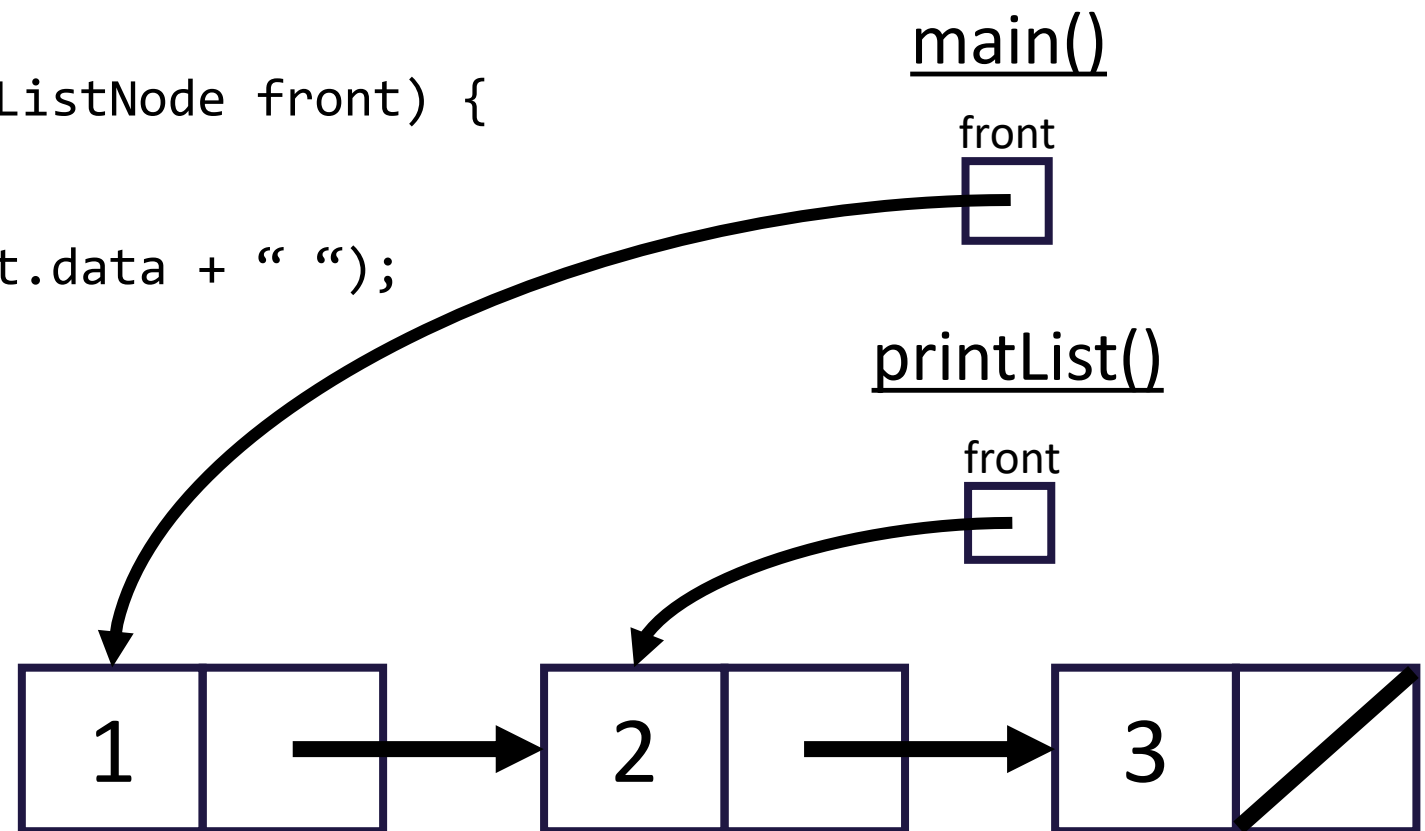
```
public static void printList(ListNode front) {  
    while (front != null) {  
        System.out.print(front.data + " ");  
        front = front.next;  
    }  
    System.out.println();  
}
```



Why curr?

```
public static void main(String[] args) {  
    ListNode front = new ListNode(1, new ListNode(2, new ListNode(3)));  
}
```

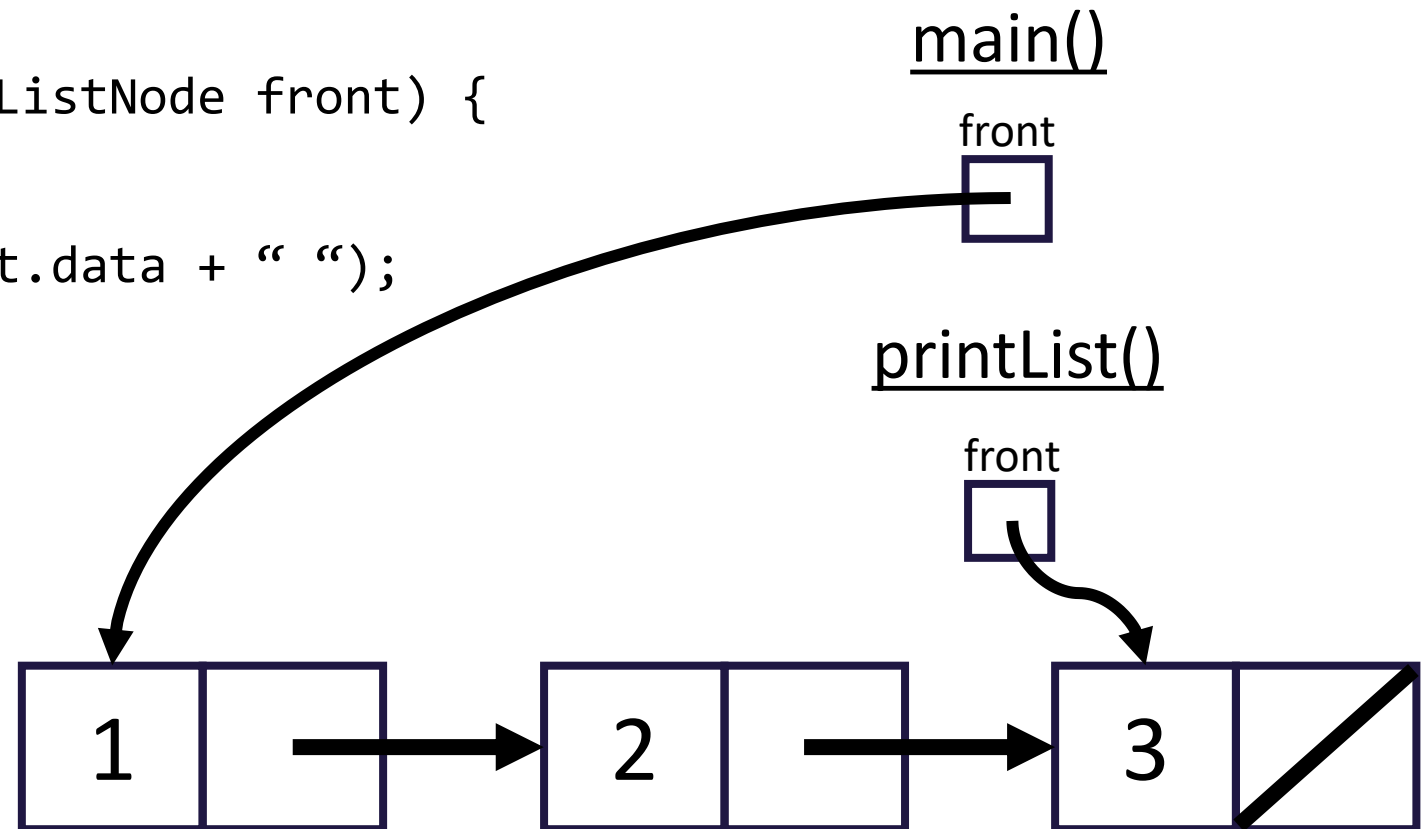
```
public static void printList(ListNode front) {  
    while (front != null) {  
        System.out.print(front.data + " ");  
        front = front.next;  
    }  
    System.out.println();  
}
```



