CS 106 INTRODUCTION TO DATA STRUCTURES

SPRING 2020

PROF. SARA MATHIESON

HAVERFORD COLLEGE

Slides modified from Sorelle Friedler

ADMIN

- Lab 2 due TOMORROW (Wed night)
- Lab 3 posted tomorrow, due next Sunday
- Before lab this week, begin next pre-lab (okay to finish in lab)

- TA hours tonight 7-9pm in H110
- No office hours today (they were yesterday)

ADMIN

All scheduling info is on the google calendar!

FEB 18 OUTLINE

- Introduce Lab 3
- Recap Singly Linked Lists
- Motivation for Doubly Linked Lists (sorting)
- Doubly Linked Lists
- Equality and extras

FEB 18 OUTLINE

- **Introduce Lab 3**
- Recap Singly Linked Lists
- Motivation for Doubly Linked Lists (sorting)
- Doubly Linked Lists
- Equality and extras

LAB 3: BABY NAMES

- **Goal:** create sorted lists of baby names using your own custom Nodes and Linked List class
- **Practice:** using command line arguments to let the user query their favorite names
- **Return:** information about the popularity of baby names in the U.S. over time
- **Note:** very gendered way of viewing this historical data, how could we change the way data is collected in the future?

Year: 2002

```
1,Jacob,30568,Emily,24463
2,Michael,28246,Madison,21773
3,Joshua,25986,Hannah,18819
4,Matthew,25151,Emma,16538
5,Ethan,22108,Alexis,15636
...
996,Ean,157,Johana,221
997,Jovanni,157,Juana,221
998,Alton,156,Juanita,221
999,Gerard,156,Katerina,221
1000,Keandre,156,Amiya,220
```

HANDLING INPUT

Where input goes when it is given to a Java program. Each space separated word becomes a different string in the args array.

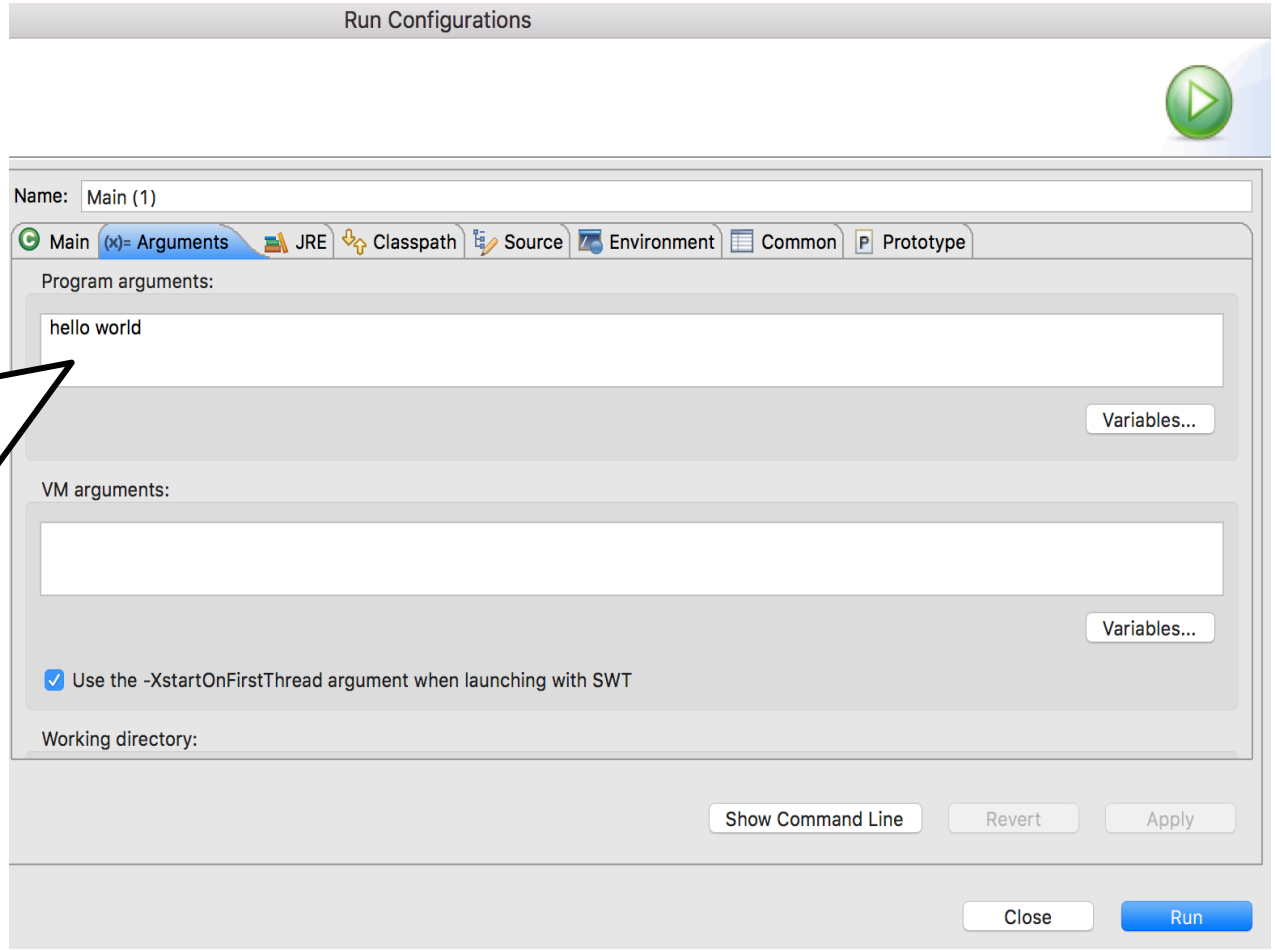
```
public class Main {
    public static void main(String[] args) {
        // This is where you can parse the
        // command line arguments
    }
}
```

```
java Main arg0 arg1
```

What this looks like from the command line when the program is called with arguments.

