CompSci 201, L11: Linked List and Pointer Problems

Logistics, Coming up

- Monday, 10/3 (today)
  - Project 2: Markov Due
  - Project 3: DNA (Linked List) releases tomorrow, due 10/17
- Wednesday, 10/5
  - APT 5 Due
- Friday, 10/7
  - Discussion, linked list
- Monday 10/10 – Tuesday 10/11
  - Fall break, no class meeting, no helper/office hours

Outline

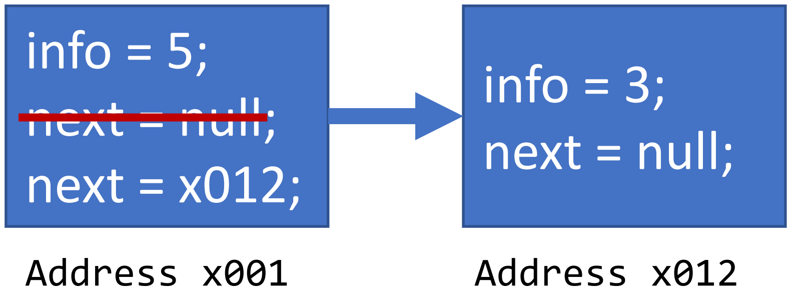
- Part 1: Implementing DIYLinkedList
- Part 2: Working directly with List Node objects, algorithmic problem-solving
    1. Get to index'th node
    2. Append one list to another
    3. Reverse a list in place

Linked list is a list implemented by linked nodes. What is a node?

- Just a Java object of a class we write, like any other!
- We want to "link" them together, so each node has a *reference* (~pointer, a memory location) to another node.

```
public class ListNode {
    int info;
    ListNode next;
    ListNode(int x){
        info = x;
    }
    ListNode(int x, ListNode node){
        info = x;
        next = node;
    }
}
```

```
ListNode first = new ListNode(5);
ListNode second = new ListNode(3);
first.next = second;
```



Creating Nodes, constructing lists

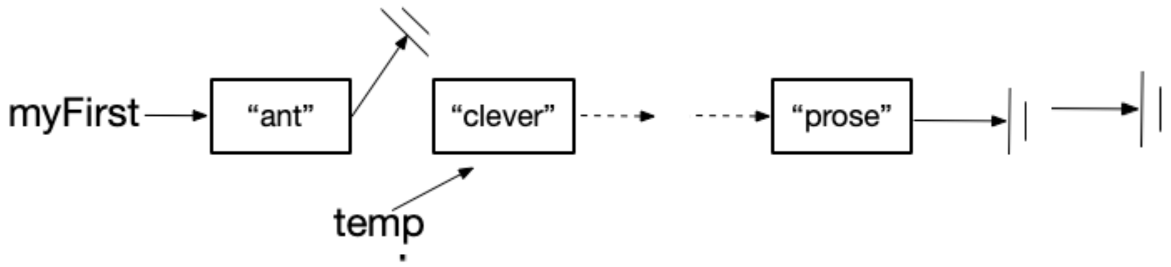
1. Calling `new Node(…)` always creates a Node in memory that did not exist before

2. Writing `node.next = otherNode;` makes node "→" `otherNode`

3. `node.next` or `node.info` gives an error (null pointer exception) if node is `null`

Why add and remove at front are O(1)

- How to remove first node?



```
public void removeFirst() {
    Node temp = myFirst.next;
    myFirst.next = null;
    myFirst = temp;
}
```

DIYLinkedList

Live Coding

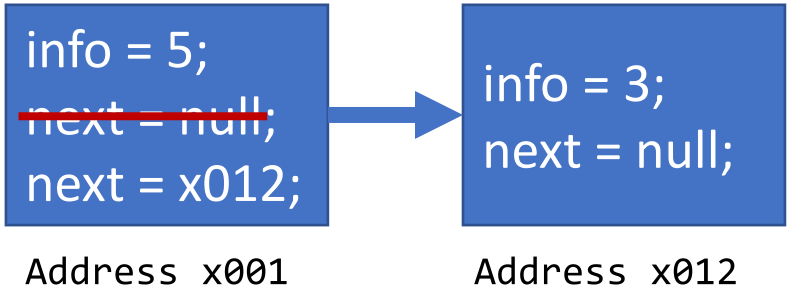
Part 2: Working Directly with List Node objects, algorithmic problem-solving

Linked list is a list implemented by linked nodes. What is a node?

