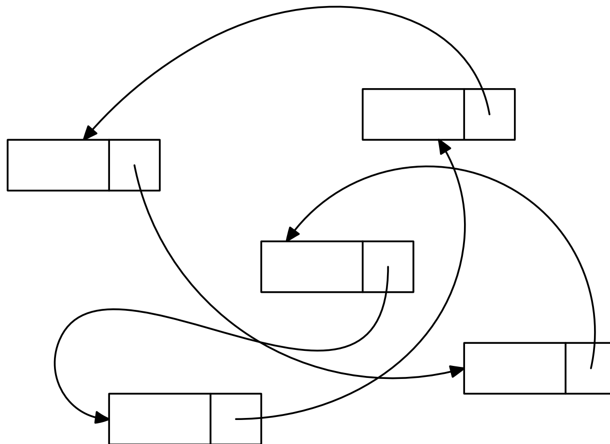
Topic 1:

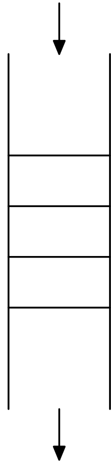
Review Material

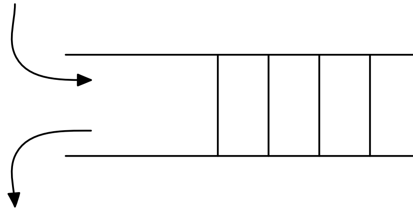
(or: Let's Spackle Some Knowledge Holes!)

What is This, and What is Missing?



What are these Data Structures?





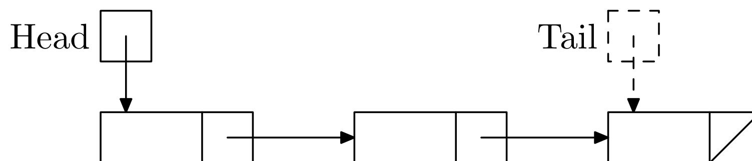
Singly–Linked List (SLL) Review (1 / 2)

Singly–Linked List (SLL) Review (2 / 2)

What are some common SLL operations?

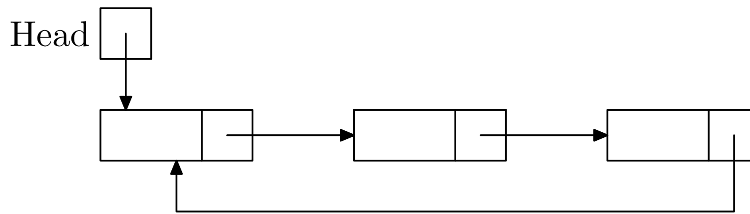
SLLs: Tail Reference or no Tail Reference?

When would a Tail reference be helpful? Or annoying?



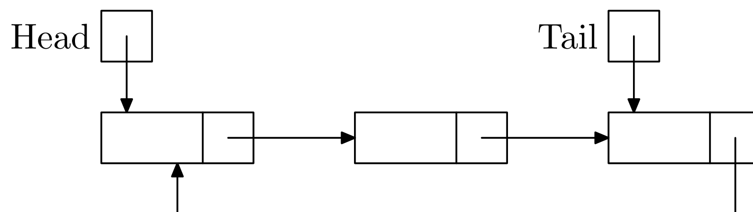
Circularly–Linked Lists (CLLs)

Idea: Allow easy travel from last node to first.

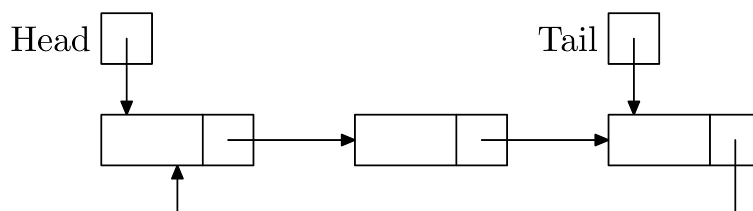


CLLs: Are Tail References Helpful?