HANDLING INPUT IN ECLIPSE

Go to Run Configurations and choose Arguments:



The space-separated arguments that are given here are then passed into the main function as String[] args

FEB 18 OUTLINE

- Introduce Lab 3
- **Recap Singly Linked Lists**

Handout 9, hand back Handout 8

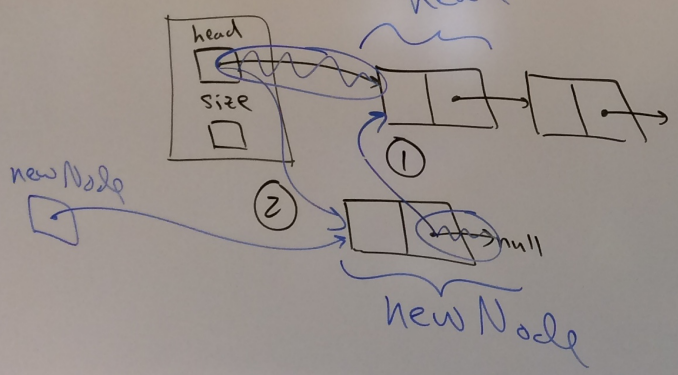
- Motivation for Doubly Linked Lists (sorting)
- Doubly Linked Lists
- Equality and extras

Handout 9

① Number of Courses: 3

CS 105
CS 106
Math 102

② Linked List



addFirst

Node<E> newNode = new Node<E>(newData);

newNode.setNext(head);

head = newNode;

size += 1;

Constant: O(1)

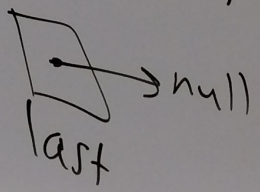
we will write (for Lab 3 too)

③ addLast: O(n) (linear)

with tail pointer: O(1)

④ get(i): O(n) (linear)

⑤ while (currNode.next() == null



JAVA NOTES

Each public class should be in its own file with the same name. The basic structure of a Java program looks like:

```
public class Name {
    private int instanceVariable;

    public Name(int instanceVar) {
        this.instanceVariable = instanceVar;
    }

    public int getVariable() {
        return this.instanceVariable;
    }
}
```

Instance variables should be private and at the top of the class file.

The constructor should have the same name as the class and should initialize the instance variables.

Note that all lines of code are either variable declarations or inside methods.

Methods, including getters and setters, should follow the constructor.

JAVA CONSTRUCTORS AND INSTANCE VARIABLES

```
public class Demo {
    private int var;

    public Demo() {
        this.var = 0;
    }

    public Demo(int var) {
        this.var = var;
    }
}
```

(Non-static) instance variables are different for each instance (object) of the class type and are the main way to store data and meaning using a class.

The constructor should initialize all instance variables.

You can have more than one constructor for a single class as long as they take different arguments!

JAVA VARIABLE DECLARATIONS

```
public class Demo {
    private int var;

    public Demo() {
        this.var = 0;
    }

    public void badFunction() {
        int var = 5;
    }
}
```

Variables are always **declared** (created) by indicating a type and a name.

Variables are **initialized** by setting the right hand side of the equals sign as the value of the variable.

Warning: Putting a type in front of the variable name creates a **new** variable with that name.

FEB 18 OUTLINE

- Introduce Lab 3
- Recap Singly Linked Lists
- **Motivation for Doubly Linked Lists (sorting)**
- Doubly Linked Lists
- Equality and extras

Doubly Linked List (motivation: sorting)

Example: {9, 5, 1, 7, 3, 8} Goal: sort

Linked List

↪ n = 6

1 {9}

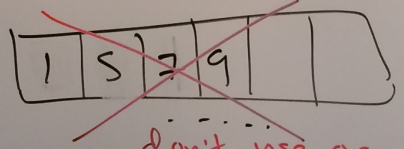
2 {5, 9}

3 {1, 5, 9}

4 {1, 5, 7, 9}

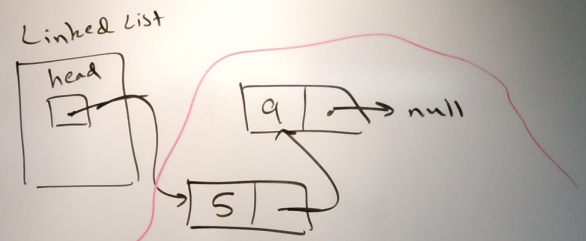
5 {1, 3, 5, 7, 9}

6 {1, 3, 5, 7, 8, 9}



don't use array!

insertion Sort

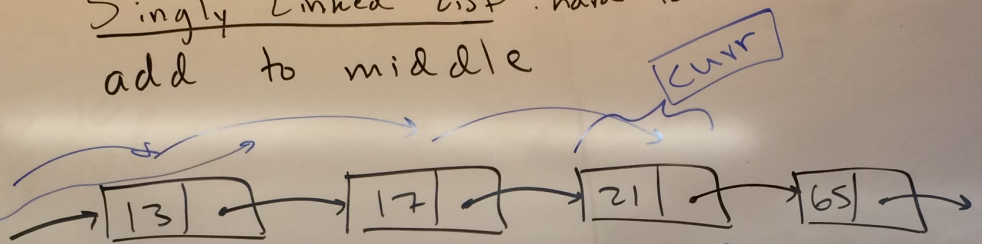


FEB 18 OUTLINE

- Introduce Lab 3
- Recap Singly Linked Lists
- Motivation for Doubly Linked Lists (sorting)
- **Doubly Linked Lists**
- Equality and extras

