**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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| theArray |  | **3** | **3** | **7** | **9** |
|  |  | [0] | [1] | [2] | [3] |

Data structure: *array*.

Data is accessed through an *index*.

Arrays are fixed-length.

Quick to retrieve any value, by jumping to its index.

More difficult to insert or remove values, because the values may need to be shifted.

Data structure: *linked list,* built using one or more *list* *nodes*.

Data accessed by traversing each node from the *head*.

Linked lists can grow or shrink as needed.

3

7

9

head

3

Impossible to jump directly to a value.

Quick to insert or remove a value, especially   
 at the front or the rear.

**"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"**

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

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

**//before “ball”**