Append

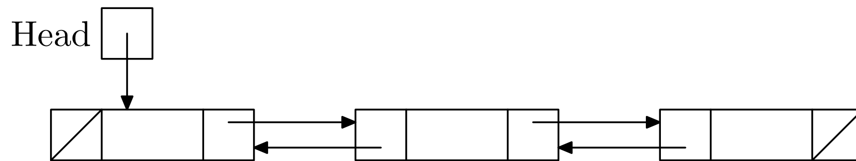


Prepend



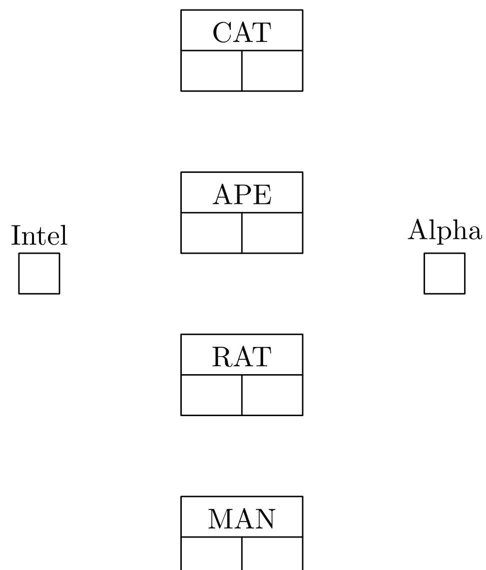
Doubly–Linked Lists (DLLs)

An extra reference per node allows two–way travel.

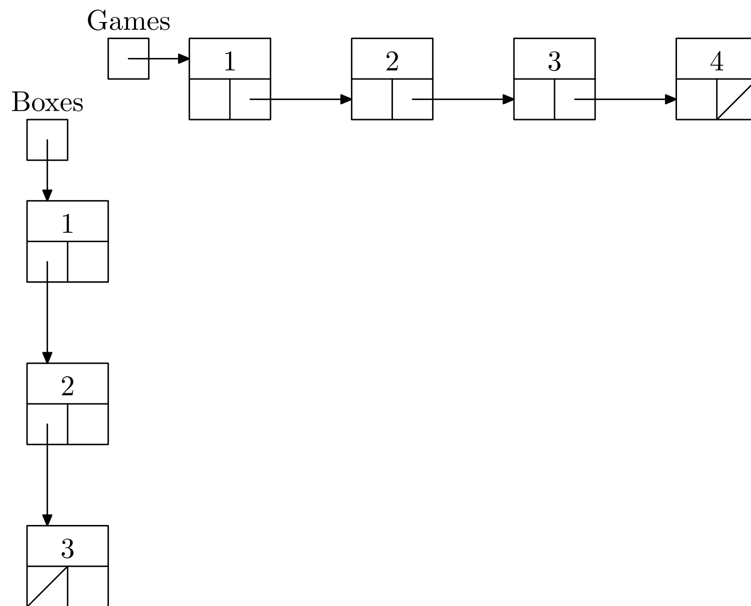


MultiLists

Here, each node is part of two distinct lists of the same data.



Orthogonal Lists



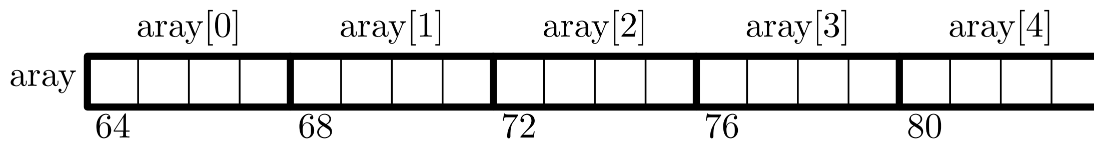
1D Array Storage

Array elements are stored *contiguously* in memory.

Example(s):

Where in Memory is `aray[index]`? (1 / 3)

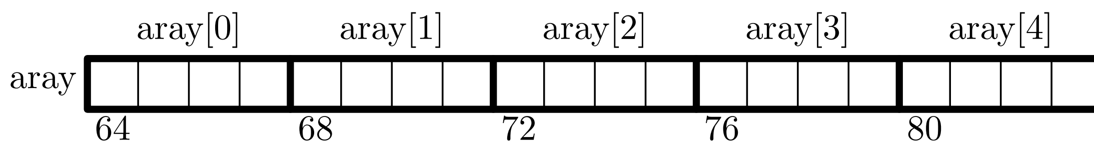
We need to know a few things first:



-
-
-

Where in Memory is `aray[index]`? (2 / 3)

To reach index i's element, we have to 'jump' over i elements.