Singly Linked List: hard to add to middle

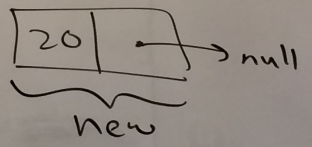
new Data = 20



prev = curr

curr = curr.next

"walking"

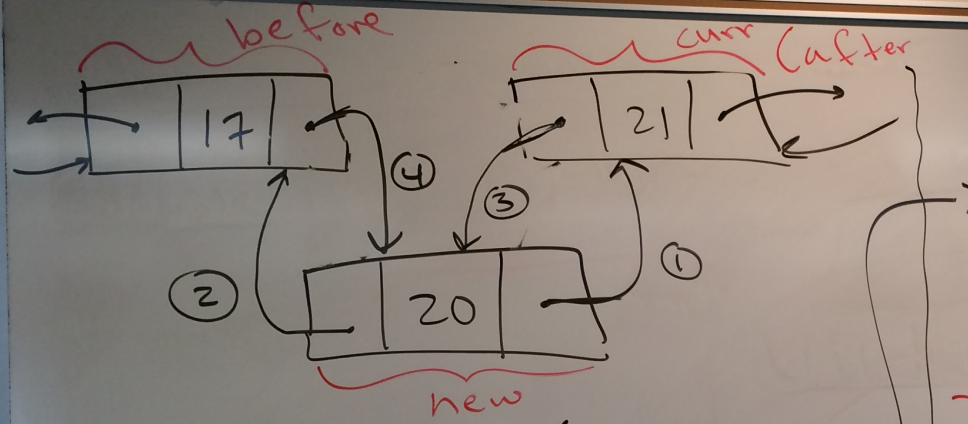


new.next = curr

Solution: Doubly Linked Lists →

code: constructor

```
public Node(Data d){
    data = d;
    prev = null;
    next = null;
}
```



add Between: (4 lines)

1. new.next = curr
2. new.prev = curr.prev
3. curr.prev = new
4. new.prev.next = new

before = curr.prev

Sentinels

before.next = new

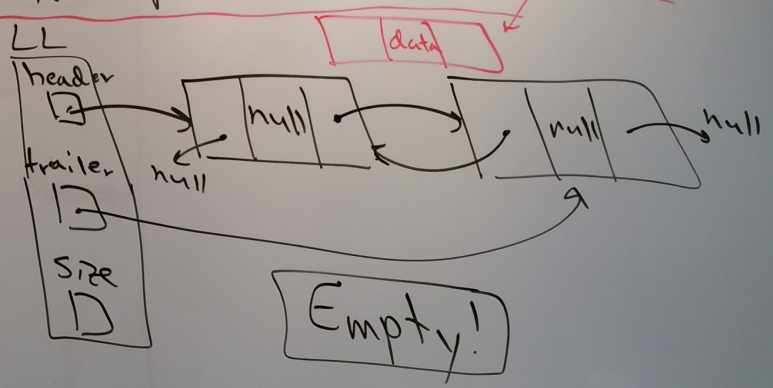
~~curr~~.prev = new

after

new.next = ~~curr~~ after

new.prev = before

first element



Constructor:

header = Node(null)

trailer = Node(null)

~~head = null~~

~~tail = null~~

LINKED LIST VARIATIONS

Suppose we know that a data structure will be used for a specific task – we can make modifications to linked lists to make them better for that task!

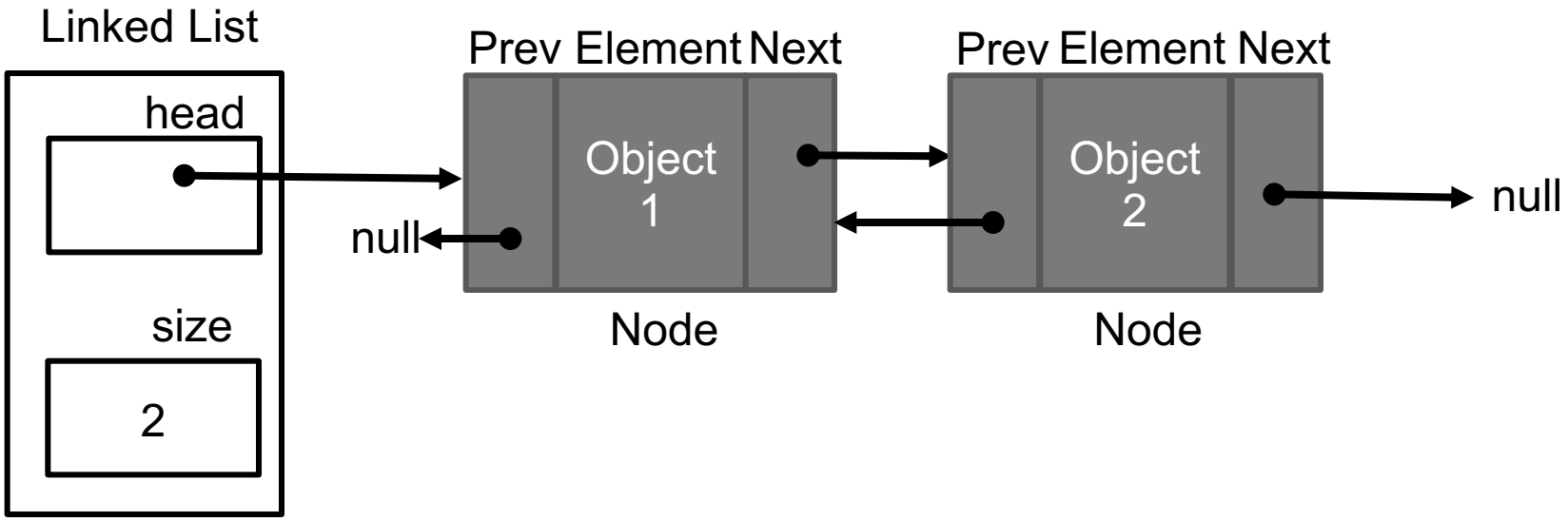
Last time:
adding a tail pointer to make adding to the end of the list fast