Why curr?

```
public static void main(String[] args) {  
    ListNode front = new ListNode(1, new ListNode(2, new ListNode(3)));  
}
```

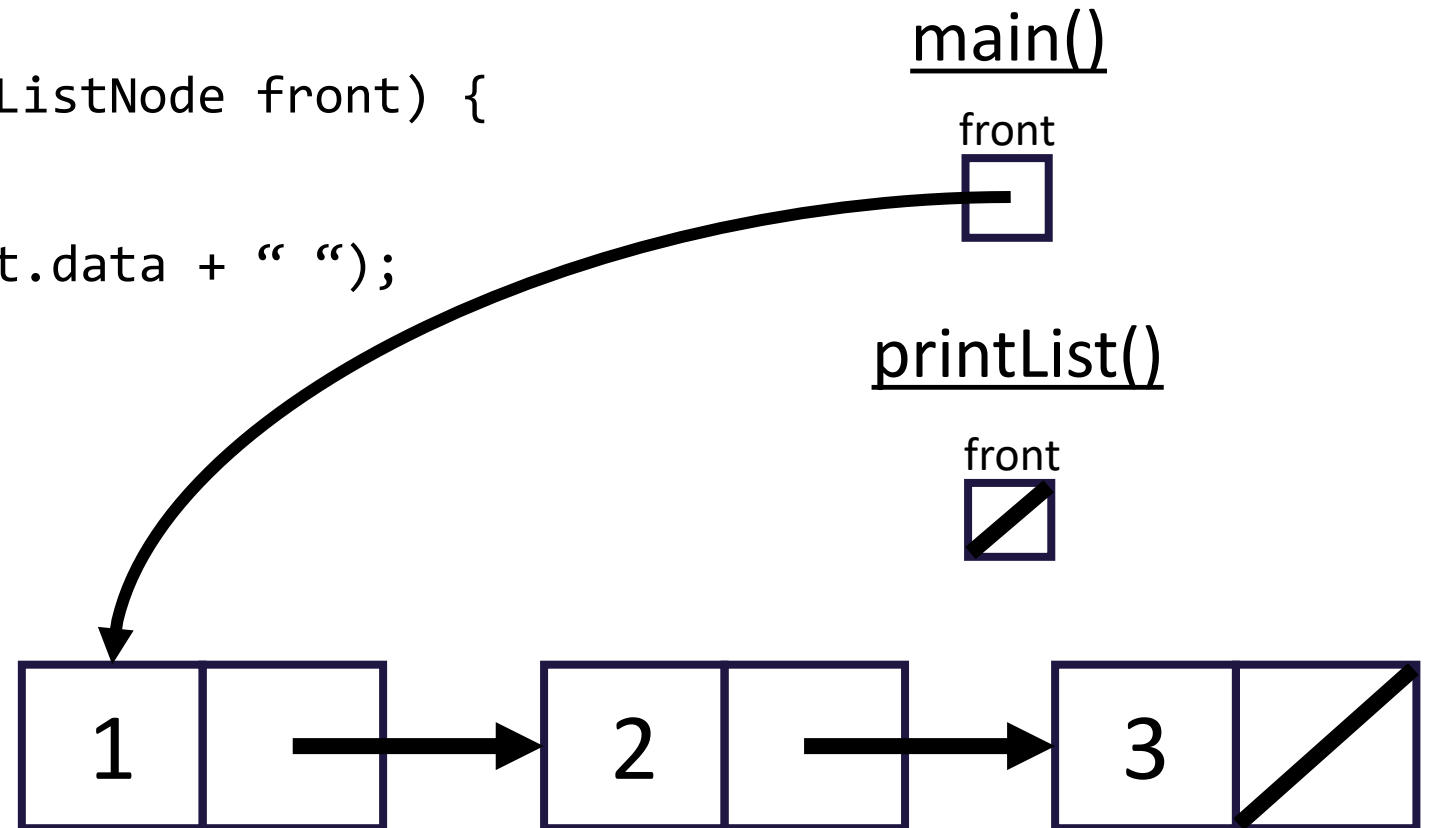
```
public static void printList(ListNode front) {  
    while (front != null) {  
        System.out.print(front.data + " ");  
        front = front.next;  
    }  
    System.out.println();  
}
```



Why curr?