- Just a Java object of a class we write, like any other!
- We want to "link" them together, so each node has a *reference* (~pointer, a memory location) to another node.

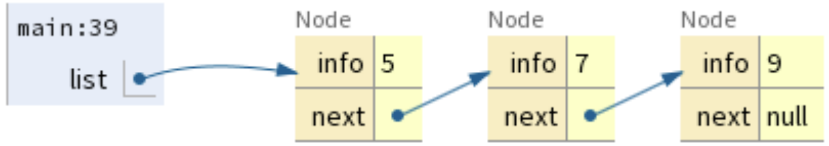
```
public class ListNode {
    int info;
    ListNode next;
    ListNode(int x){
        info = x;
    }
    ListNode(int x, ListNode node){
        info = x;
        next = node;
    }
}
```

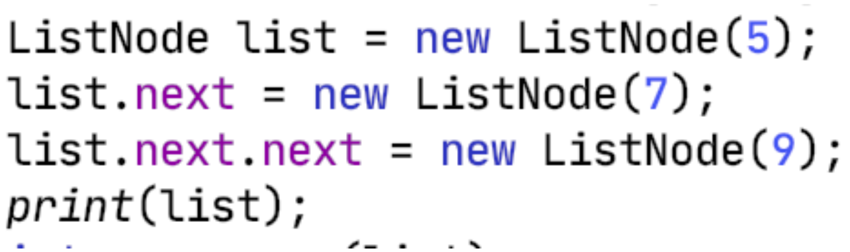
```
ListNode first = new ListNode(5);
ListNode second = new ListNode(3);
first.next = second;
```



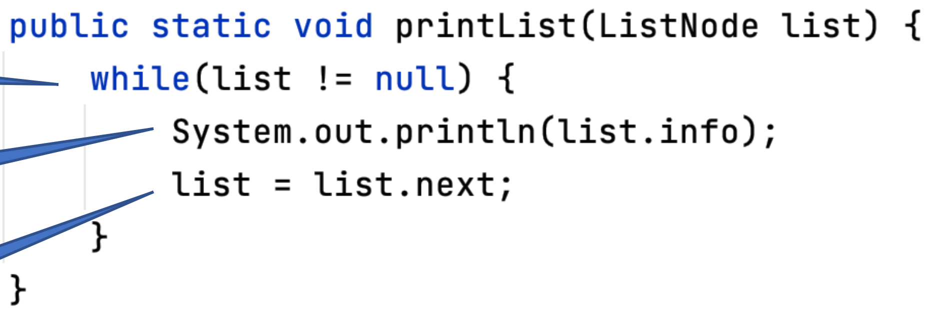
Creating and traversing a linked list

- `ListNode` class used in APTs, etc.
  - The variable for the "linked list itself" is just a reference to the first `ListNode`





```
ListNode list = new ListNode(5);
list.next = new ListNode(7);
list.next.next = new ListNode(9);
print(list);
```



```
public static void printList(ListNode list) {
    while(list != null) {
        System.out.println(list.info);
        list = list.next;
    }
}
```

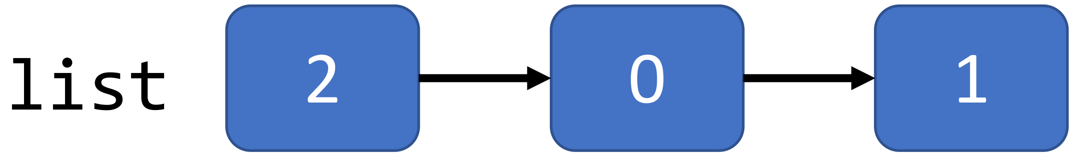
While there is a next node...