Note:

There are many, many ways to implement and customize linked lists!

DOUBLY LINKED LISTS

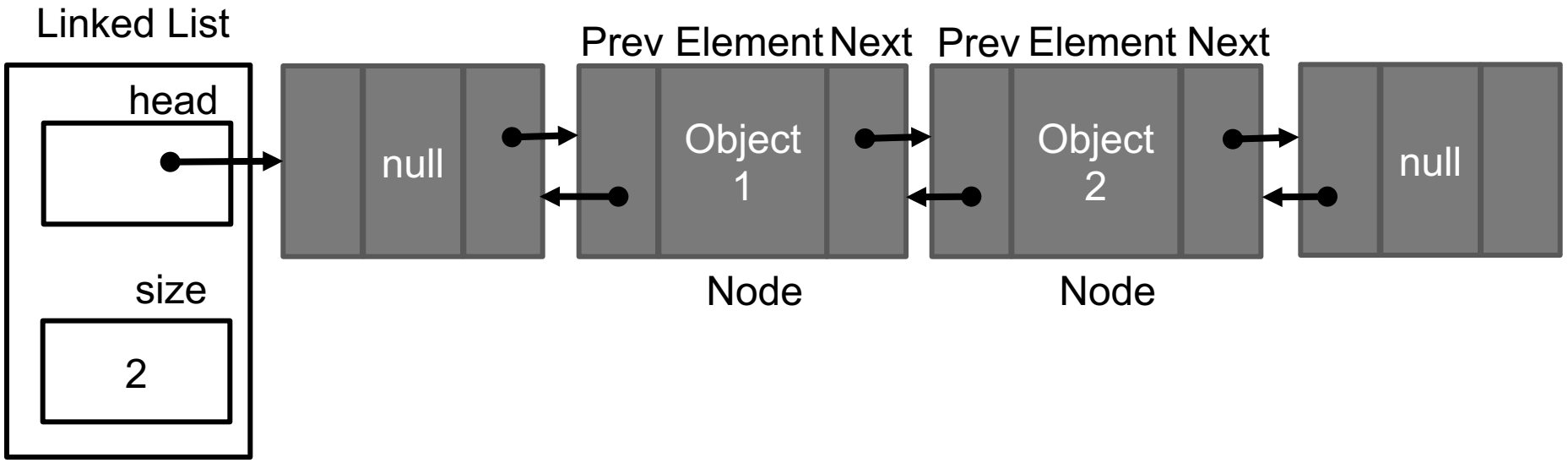
What if we want to be able to add / remove nodes in the middle of the list more easily?



Want more info? See book section 3.4.

