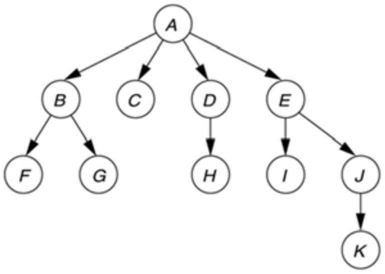


CSSE 230 Day 6

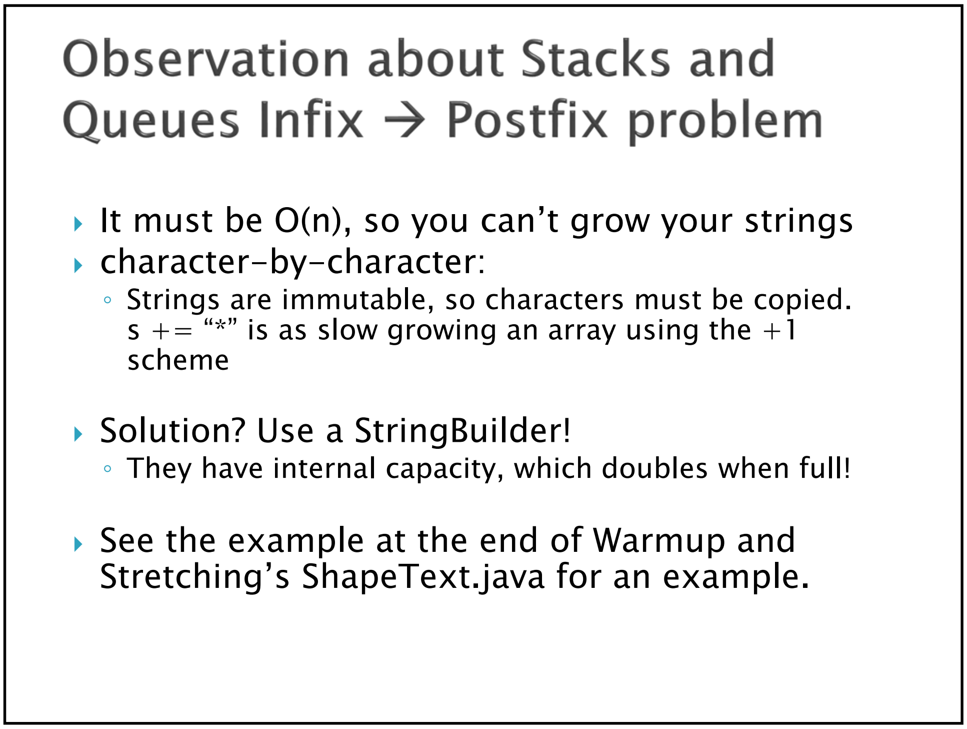
Intro to Trees

After today, you should be able to…
…use tree terminology
…write recursive tree functions

