**ListNodes and Linked Lists: First Lecture**

Here is a college lecture on linked lists. <https://archive.org/details/ucberkeley_webcast_htzJdKoEmO0> The list node part starts at minute 8:00.

# Arrays vs. Linked Lists

A data structure that you have used before is an *array*. Each value is stored in a memory location and is accessed through its index. This data structure makes it easy to jump to any value but more difficult to insert or remove values, because the values may need to be shifted.

Another data structure is the *linked list,* which is built using one or more *list* *nodes*. This data structure makes it easy to insert or remove a value (especially at the front and rear), but impossible to jump directly to a value. Here is a picture of a singly linked list:

3

7

9

head

3

**"Throw", "the", "ball"**

The building block of a linked list is the *list node*. Each list node has a field for the value and field for a reference (or pointer), which points to the next node. Your job now is to gain some experience in rearranging these pointers.

For simplicity, today we will use this non-standard Node class:

public class Node{

|  |  |
| --- | --- |
|  |  |

public Object data;

public Node next;

public Node()

{

data = null;

next = null;

}

public Node(Object v, Node n)

{

this.data = v;

this.next = n;

}

}

We will use this Node class three different times, each time demonstrating how to build the same list "Throw", "the", "ball"

Try #1 (a novice’s attempt):

Node head, node0, node1, node2;

node0 = new Node();

node1 = new Node();

node2 = new Node();

head = node0;

node0.data = "Throw";

node0.next = node1;

node1.data = "the";

node1.next = node2;

node2.data = "ball";

node2.next = null;

Try #2 (an apprentice’s attempt):

Node head, node0, node1, node2;

node2 = new Node("ball", null);

node1 = new Node("the", node2);

node0 = new Node("Throw", node1);

head = node0;

Try #3 (a Jedi master’s attempt):

Node head;

head = new Node ("ball", null);

head = new Node ("the", head);

head = new Node ("Throw", head);

Traverse a linked list:

Node pointer;

pointer = head;

pointer = pointer.next; //dereference the pointer

pointer = pointer.next;

System.out.println(pointer.data);

Traverse a linked list using a **for-loop** to look for “the”:

Node pointer;

for(pointer = head; pointer != null; pointer = pointer.next)

if (pointer.data(equals("the")))

return true;

return false;

//precondition for all 3 methods: head is not null, index is a valid index

public Object get(int index) **//get(1) will return "the"**

Node pointer;

for(pointer = head; pointer <= index; pointer = pointer.next)

continue;

return pointer.data;

public void delete(int index) **//delete(1) will delete "the"**

Node pointer;

for(pointer = head; pointer < index; pointer = pointer.next)

continue;

pointer.data = pointer.next.next.data;

public void insert(int index, Object obj) **//insert(2,"big") will insert “big”**

**//before “ball”**