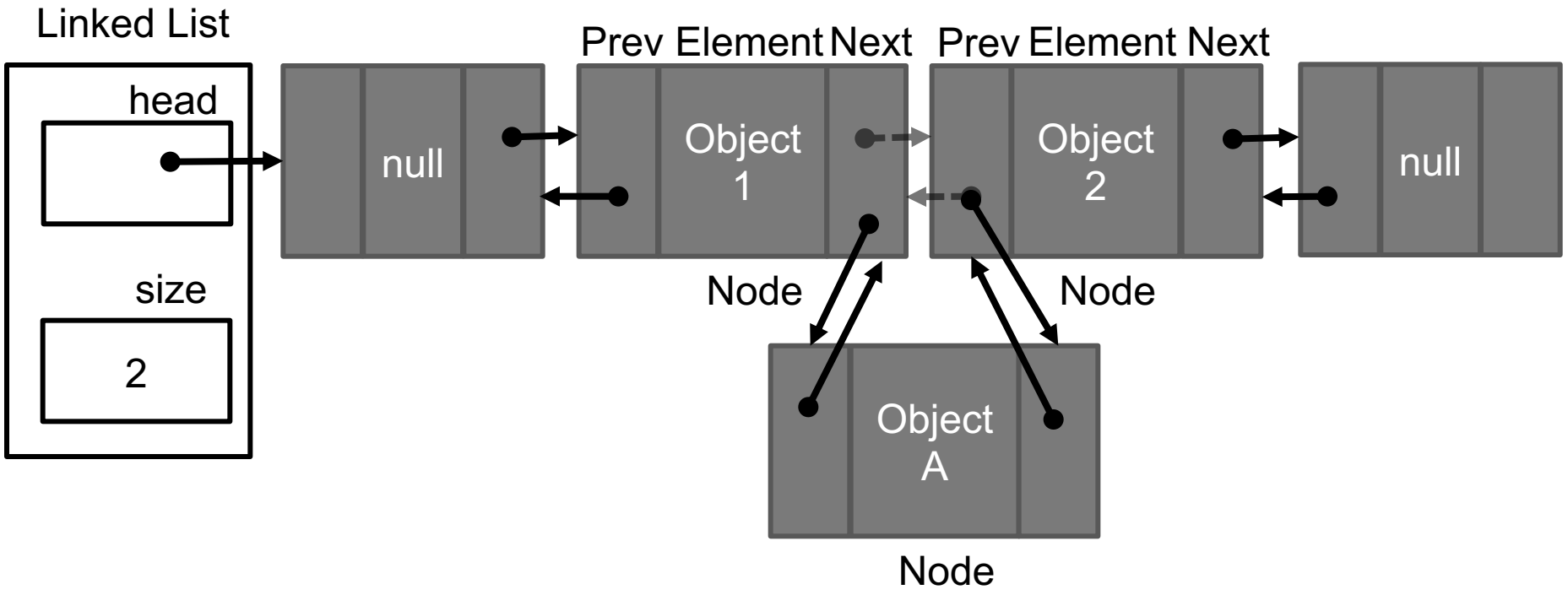
DOUBLY LINKED LISTS

We can make the coding easier by adding sentinels.



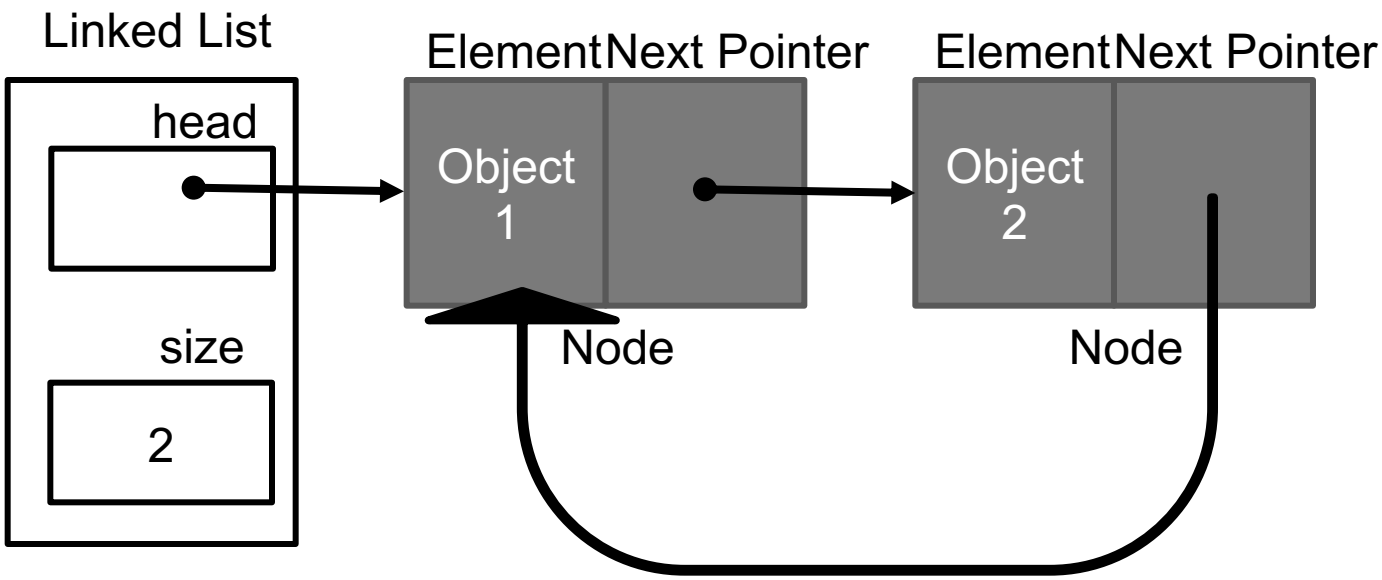
Doubly Linked Lists – adding a Node

Suppose we already have a pointer to the node after the one we want to add – then we can make adding (and deleting) faster!



CIRCULARLY LINKED LISTS

What if we want to support game play or round-robin scheduling? Make the linked list circular!



Want more info? See book section 3.3.

FEB 18 OUTLINE

- Introduce Lab 3
- Recap Singly Linked Lists
- Motivation for Doubly Linked Lists (sorting)
- Doubly Linked Lists
- **Equality and extras**

TESTING EQUALITY

.equals needs to be used to test String equality:

```
String str1 = new String("hello");
String str2 = new String("hello");
System.out.println(str1 == str2);       // false
System.out.println(str1.equals(str2)); // true
```

