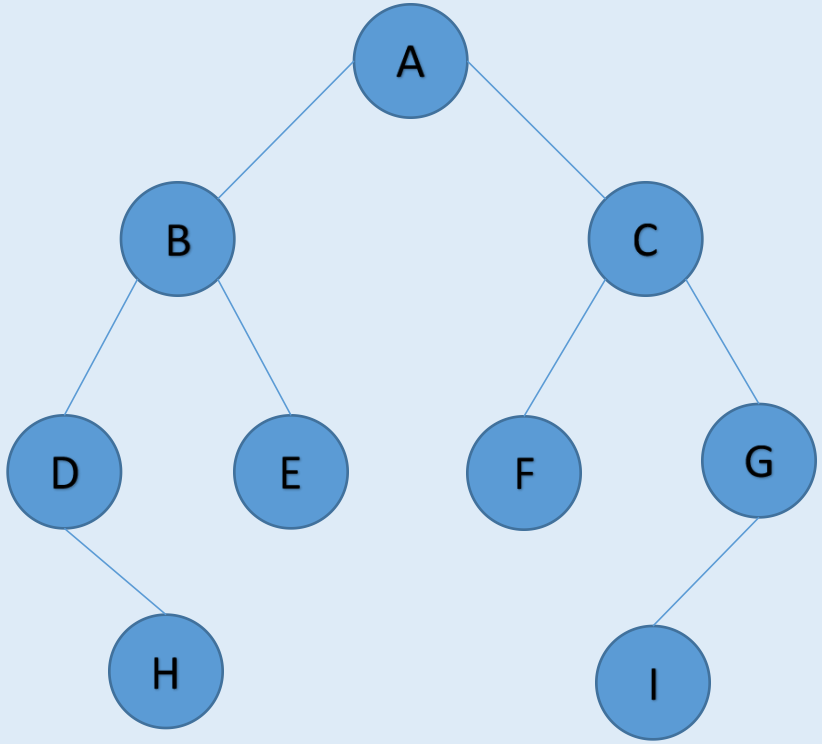
Binary Trees

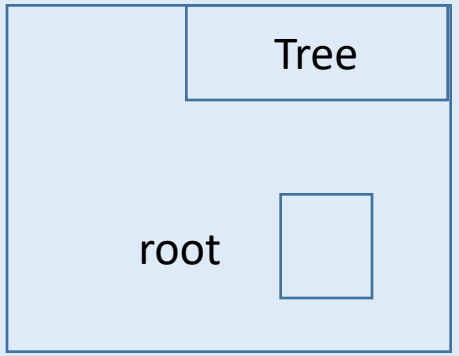
Example 1:



- Nodes F and G are the children of C where
  - F is the left child
  - G is the right child
  - F and G are siblings
- Node A is the root
- Nodes H, E, F, I are the leaves
- Nodes B, D, E, H are the left subtree
- Nodes C, F, G, I are the right subtree

Building a binary tree using a linked list

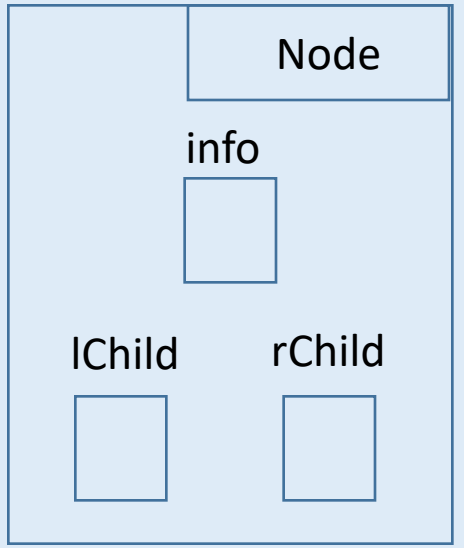
```
public class Tree {
    private Node root;
}
```



The Node class

```
public class Node {
    int info;  // can be a link to an object
    Node lChild;  // refers to Null if empty
    Node rChild;

    Node (int i, Node l, Node r){
    // constructor for node
        info = i;
        lChild = l;
        rChild = r;
    }
}
```

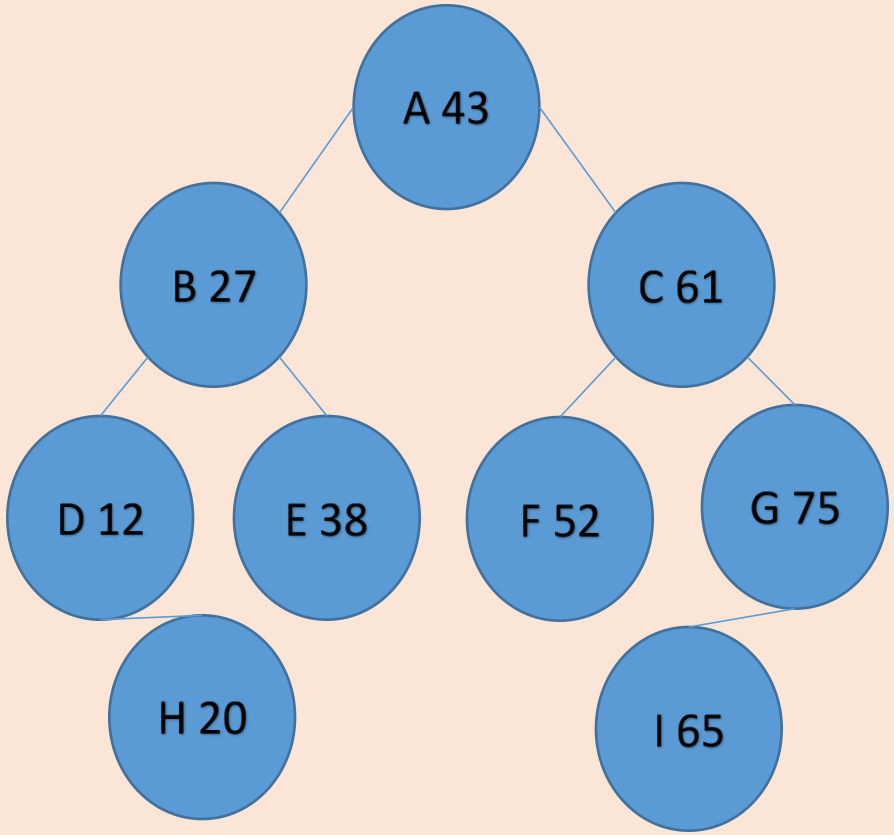


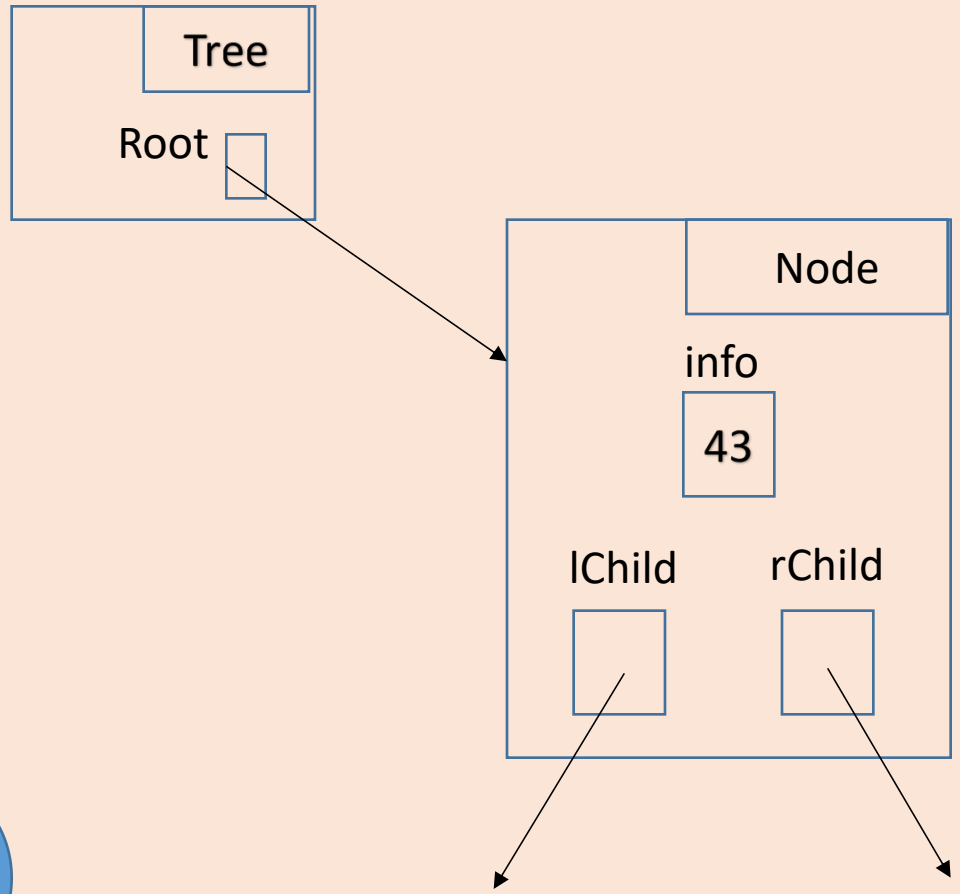
```
public class Tree {
        private Node root;

        public class Node {
                int info;  // can be a link to an object
                Node lChild;  // refers to another Node or Null if empty
                Node rChild;

                Node (int i, Node l, Node r){
                // constructor for node
                        info = i;
                        lChild = l;
                        rChild = r;
                }
        }
}
```

Example 2:


A 43
B 27
C 61
D 12
E 38
F 52
G 75
H 20
I 65


Tree
Root
Node
info
43
lChild
rChild

Recursion

- Process root directly and process left and right subtrees recursively
  - An empty tree acts as a stopping condition

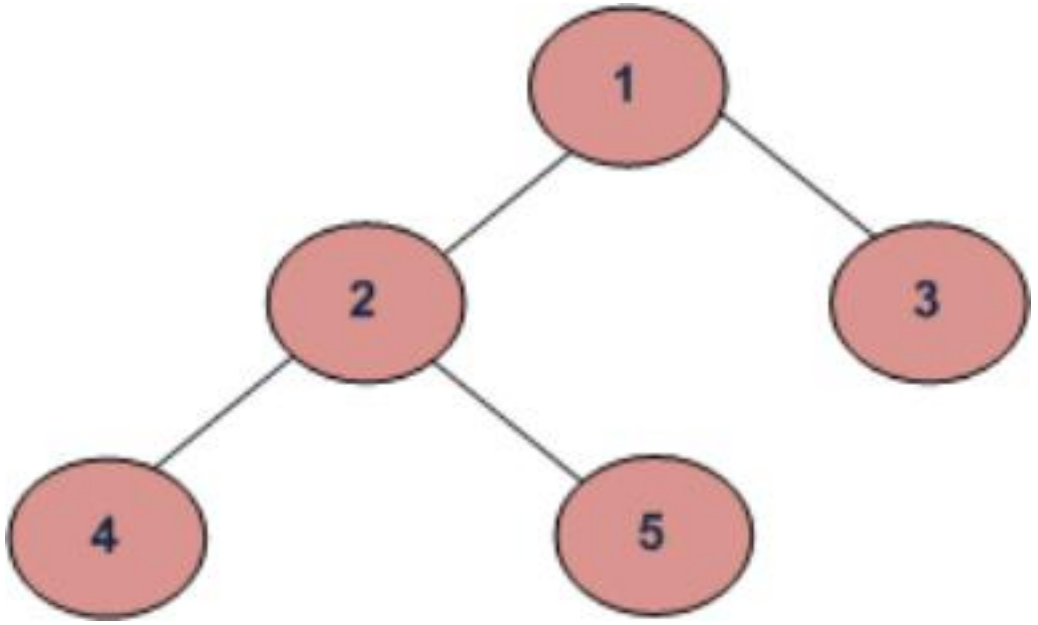
Common Binary Tree Traversals

- To traverse a tree, we want to visit each node, performing some task (such as printing the data) as we carry out the visitation

Preorder
Root
Left Subtree
Right Subtree

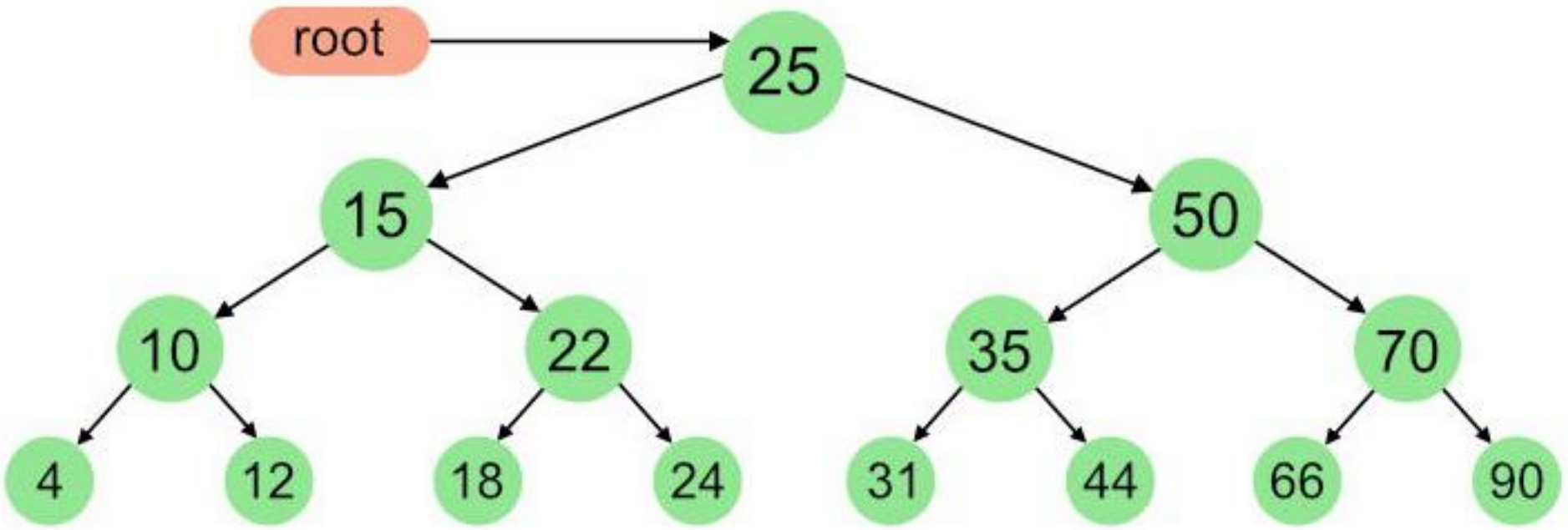
Inorder
Left Subtree
Root
Right Subtree

Postorder
Left Subtree
Right Subtree
Root



Traversals	
Inorder (Left, Root, Right):	4 2 5 1 3
Preorder(Root,Left,Right):	1 2 4 5 3
Postorder(Left, right, Root):	4 5 2 3 1

- In what order would the nodes of the following tree be visited using
  - an inorder traversal?
  - a postorder traversal?
  - a preorder traversal?



InOrder(root) visits nodes in the following order:

4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90

A Pre-order traversal visits nodes in the following order:

25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90

A Post-order traversal visits nodes in the following order:

4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25

Example 3 – Algorithm for Inorder traversal

To TRAVERSE THE TREE

    if the tree is not empty

        TRAVERSE THE LEFT SUBTREE, i.e., call Inorder (left-subtree)

        visit the root

        TRAVERSE THE RIGHT SUBTREE, i.e., call Inorder (right-subtree)

Example 3: code in tree class (empty list)

```
public void inPrint() {
     if (root!=null)
          root.inPrint();
}
```

Example 3 – In the inner Node class

```
void inPrint() {
     if (lChild!=null)
          //prints non-empty left subtree
          lChild.inPrint();
     System.out.println(info);
     if (rChild!=null)
          //print non-empty right subree
          rChild.inPrint();
}
```