Recall: esize = 4, base = 64, index = 3

Where in Memory is `aray[index]`? (3 / 3)

OK, great,

```
address = base + index * esize
```

but . . . so what?

2D Array Allocation in Java

There are more options than you might guess. Two of them:

2D Array Storage (1 / 5)

How can we store a 2D structure in 1D memory? Two options:

	0	1	2	3
0	6	0	8	2
1	1	5	3	6
2	9	4	7	1

2D Array Storage (2 / 5)

Consider locating 4 in our 2D array:

	0	1	2	3
0				
1				
2		4		

RMO:

									4		

CMO:

					4						

Our 2D array address calculation is in two parts:

2D Array Storage (3 / 5)

For our 1D calculation, we needed:

- base address
- esize
- index

What information do we need for the 2D calculation?

2D Array Storage (4 / 5)

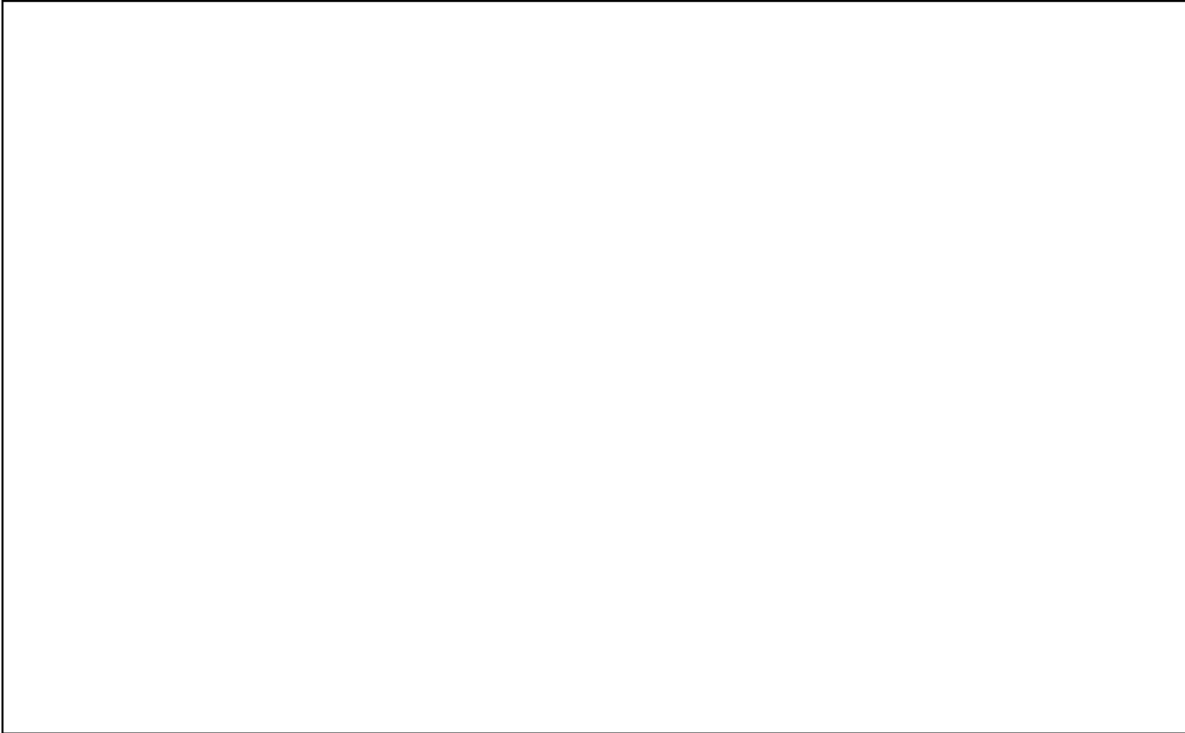
Equation time! (I'll do RMO; you can do CMO on your own.)

twod

									4		

2D Array Storage (5 / 5)

Example(s):



What About 3D Arrays?