Checkout **BinarySearchTree** from SVN

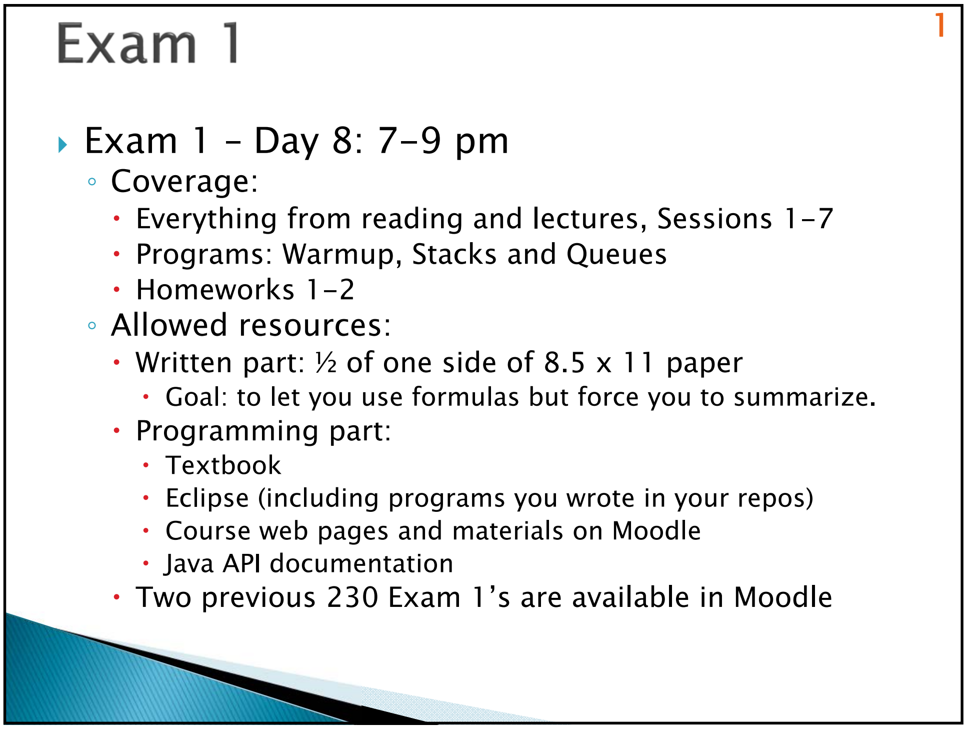
Announcements

- Review Day 5's quizzes on Java Collections and Data Structures
- Preview of HW3: includes an essay



Observation about Stacks and Queues Infix → Postfix problem

- It must be O(n), so you can't grow your strings
- character-by-character:
  - Strings are immutable, so characters must be copied. s += "*" is as slow growing an array using the +1 scheme
- Solution? Use a StringBuilder!
  - They have internal capacity, which doubles when full!
- See the example at the end of Warmup and Stretching's ShapeText.java for an example.



