4/7/2008
Let's look at solutions to the interclass problem.
Do you have any questions about the assignment?
Do you have any questions about the reading?
Comments on recursion
  - Do I use recursion in my research? Yes, for trees.
  - How is recursion used in advanced programming?
  - Is recursion used differently in different languages?
  - Tracing the midterm EC.
Goals of this class. Why is it hard? Why do we move so fast?
Let's finish writing that quicksort routine that we started last class.
What is a Tree?

• You are all familiar with what normal trees look like. In CS we use the term somewhat differently, and more formally.

• To describe trees we need some basic terminology
  – Node - an element of a tree. One node is designated as the “root”
  – Edge - a directed connection from one node to another.
Every node, C, has exactly one incoming edge from another node, P. P is said to be the parent of child node C. Root has 0.

There is a unique path from the root to any node. The number of edges on that path is called the path length. It is also called the depth of the node.

A node with no children is called a leaf. The path length from a node to the deepest leaf in the height of that node.
• Following the parent-child analogy, children of the same node are called siblings. We also call any node on a path below a given node a descendant and any above an ancestor.
• You might also hear the size of a node referred to as the number of descendants of a node, including itself.
• We can also define a tree as either empty, or a root with zero or more subtrees where the root connects to the roots of those subtrees.
In a general tree, each node can have zero or more children. That is a lot of flexibility. We want a class to represent nodes. To get this flexibility we can use a linked list. Each node has pointers to a first child and the next sibling.

It might be just as easy to have the child member be an ArrayList that we put Nodes in. File systems are a good example of this.
As with any data structure one of the things you want to be able to do is to traverse through all the elements.

Think for a while about how you would do this? There is even a question about the order you traverse them in. Do you want to process a node before you process its children or after? If before we call it a preorder traversal. If after it is a postorder traversal.
The simplest way to do a traversal is through recursion. If you want to do it with a loop you have to implement a data structure to store some nodes or have the tree specially set up.

The traverse function takes a node and calls itself once with each child node. It also does whatever the visit operation is.

Preorder does a visit before going to children and postorder visits after going to children.

Breadth first uses a queue, not recursion.
For our first example of a tree, I want to make some changes to our formula parser.

If we introduce variables we might evaluate the same formula many times with different values. It is inefficient to do the same string processing over and over.

It is more efficient to parse the string once and build a tree that represents the formula then do the evaluation on that tree.

Let's code this.
Do you have any questions about the things that we did today?
Assignment #5 is due today.
Thursday is your one “free” day.
The mock programming competition in Wednesday 3-6pm in this room.
Interclass Problem – Take the formula parser with variable support and edit a drawable so that it can use formulas