Linked Lists

ArrayLists are slow to add/remove from the front.

```
public void add(int idx, E data) {
    for (int i=numElems; i>idx; i--) {
        arr[i] = arr[i-1];
    }
    arr[idx] = data;
}
```

A linked structure is faster.

ListNode

```
public class ListNode {
    public int data;
    public ListNode next;
}
```

Note: If you want a type that simply groups data, it is common to have no methods and the fields `public`.

What does the following code produce?

```
ListNode a = new ListNode();
a.next = new ListNode();
a.next.next = new ListNode();
```

Making lists

We can make arbitrary lists. For [1,2,3,4]:

```
ListNode front = new ListNode();
front.data = 1;
front.next = new ListNode();
front.next.data = 2;
front.next.next = new ListNode();
front.next.next.data = 3;
front.next.next.next = new ListNode();
front.next.next.next.data = 4;
```

Changes at the front

Adding at the front is more efficient than ArrayList

```
ListNode newNode = new ListNode();
newNode.next = front;
front = newNode;
```

Just a few instructions instead of a loop.

Encapsulation

Tedious and error prone to work like this.

Let's make a class that implements a list this way.

Hides the implementation details from the user.

Provides a simple abstraction for keeping lists this way.

This is encapsulation, data hiding, abstraction.

Steps to writing a class

1) Decide you need a class. (We just did that.)

2) Choose the methods to support.

3) Choose data representation.

4) Implement, test, document.

Methods

Let's implement the basic list methods from the book.

```
add(value)
add(idx, value)
clear()
get(idx)
remove(idx)
set(idx, value)
size()
```

Data representation

What does a list need to remember between method calls?

The list of data, obviously.

What does that mean for a linked list? We can't remember everything in the list without an array.

All we really need to know is where the front of list is.

```
private ListNode front;
```

Implement and test

To keep simple, we'll only have each node hold an int.

Strategy:
- Write constructor and toString first, test them
- Repeat: Write a method and test it

Book has lots of examples of methods, so none are in these slides.

Starting point

```
public class ListNode {
    public int data;
    public ListNode next;
}

public class LinkedIntList {
    private ListNode front;
    ... methods go here ...
}

public class Tester {
    public static void main(String[] args) {
        LinkedIntList list = new LinkedIntList();
        ... test code goes here ...
    }
}
```