Print value of current node

Go to next node

“get(index)” for low level linked list?

Given a linked list of `ListNode` objects (call it `list`), and an integer `index`, return the `info` of the `index`’th node?



Need to use next index times?

In the above example…

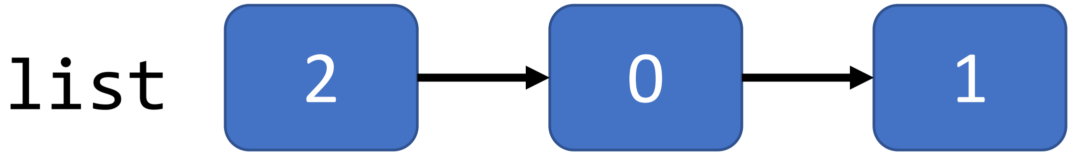
- get(0) should return 2 `list.info`
- get(1) should return 0 `list.next.info`
- get(2) should return 1 `list.next.next.info`

First attempt: for loop

```java
public static int get(int index, ListNode list) {
    for (int i=0; i<index; i++) {
        list = list.next;
    }
    return list.info;
}
```

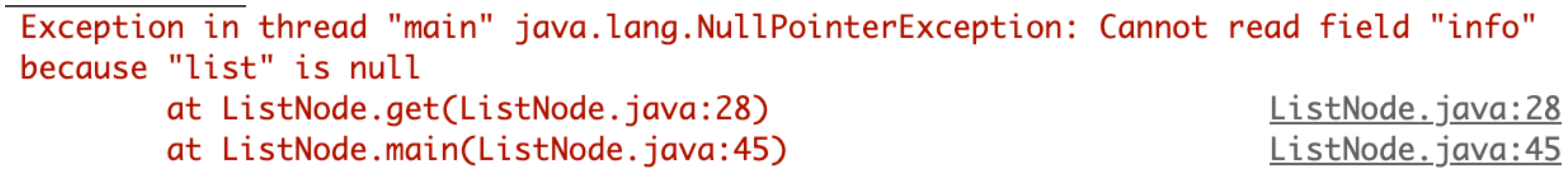
Advances list reference index times

What if this is called with index > the number of nodes in list?



Then get(3, list) is…error? Null pointer exception!

Reminder: What is a null pointer exception?



```
Exception in thread "main" java.lang.NullPointerException: Cannot read field "info"
because "list" is null
        at ListNode.get(ListNode.java:28)                                    ListNode.java:28
        at ListNode.main(ListNode.java:45)                                   ListNode.java:45
```

- null: The reserved keyword for an uninitialized object.
- Has no instance variables, attributes, methods, etc.
- Trying to call `.<anything>` on a null reference generates a null pointer exception.

Second attempt

Instead of a null pointer, would be nice to recognize if the index is out of bounds…but how many nodes are in the list?

```
public static int get(int index, ListNode list) {
    int i=0;
    while ((list != null) && (i < index)) {
        list = list.next;
        i++;
    }
    if (list == null) {
        throw new IndexOutOfBoundsException();
    }
    return list.info;
}
```

More informative error message

WOTO

Go to duke.is/v74au

Not graded for correctness, just participation.

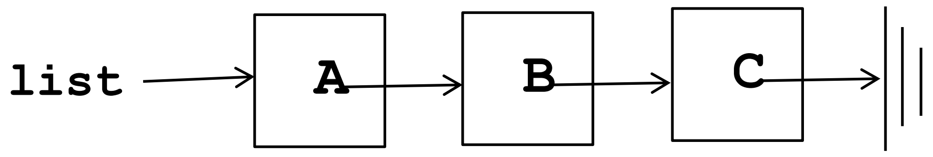
Try to answer *without* looking back at slides and notes.