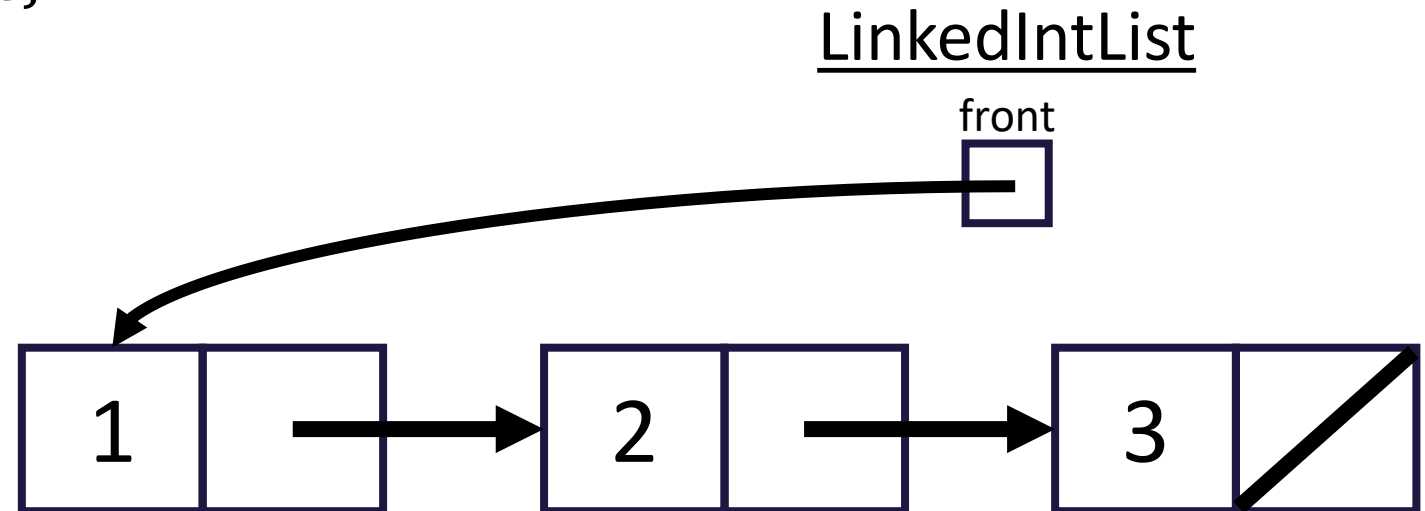
```
public static void main(String[] args) {  
    ListNode front = new ListNode(1, new ListNode(2, new ListNode(3)));  
}
```

```
public static void printList(ListNode front) {  
    while (front != null) {  
        System.out.print(front.data + " ");  
        front = front.next;  
    }  
    System.out.println();  
}
```



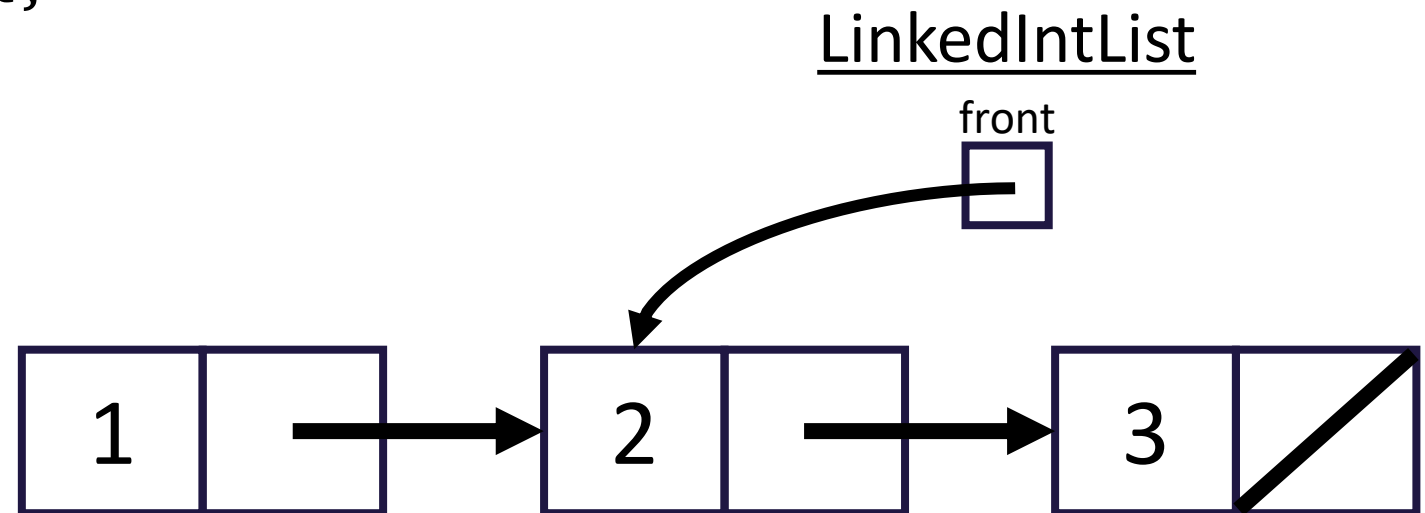
Why curr?

```
public class LinkedList {  
    private ListNode front;  
  
    public void printList() {  
        while (front != null) {  
            System.out.print(front.data + " ");  
            front = front.next;  
        }  
        System.out.println();  
    }  
}
```



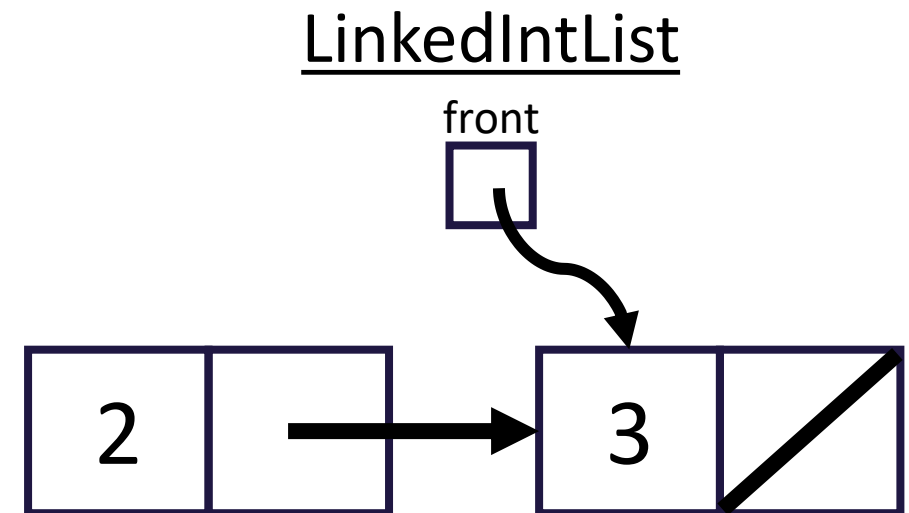
Why curr?

```
public class LinkedList {  
    private ListNode front;  
  
    public void printList() {  
        while (front != null) {  
            System.out.print(front.data + " ");  
            front = front.next;  
        }  
        System.out.println();  
    }  
}
```



Why curr?

```
public class LinkedList {  
    private ListNode front;  
  
    public void printList() {  
        while (front != null) {  
            System.out.print(front.data + " ");  
            front = front.next;  
        }  
        System.out.println();  
    }  
}
```