Exam 1

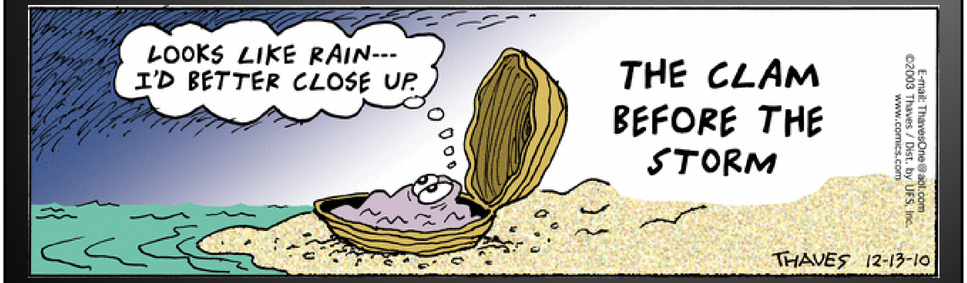
1

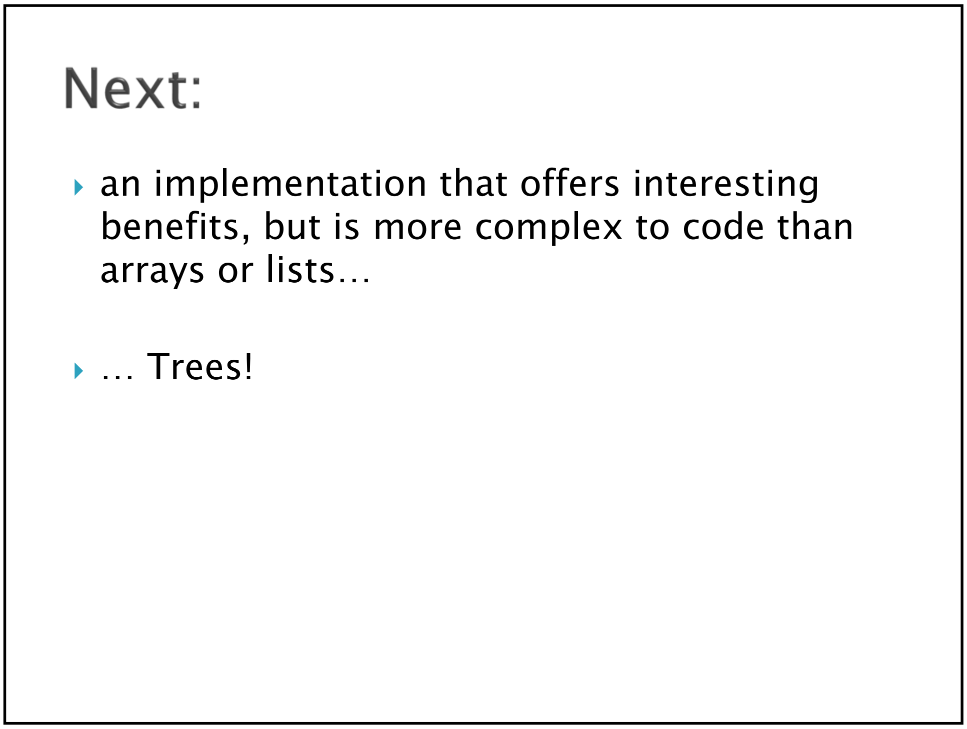
- Exam 1 – Day 8: 7-9 pm
  - Coverage:
    - Everything from reading and lectures, Sessions 1-7
    - Programs: Warmup, Stacks and Queues
    - Homeworks 1-2
  - Allowed resources:
    - Written part: ½ of one side of 8.5 x 11 paper
      - Goal: to let you use formulas but force you to summarize.
    - Programming part:
      - Textbook
      - Eclipse (including programs you wrote in your repos)
      - Course web pages and materials on Moodle
      - Java API documentation
    - Two previous 230 Exam 1's are available in Moodle

Exam 1 Possible Topics

- Written (50-70%):
  - Growable Arrays
  - MCSS
  - big O/θ/Ω: true/false, using definitions, limits, code analysis
  - Binary search
  - ADT/Collections
  - Choosing an ADT to solve a given problem
- Programming (30-50%):
  - Implementing an ADT using an array, nodes, or another ADT
  - Writing an efficient algorithm to solve a simple array-based problem

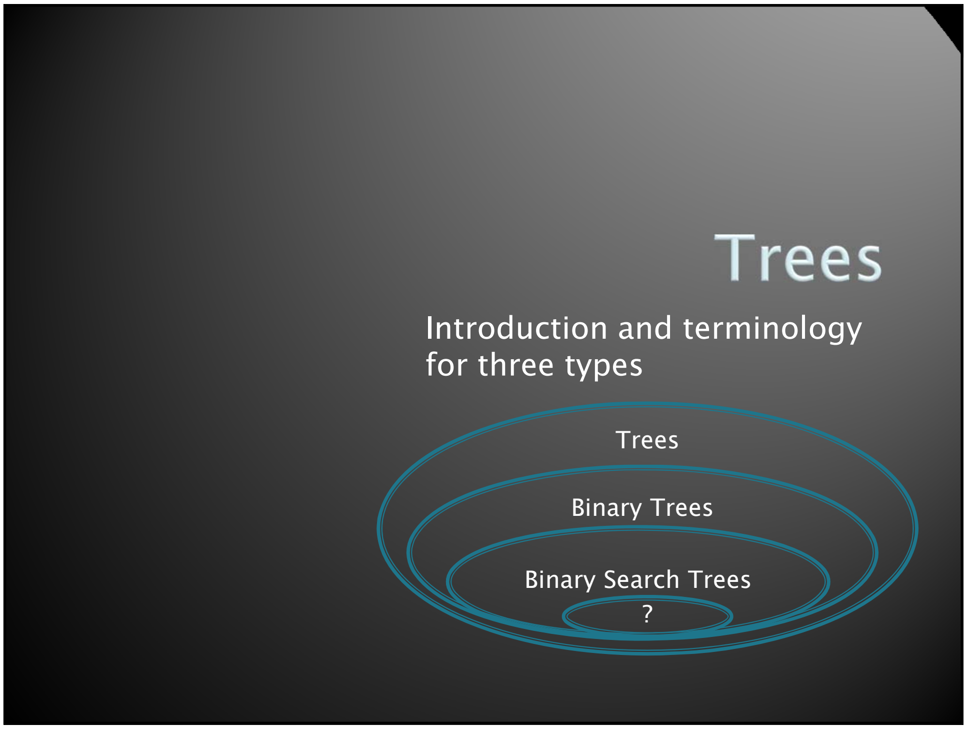
Questions?