But do talk to your neighbors!

ListNode Pointer Problems

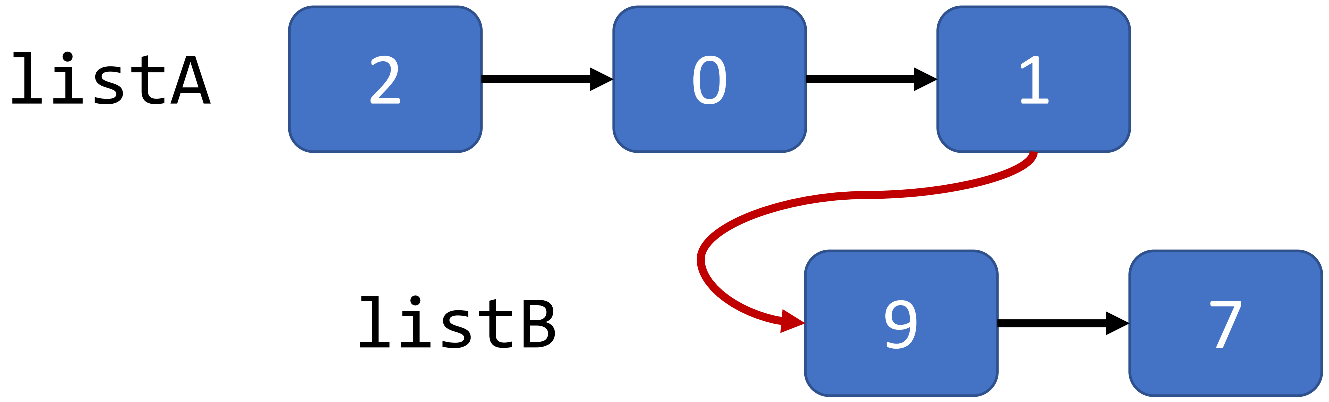
Drawing Pictures

- Visualization is very important: Draw pictures!
  - Try your algorithm/code one step at a time with:
    - 0 nodes
    - 1 node
    - 2 nodes
    - 3 nodes
  - Check boundary conditions
  - Is this pointing to what I think it's pointing to? Check!



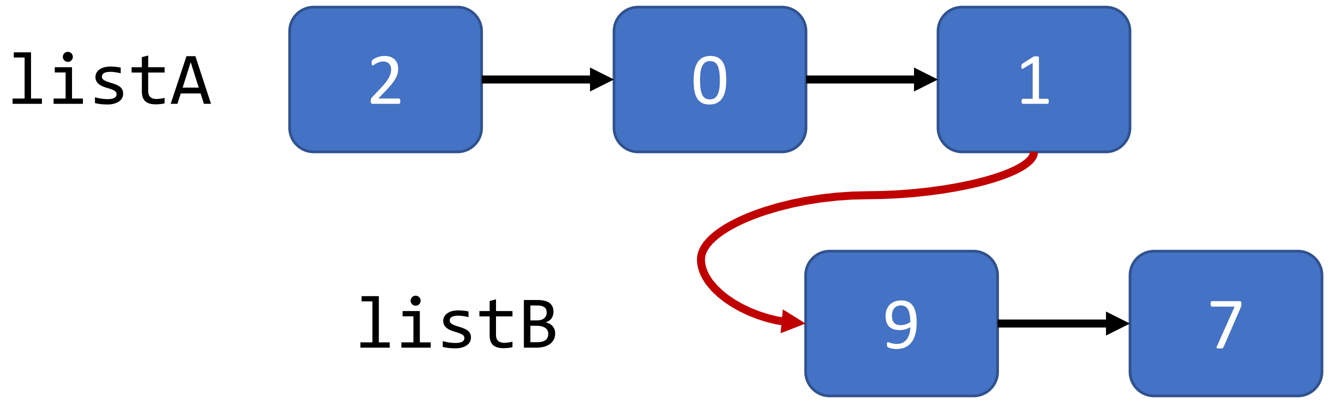
Append linked lists of ListNodes

- Append `listB` to `listA` using…
  - O(1) additional memory,
  - No copying values,
  - Just changing pointers in the input lists.



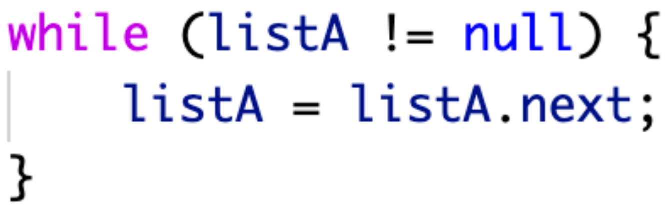
Append linked lists of ListNodes

- Conceptual algorithmic questions:
  - How to get a reference to the *last* node of `listA`?
  - How to update last node to point to the first node of `listB`?
  - What to return?



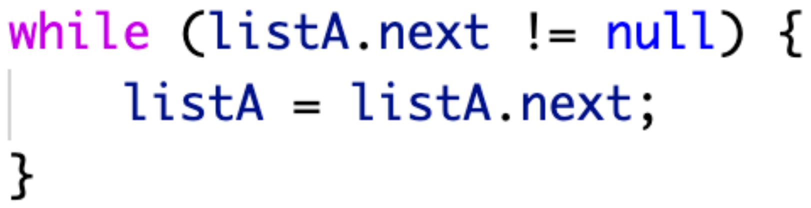
How to get a reference to the last node?

Starting with the standard list traversal idiom we know…



```
while (listA != null) {
    listA = listA.next;
}
```

But after exiting this loop, `listA` is just null. Stop one node before…

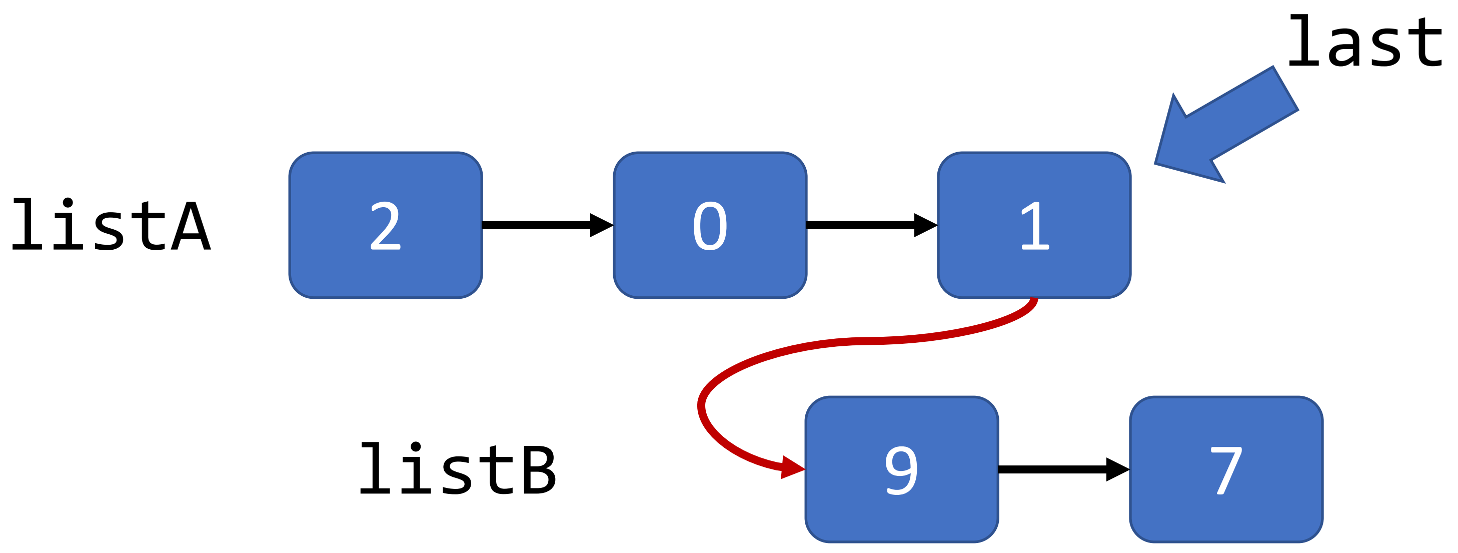


```
while (listA.next != null) {
    listA = listA.next;
}
```

How to update last node to point to the first node of listB?