Why curr?

```
public class LinkedList {  
    private ListNode front;  
  
    public void printList() {  
        while (front != null) {  
            System.out.print(front.data + " ");  
            front = front.next;  
        }  
        System.out.println();  
    }  
}
```

LinkedList



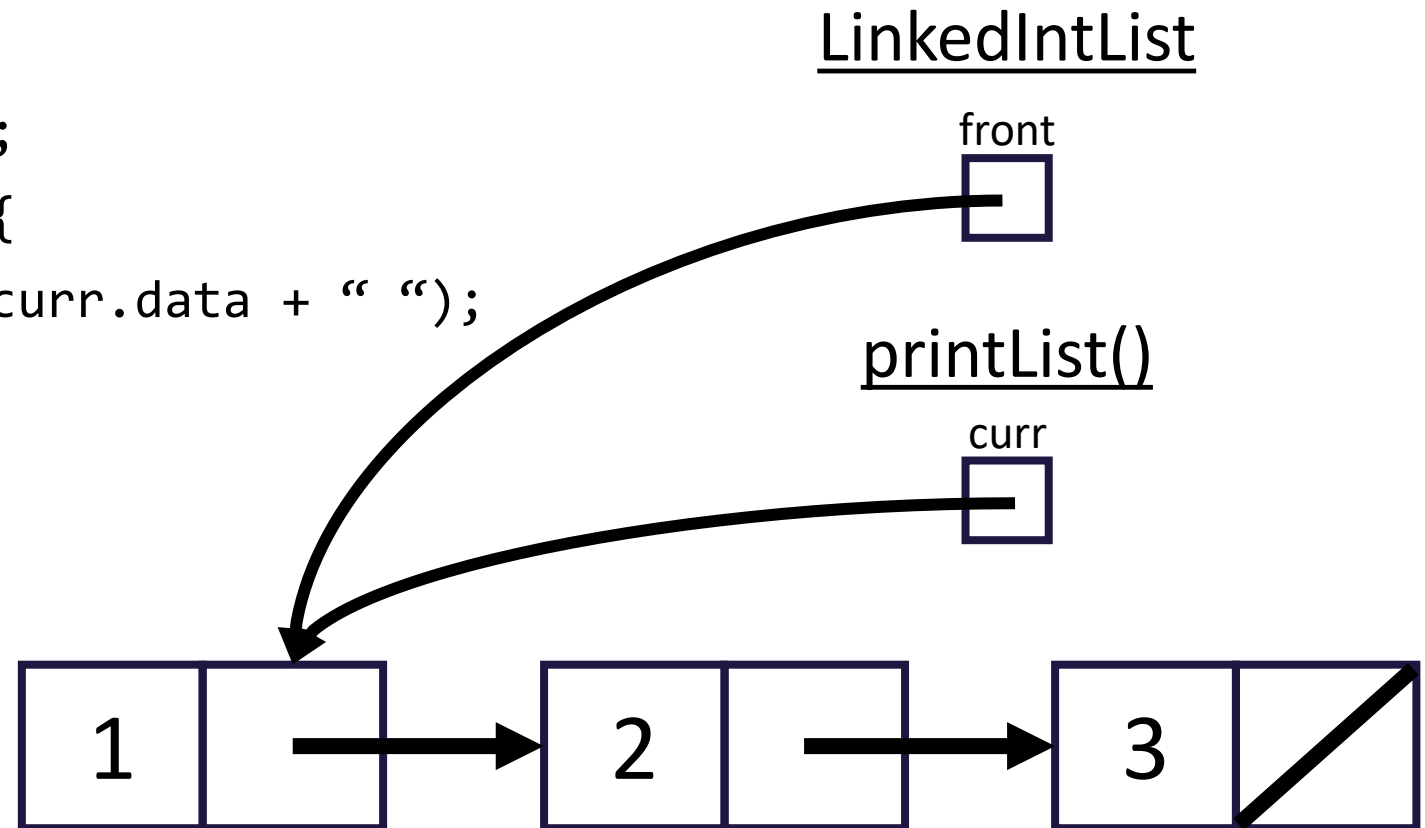
Modifying front now modifies the list!



Why curr?

```
public class LinkedList {  
    private ListNode front;
```

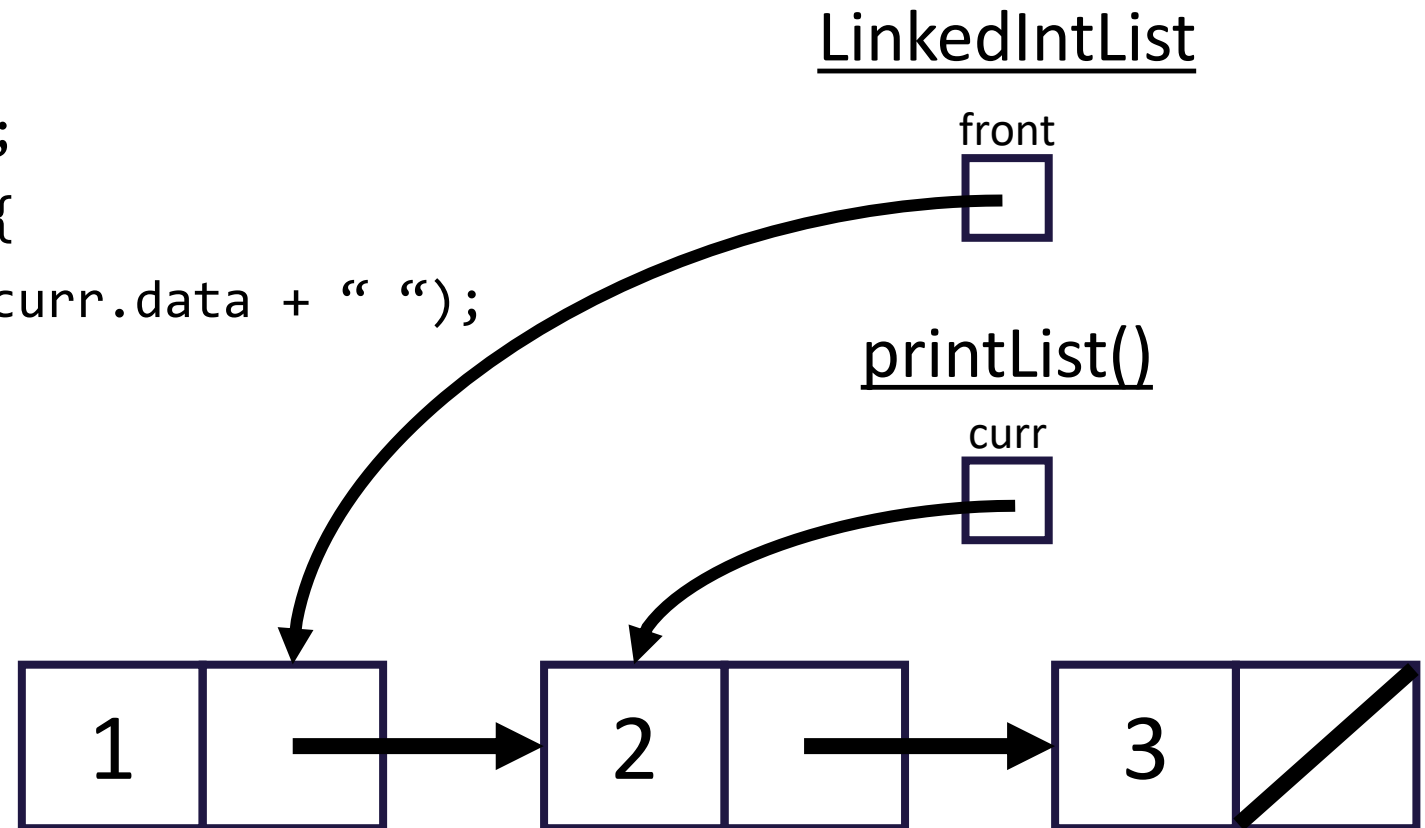
```
    public void printList() {  
        ListNode curr = front;  
        while (curr != null) {  
            System.out.print(curr.data + " ");  
            curr = curr.next;  
        }  
        System.out.println();  
    }  
}
```