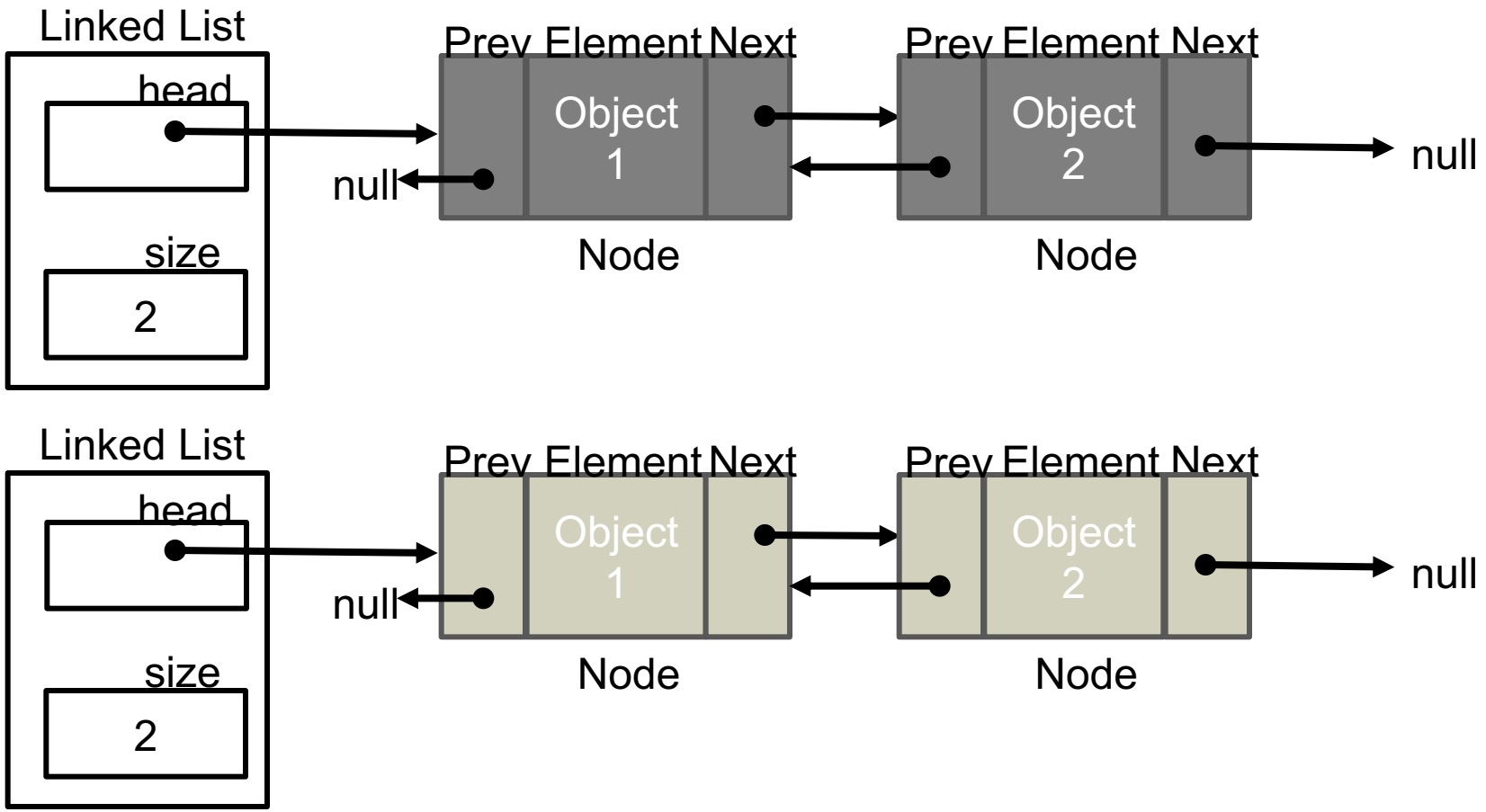
This is the case for all other Objects as well: .equals needs to be implemented in order to compare the *contents* and not the reference address.

For primitive types (int, bool, double) == is okay

Want more info? See book section 3.5.

TESTING EQUALITY: LINKEDLISTS

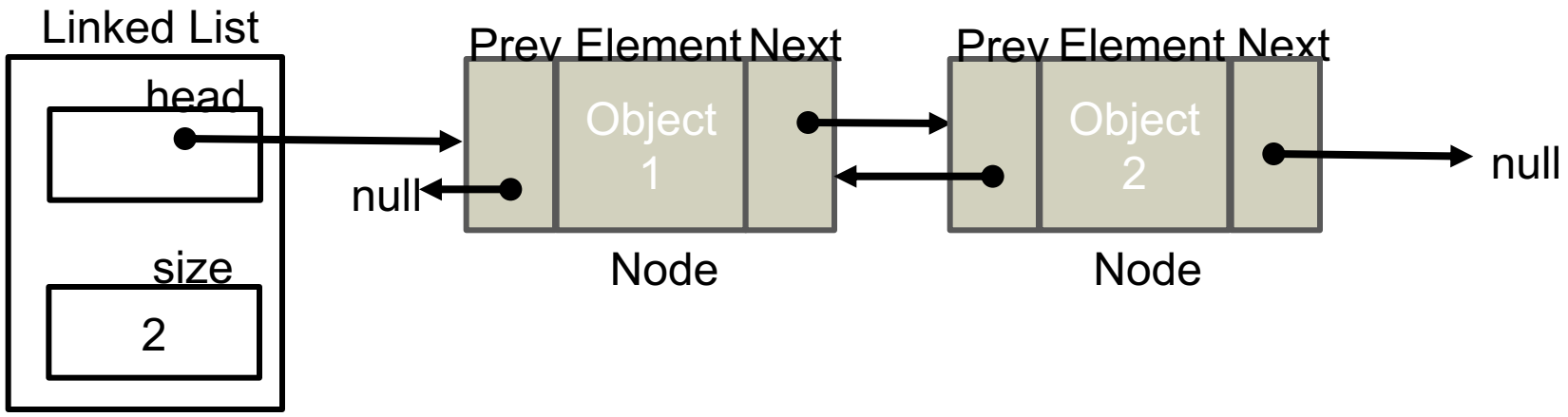
How should equality be tested between LinkedLists?