Recall: Writing `node.next = otherNode;` makes node → (point to) otherNode.

```
last.next = listB;
```

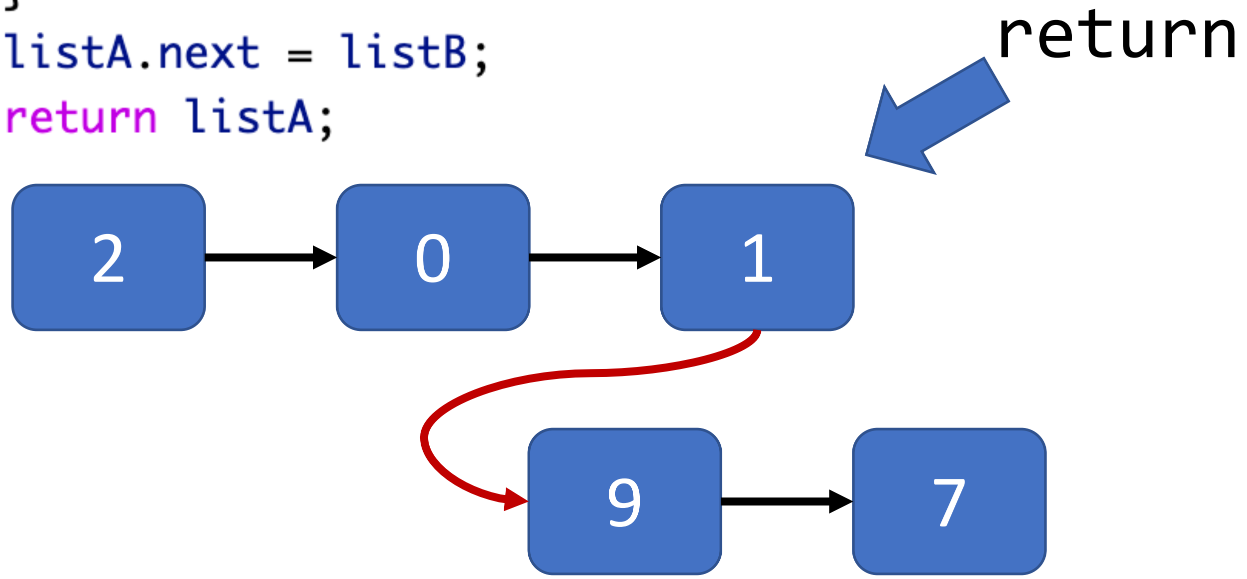


What to return?

If `listB` is appended *to the end* of `listA`, need to return a reference to the first node of `listA`.

```
while (listA.next != null) {
    listA = listA.next;
}
listA.next = listB;
return listA;
```

Correctly changes list in memory, but returns reference to middle



Append linked lists of ListNodes: Putting it all together

```java
public static ListNode append(ListNode listA, ListNode listB) {
    ListNode first = listA;
    while (listA.next != null) {
        listA = listA.next;
    }
    listA.next = listB;
    return first;
}
```

- Reminding again: Accomplished with O(1) additional memory and without copying any values.
- Not necessarily a lot of lines of code, but…
- easy to get lost without planning and visualization before/while coding.

WOTO

Go to duke.is/zsttj

Not graded for correctness, just participation.

Try to answer *without* looking back at slides and notes.

But do talk to your neighbors!