Why curr?

```
public class LinkedList {  
    private ListNode front;
```

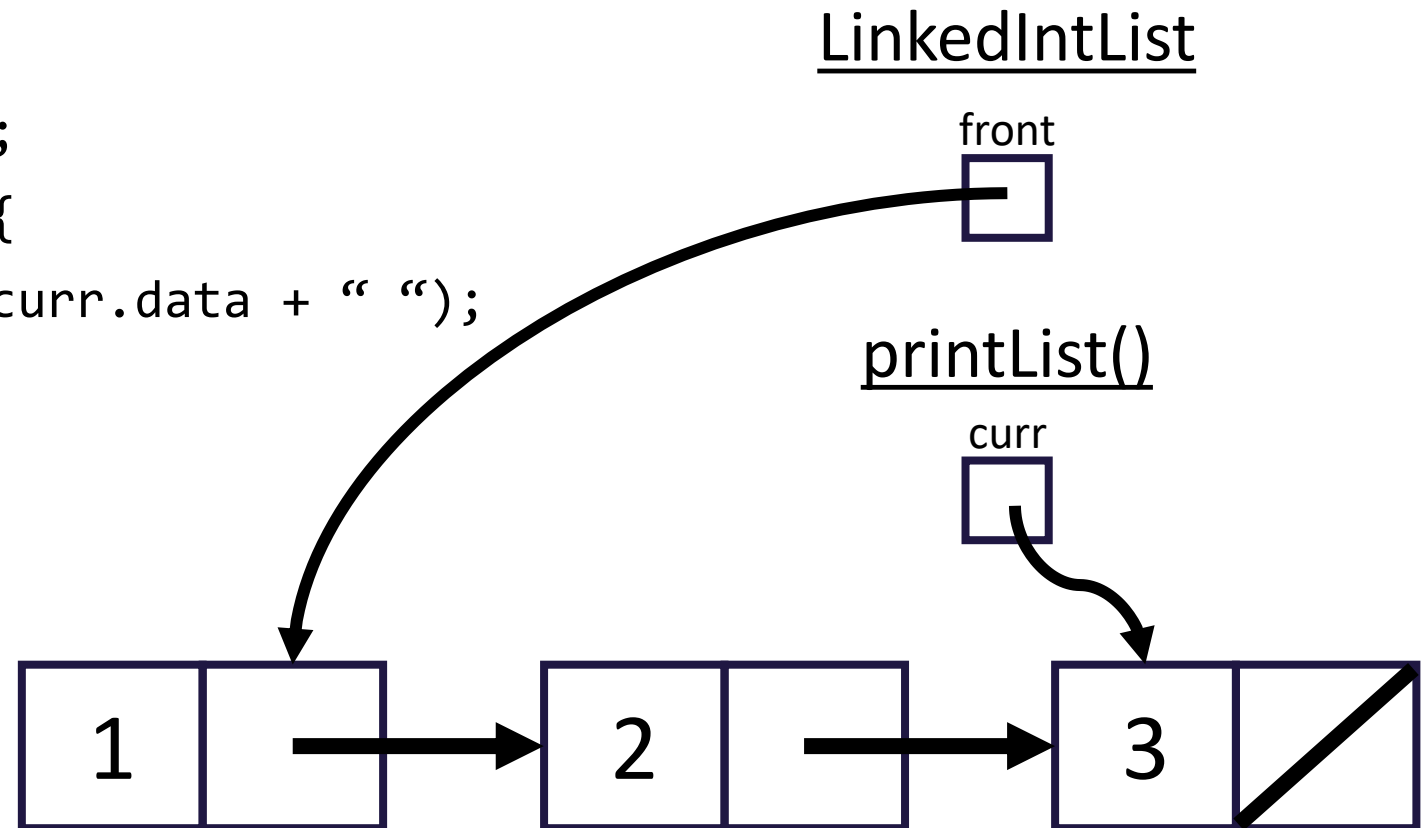
```
    public void printList() {  
        ListNode curr = front;  
        while (curr != null) {  
            System.out.print(curr.data + " ");  
            curr = curr.next;  
        }  
        System.out.println();  
    }  
}
```



Why curr?

```
public class LinkedList {  
    private ListNode front;
```

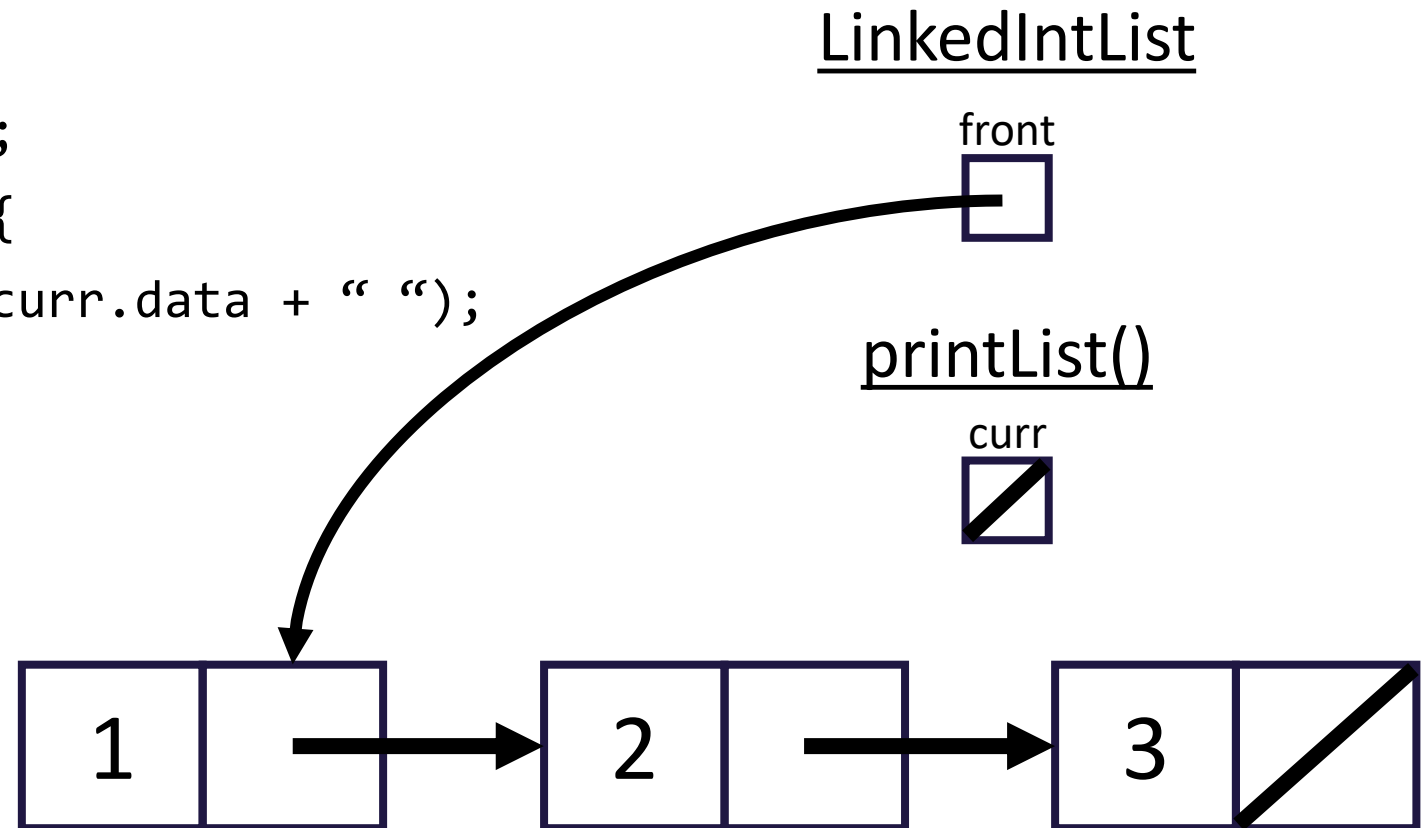
```
    public void printList() {  
        ListNode curr = front;  
        while (curr != null) {  
            System.out.print(curr.data + " ");  
            curr = curr.next;  
        }  
        System.out.println();  
    }  
}
```



Why curr?

```
public class LinkedList {  
    private ListNode front;
```

```
    public void printList() {  
        ListNode curr = front;  
        while (curr != null) {  
            System.out.print(curr.data + " ");  
            curr = curr.next;  
        }  
        System.out.println();  
    }  
}
```

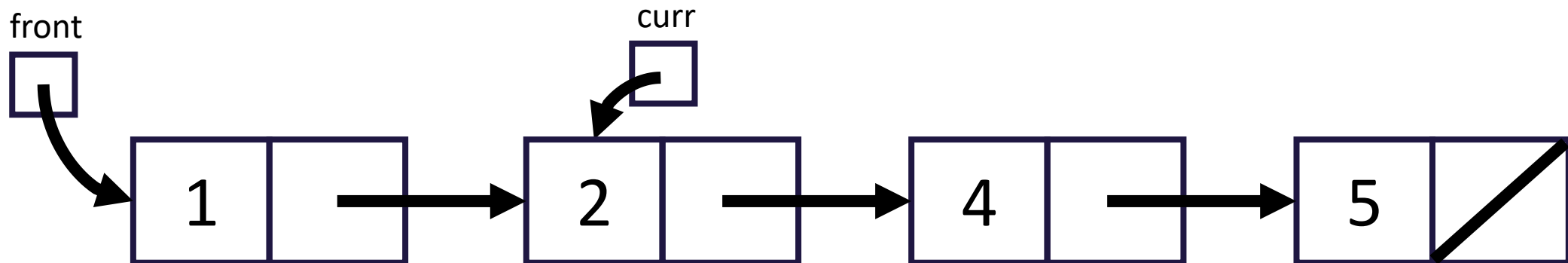


Lecture Outline

- Announcements/Reminders
- LinkedList
 - ListNodes cont.
- Why curr?
- **Modifying LinkedLists** 
 - Special cases (MFEE)

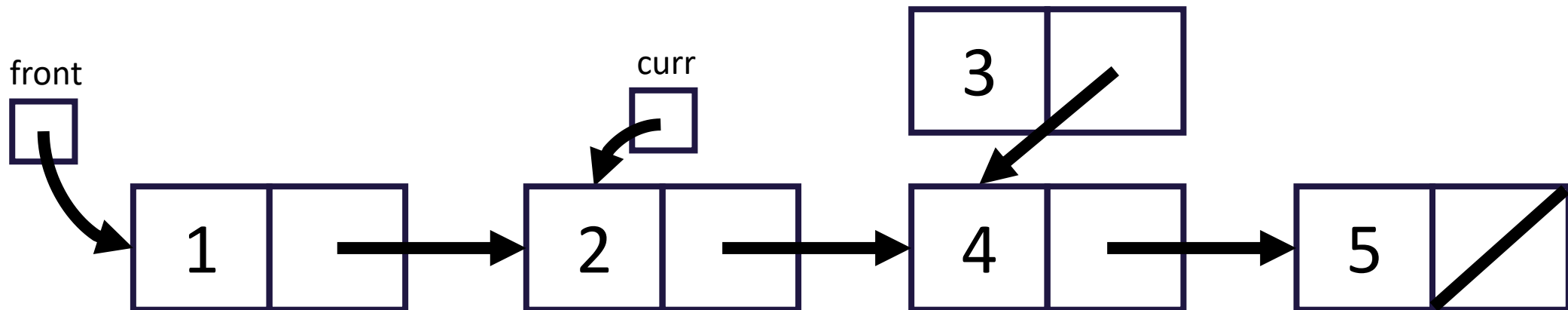
Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 3 node between 2 and 4
- Does changing curr actually update our chain?
 - What will?



