**Trees**

A tree us an abstract model of a hierarchical structure.

Consists of nodes with a parent-child relationship.

Applications:  
File systems  
Programming environments  
Organisation charts

1st node: root (node without a parent)  
other terminology in slides.  
  
Depth of a specific node  
-you can recursively call 1+depth(parent) to get depth of a node.  
Height is whole tree (can do sub tree as well though)  
-we look at max height, as you need to look at the node which is the furthest away from the root. You take the max of all the children to determine this.

**Tree ADT**

Generic Methods

Int size()  
Boolean isEmpty()  
Iterator elements()  
Iterator positions()  
  
Accessor methods  
position root()  
position parent(p)  
positionIterator children(p)

Query methods

Boolean isInternal(p)  
bool isExternal(p)  
Boolean isRoot(p)

Update Methods   
… missed this one

You can make all external nodes all null, and the root null.

Traversal: Moving around/accessing nodes in a tree.

Preorder traversal: a node is visited before its descendants

Postorder traversal: a node is visited after its descendants (visit comes last)

**Binary Trees**

Only have at most 2 children per node(left and right child)  
-if has exactly 2 or none for every node, called **proper binary tree**

Application  
-arithmetic expressions  
-decision processes  
-searching

Arithetic Expression Tree (must be a proper binary tree)  
-make the internal nodes \*+-/ and the external nodes the 3,5,1,2 ect, and a inorder traversal to resolve them.

Inorder: you resolve by visiting in between nodes.

Methods:

Position left(p)  
position right(p)  
hasLeft  
hasRight

Bianry Tree Properties, **very NB – these make these attractive to use.**

External nodes = e  
internal nodes = i  
number of nodes = n  
height = h  
  
e = i+1  
n = 2e-1  
h <= i  
h <= n-1  
e <= 2^h  
h >= log2(e)  
h >= log2(n+1) – 1

A binary tree has linear worst time complexity, as if it unbalanced it becomes a linear data structure.