The same basic idea, just one more dimension!

Think about slicing pre–croutons from a loaf of bread. $\Longrightarrow$

nD Arrays in OO Languages (1 / 3)

In object–oriented (OO) languages, arrays are objects.

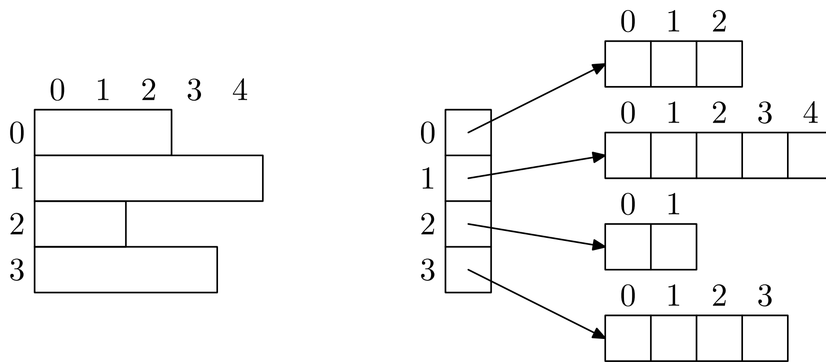
A 1D array object is a contiguous collection of references to data objects.

nD Arrays in OO Languages (2 / 3)

Because each row is a distinct object, they can . . .

nD Arrays in OO Languages (3 / 3)

How do we declare & allocate such arrays in Java?



Recursion Review

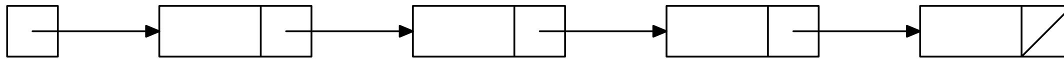
Want to solve a problem recursively? Try to answer both of these questions:

1.

2.

Simple Recursion Example #1 (1 / 2)

Task: Print the content of an SLL front to rear.



Q1: What's somewhat simpler than printing an SLL of n items front to rear?

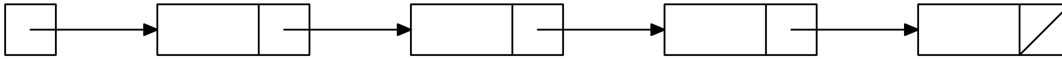
Q2: How does that help print the list of n items front to rear?

Simple Recursion Example #1 (2 / 2)

Let's turn our answers into a pseudocode algorithm:

Simple Recursion Example #2 (1 / 2)

Task: Print the content of an SLL rear to front.



Q1: What's somewhat simpler than printing an SLL of n items rear to front?

Q2: How does that help print the list of n items rear to front?

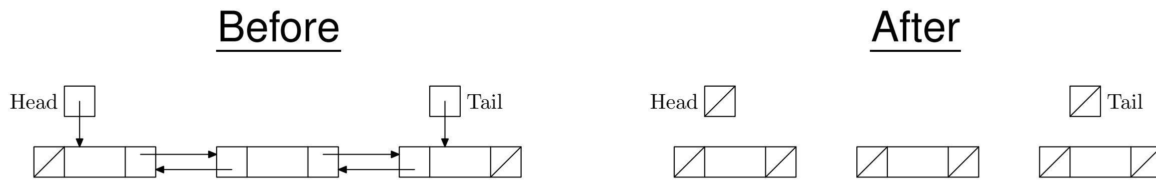
Simple Recursion Example #2 (2 / 2)

There's just one change from the 'front to rear' version:

```
1  subprogram printLL (given: head) returns nothing
2      if head is null:
3          return
4      else:
5          call printLL with head's successor
6          print head's data
7      end if
8  end subprogram
```

Simple Recursion Example #3 (1 / 3)

Task: Totally unlink a DLL.



Q1: What's somewhat simpler than unlinking a DLL of n items?

Q2: How does that help unlink the DLL of n items?

Simple Recursion Example #3 (2 / 3)

In pseudocode:

Simple Recursion Example #3 (3 / 3)

We need to set Head and Tail to null . . . but when and where?

Pseudocode: