CS 367 - Introduction to Data Structures
Tuesday, February 9, 2016

Homework 3 due 10 pm Friday, February 12th
Program 1 due 10 pm Sunday, February 14th

Assignment questions? Post on Piazza or consult with a TA during scheduled hours.

Report exam conflicts or McBurney exam accommodations by this Friday, 2/12

Last Time

Exceptions Review
- throws and checked vs. unchecked
- defining

Java Primitives vs. References Review
Chains of Linked Nodes
- Listnode class
- practice with chains of nodes

Today

Chains of Linked Nodes
- more practice with chains of nodes

Java Visibility Modifiers
LinkedList Class

Next Time

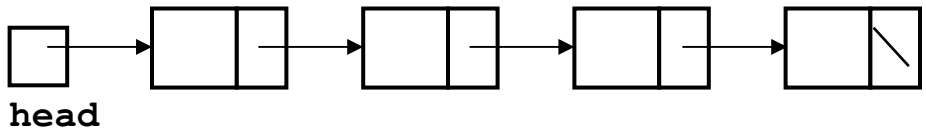
Read: continue *Linked Lists*
LinkedListIterator Class
Linked List Variations
- tail reference
- header node
- double linking
- circular linking

Recall Chain of Linked Nodes Data Structure

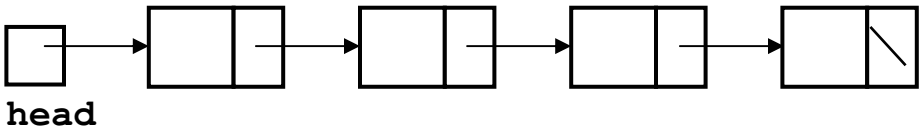
`Listnode` class

```
class Listnode<E> {

   private E data;
   private Listnode<E> next;

   public Listnode(E d)                 { . . . }
   public Listnode(E d, Listnode<E> n){ . . . }
   public E getData()                   { return data; }
   public Listnode<E> getNext()         { return next; }
   public void setData(E d)             { data = d; }
   public void setNext(Listnode<E> n) { next = n; }
}
```

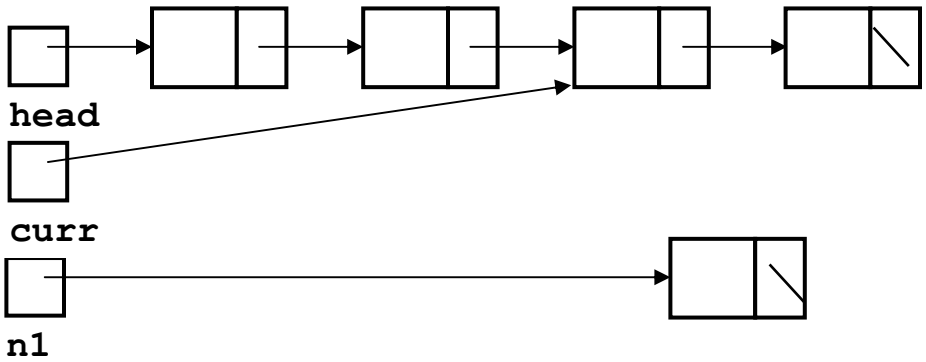
→ **Show how the memory diagrams change** as a result of executing the code beneath each:




```
head.setNext(head.getNext().getNext().getNext().getNext());
```



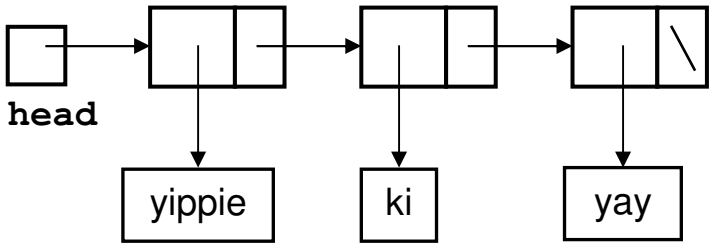

```
head.getNext().getNext().setNext(head);
```




```
n1.setNext(curr.getNext());
curr.setNext(n1);
```

Practice: Making a Chain of Nodes

→ **Create a chain of `Listnode`s** containing the **`String`**s
"yippie", "ki", and "yay" (as shown below) in as few statements as you can.



Practice: Traversing a Chain of Nodes

Assume **`head`** points to the first node in a chain of **`Listnode`**s containing **`String`**s.

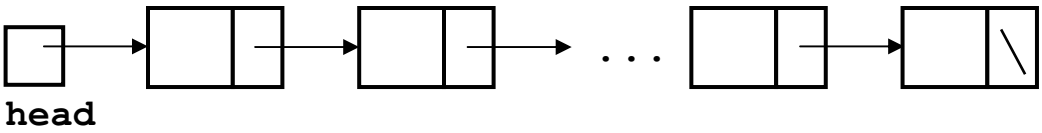
→ **Write a code fragment** that counts the number of strings in the chain of nodes.

```
int count = 0;
```

Practice: Adding a Node at the Chain's End

Assume **`head`** points to the first node in a chain of **`Listnode`**s containing **`String`**s.

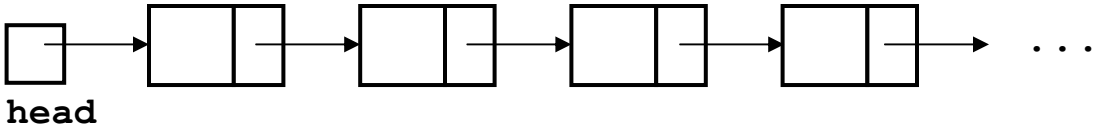
→ **Write a code fragment** that adds a node containing "rear" to the end of the chain of nodes. You may assume the chain has at least one item.



Practice: Removing a Node from a Chain

Assume **`head`** points to the first node in a chain of **`Listnode`**s containing **`String`**s.

→ **Write a code fragment** that removes the third item from the chain of nodes.
You may assume the chain has at least three items.



→ **How would you generalize your code** so it removes the Nth item from the chain of nodes?

Practice: Challenge Question

Assume **`head`** points to the first node in a chain of **`Listnode`**s containing **`String`**s.

→ **Write a code fragment** that reverses the order of the nodes in the chain.

Java Visibility Modifiers

public `public class ArrayList<E>`

private `private Object[] items`

protected `protected String name`

package `class Listnode<E>`

`LinkedList` - Implementing `ListADT` using a Chain of Nodes

```
public class LinkedList<E> implements ListADT<E> {

   private Listnode<E> head;
   private int numItems;



   public LinkedList() {




   }

   public void add(E item) {
```