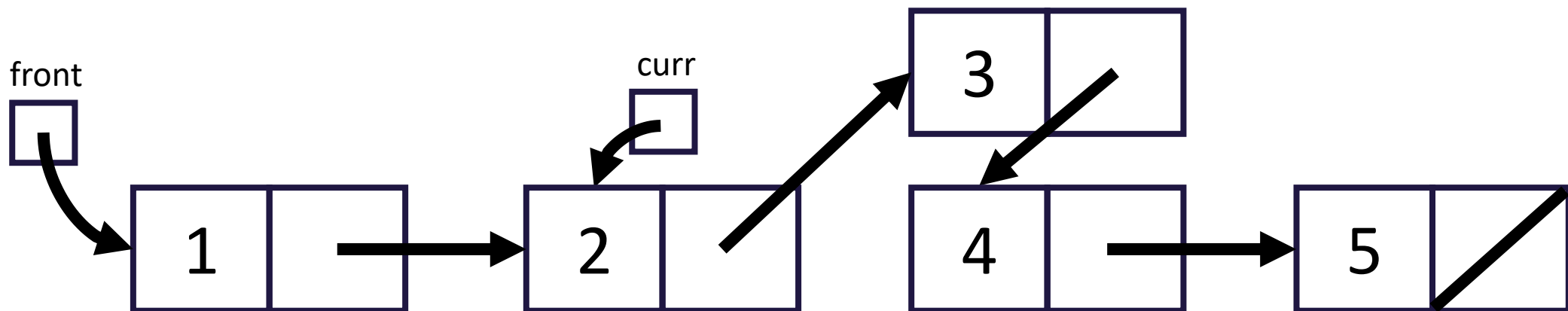
Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 3 node between 2 and 4
- Does changing curr actually update our chain?
 - 1. Make a new node storing 3 pointing to 4



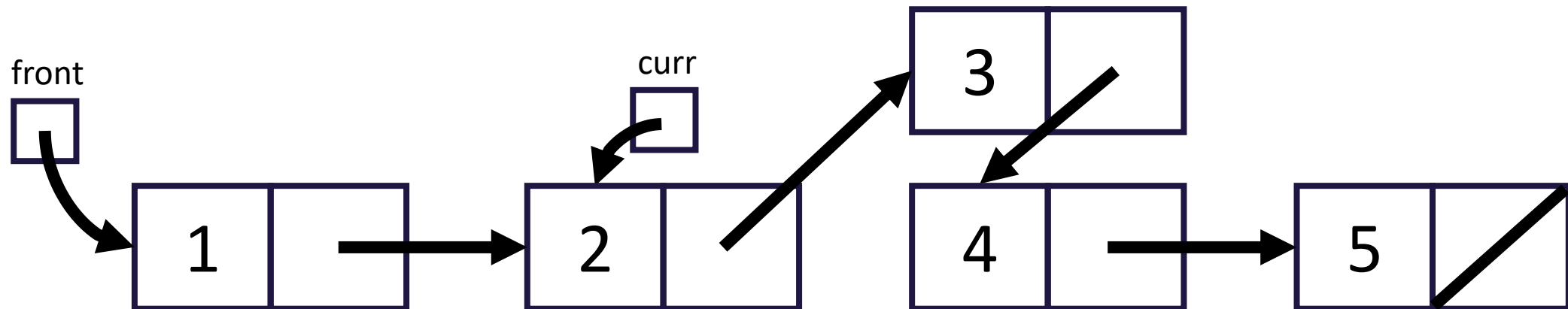
Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 3 node between 2 and 4
- Does changing curr actually update our chain?
 - 1. Make a new node storing 3 pointing to 4
 - 2. Make 2 point to 3



Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 3 node between 2 and 4
- Does changing curr actually update our chain?
 - `curr.next = new ListNode(3, curr.next);`

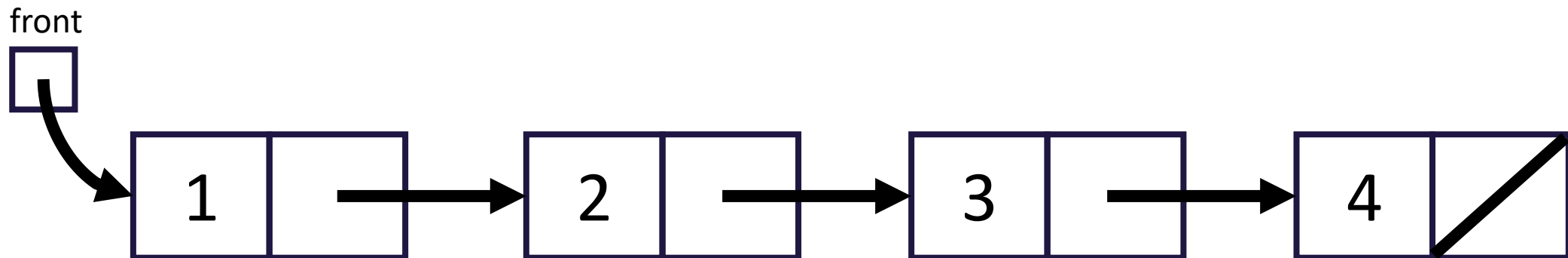


Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 3 node between 2 and 4
- Does changing curr actually update our chain?
 - What will? Changing curr.next

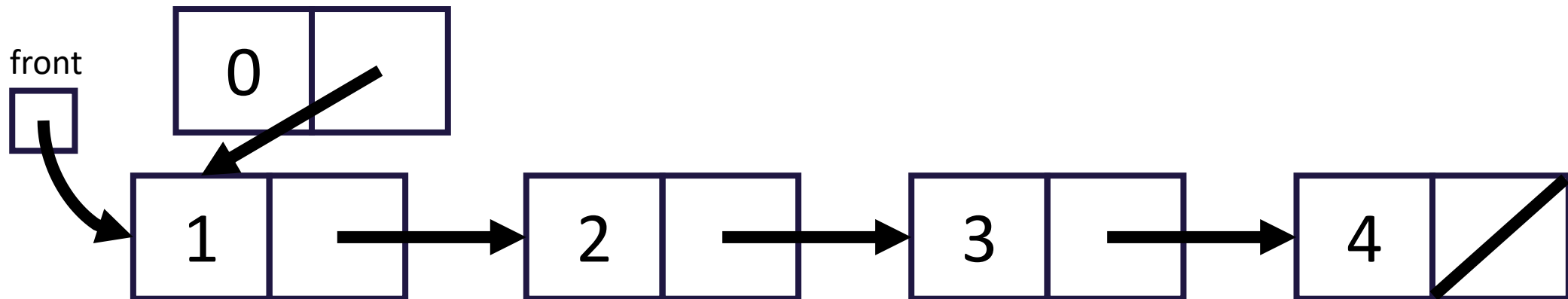
Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 0 node before 1
- Is there anyway for us to do this with curr?
 - No!



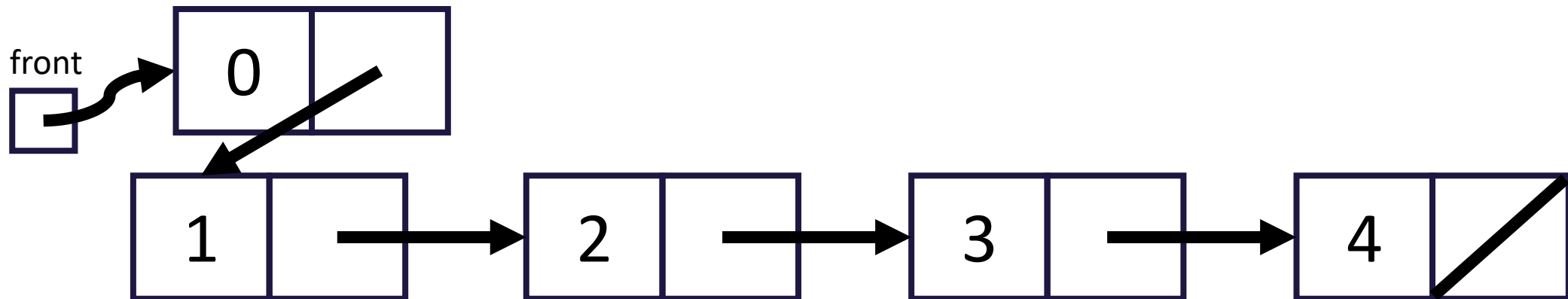
Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 0 node before 1
- Is there anyway for us to do this with curr?
 - 1. Make a new node storing 0 pointing to 1



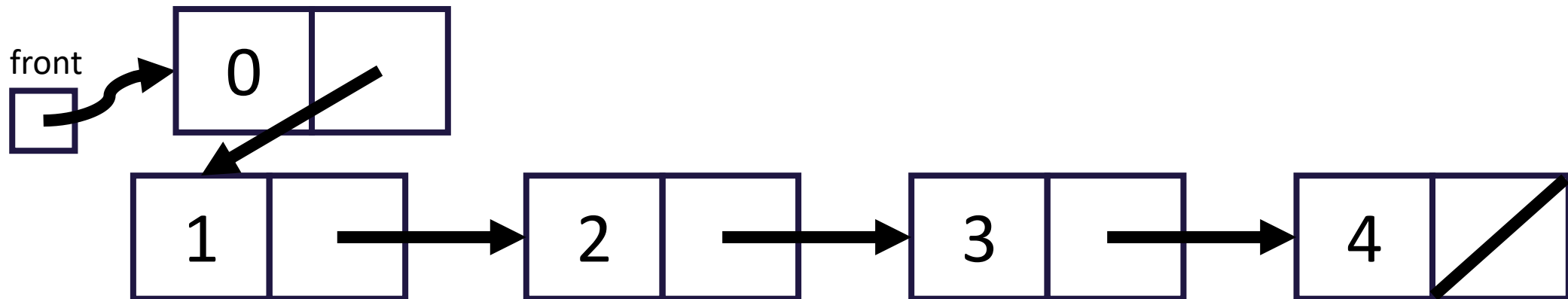
Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 0 node before 1
- Is there anyway for us to do this with curr?
 - 1. Make a new node storing 0 pointing to 1
 - 2. Make front point to 0



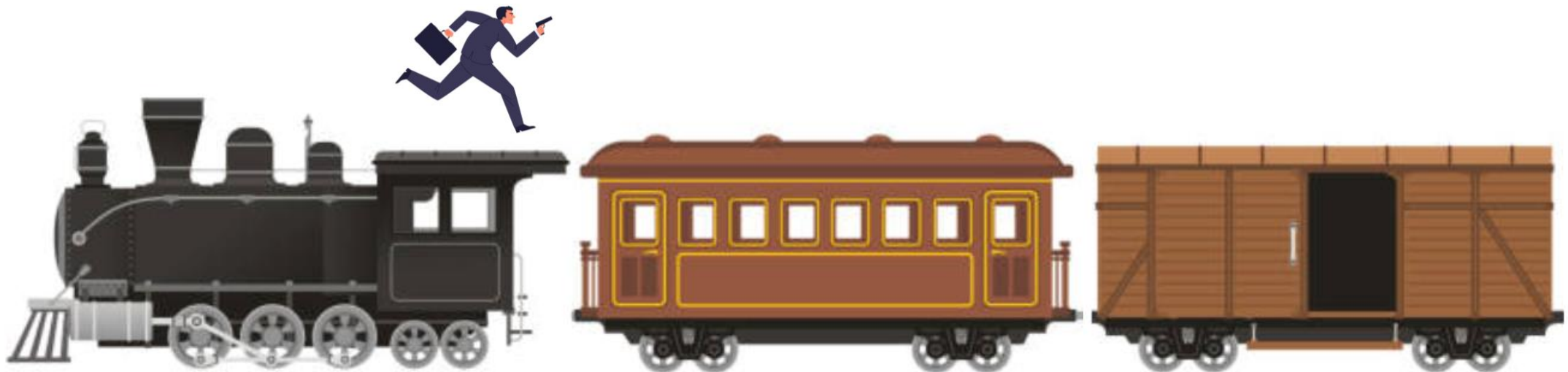
Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 0 node before 1
- Is there anyway for us to do this with curr?
 - `this.front = new ListNode(0, this.front);`



Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 0 node before 1
- So, what will actually change our list?
 - Changing curr.next, changing front
 - Need to **stop one early** to make changes



Modifying LinkedLists

- Remember: using a curr variable to iterate over nodes
- We want to insert a 3 node between 2 and 4
- Does changing curr actually update our chain?
 - What will? Changing curr .next, changing front
 - Need to **stop one early** to make changes
- Often a number of cases to watch out for:
 - M(iddle) – Modifying node in the middle of the list (general)
 - F(ront) – Modifying the first node
 - E(mpty) – What if the list is empty?
 - E(nd) – Rare, do we need to do something with the end of the list?