LOOKS LIKE RAIN---
I'D BETTER CLOSE UP.
THE CLAM BEFORE THE STORM
THAVES 12-13-10
E-mail: ThavesOne@aol.com
©2003 Thaves / Dist. by UFS, Inc.
www.comics.com



Next:

- an implementation that offers interesting benefits, but is more complex to code than arrays or lists…
- … Trees!



Trees

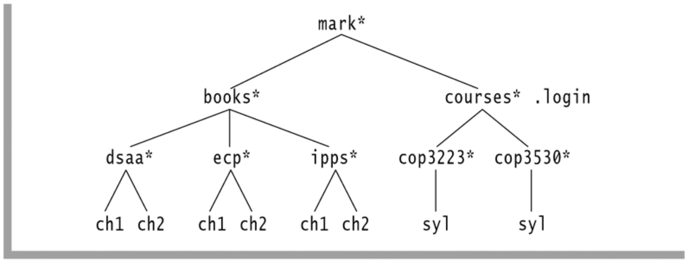
Introduction and terminology for three types

Trees in everyday life

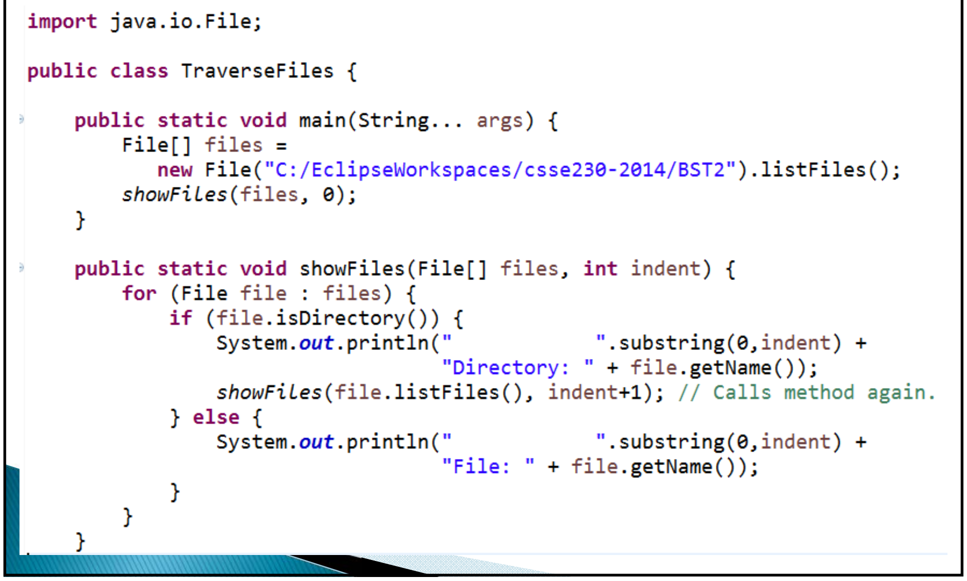
- Class hierarchy tree (single inheritance only)
- Directory tree in a file system