TESTING EQUALITY: LINKEDLISTS

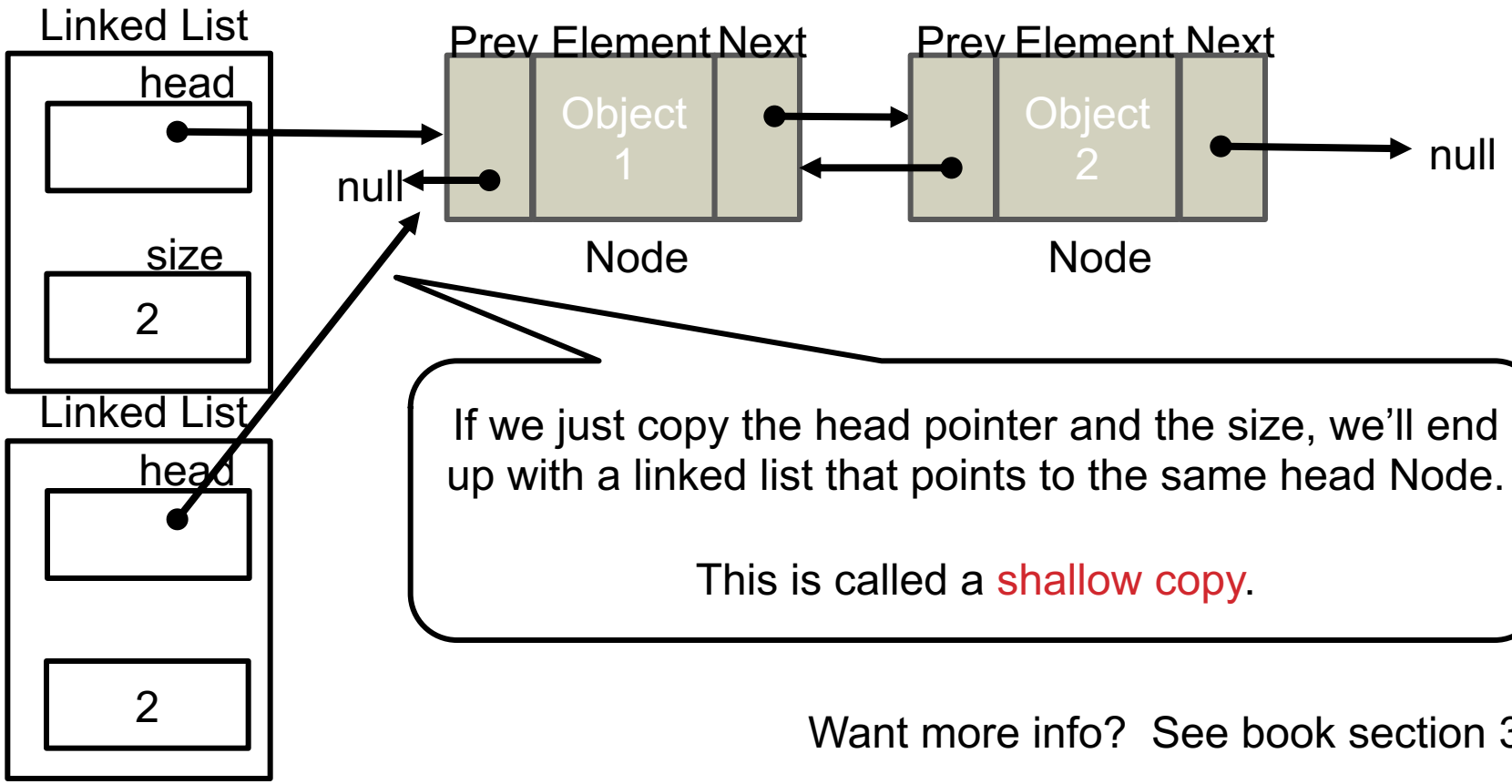
How should equality be tested between LinkedLists?

Implement .equals by traversing both lists and comparing the pairs of Node elements using .equals.



COPYING DATA STRUCTURES

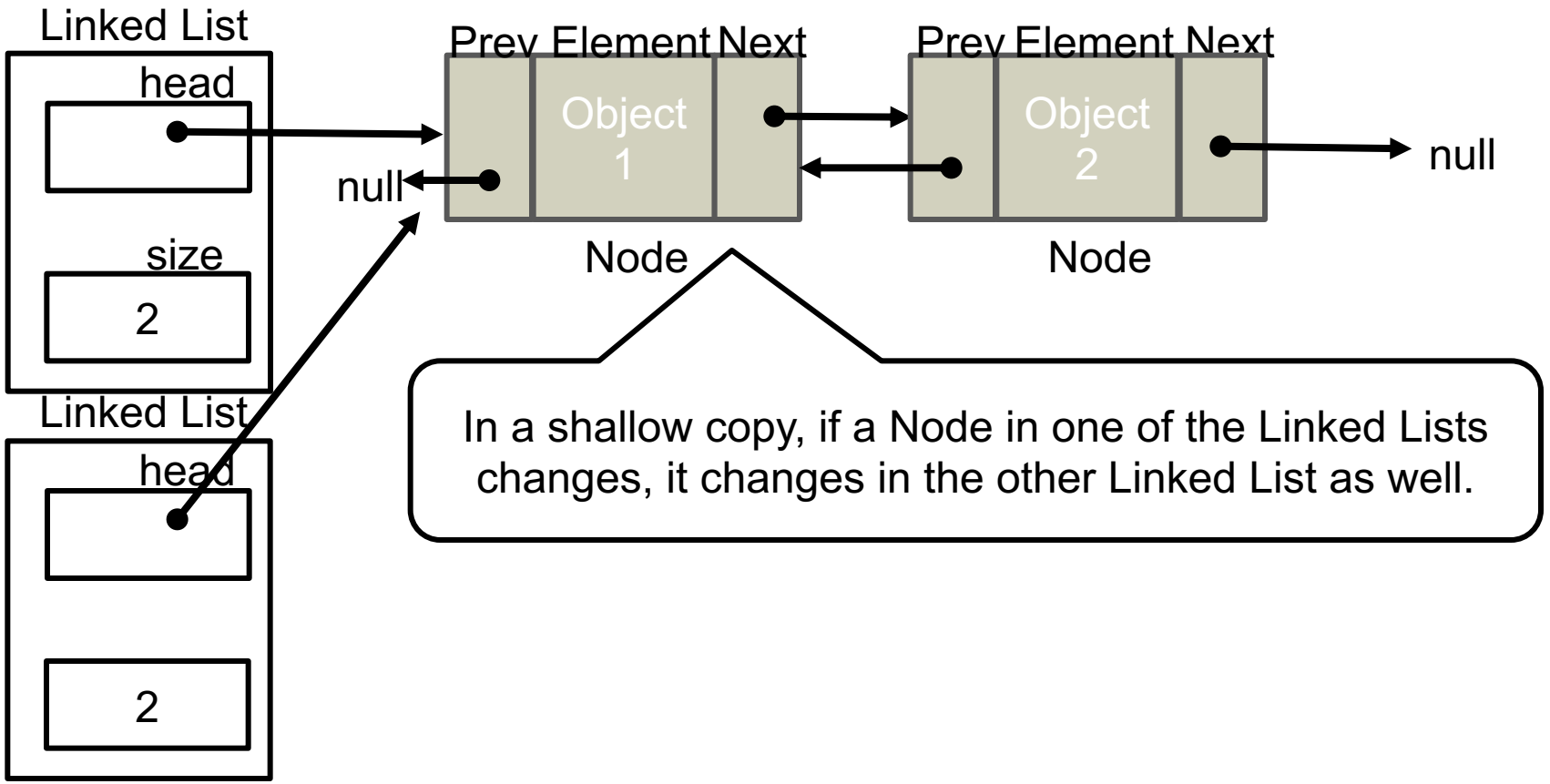
Suppose we copy a linked list by copying the head and size:



Want more info? See book section 3.6.

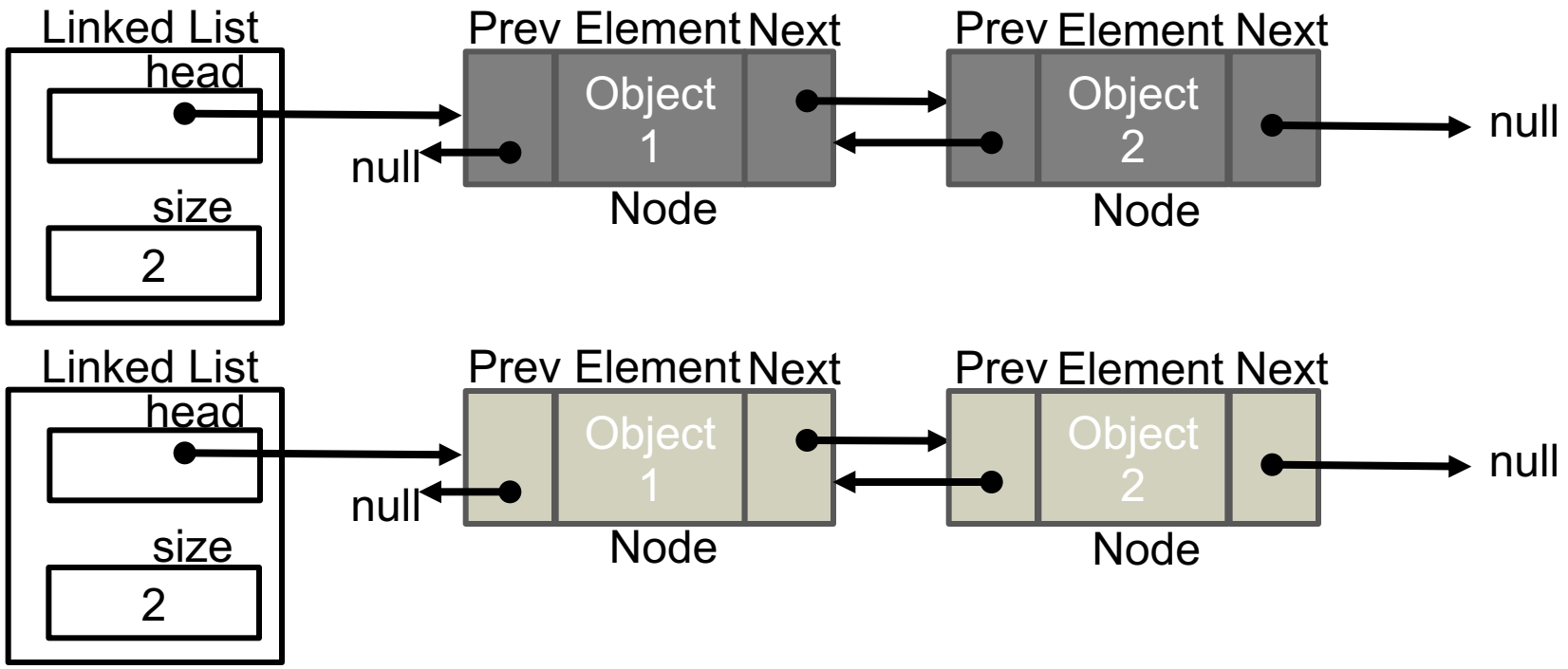
COPYING DATA STRUCTURES

Suppose we copy a linked list by copying the head and size:



COPYING DATA STRUCTURES

What if we want to make sure that we can change one of the Linked Lists without changing the other?
We need to copy each Node's element. This is a deep copy



HANDOUT 9 EXAMPLE