figure 18.4

A Unix directory



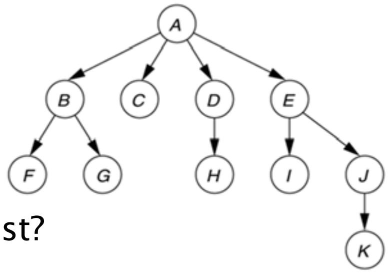
Traverse a Directory Tree



```java
import java.io.File;

public class TraverseFiles {

    public static void main(String... args) {
        File[] files =
           new File("C:/EclipseWorkspaces/csse230-2014/BST2").listFiles();
        showFiles(files, 0);
    }

    public static void showFiles(File[] files, int indent) {
        for (File file : files) {
            if (file.isDirectory()) {
                System.out.println("            ".substring(0,indent) +
                                   "Directory: " + file.getName());
                showFiles(file.listFiles(), indent+1); // Calls method again.
            } else {
                System.out.println("            ".substring(0,indent) +
                                   "File: " + file.getName());
            }
        }
    }
```

A General Tree—Global View

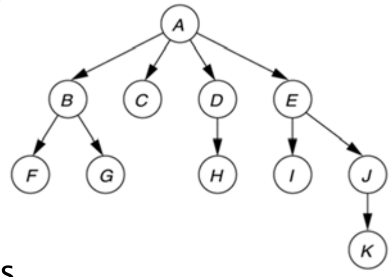
- A collection of nodes
- Nodes are connected by directed edges.
  - One special root node has no incoming edges
  - All other nodes have exactly one incoming edge
- One way that Computer Scientists are odd is that our trees usually have their root at the top!



- How are trees like a linked list?
- How are they different?

Q1, 2

Tree Terminology



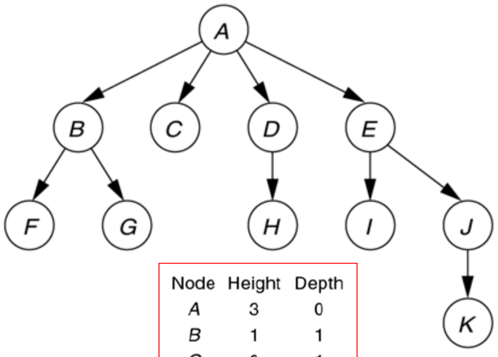
- Parent
- Child
- Grandparent
- Sibling
- Ancestors and descendants
- Proper ancestors, proper descendants
- Subtree
- Leaf, interior node
- Depth and height of a node
- Height of a tree

Q3-6

Node height and depth examples

figure 18.1

A tree, with height and depth information



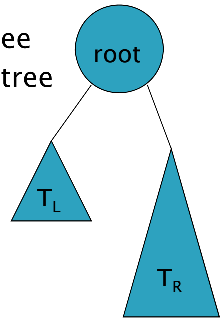
Node	Height	Depth
A	3	0
B	1	1
C	0	1
D	1	1
E	2	1
F	0	2
G	0	2
H	0	2
I	0	2
J	1	2
K	0	3

The height of **a tree** is the height of its root node.

Which is larger, the sum of the heights or the sum of the depths of all nodes in a tree?

Binary Tree: Recursive definition

- A **Binary** Tree is either
  - empty, or
  - consists of:
    - a distinguished node called the root, which contains an element, and
    - A left subtree T_L, which is a binary tree
    - A right subtree T_R, which is a binary tree
- **Binary** trees contain at most 2 children



Q7-9

Binary Search Trees (BST)

- Q: What property enables us to search BSTs efficiently?

- A: Every element in the left subtree is smaller than the root, and every element in the right subtree is larger than the root. And this is true at **every node**, not just the root.

Connections with Linked Lists

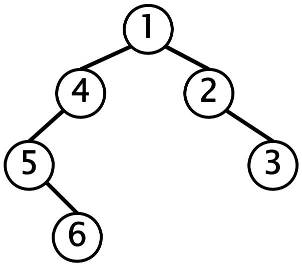
- Write size() for linked list
  - Non-recursively
  - Recursively

- Write size() for a tree
  - Recursively
  - Non-recursively (later)

Growing Trees

- Let’s start the BinarySearchTrees assignment: implement a **`BinaryTree<T>`** class

Test tree:



A single tiny recursive method for size will touch **every node in the tree.** Let’s write, then watch in debugger.