Tree and Binary Tree- part III

Tree Travesal:

3 possible traversing approaches: inorder, postorder and preorder.

Inorder Traversal

- To print a Binary Search Tree in sorted order
  Print the left subtree.
  Print the root data.
  Print the right subtree.
- This called an **inorder** traversal.

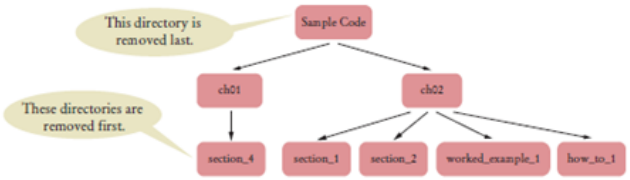
- Recursive helper method for printing the tree.

```
private static void print(Node parent)
{
   if (parent == null) { return; }
   print(parent.left);
   System.out.print(parent.data + " ");
   print(parent.right);
}
```

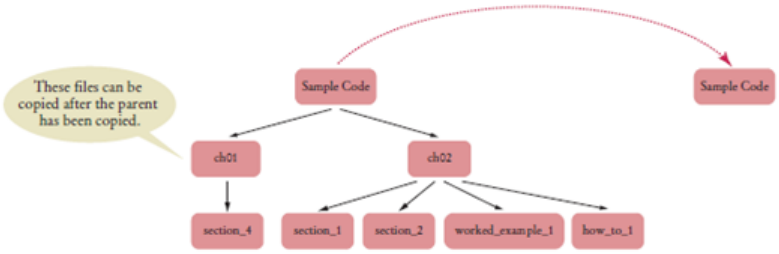
- Public `print` method starts the recursive process at the root:

```
public void print()
{
   print(root);
}
```

- Preorder
  - Visit the root
  - Visit left subtree
  - Visit the right subtree
- Postorder
  - Visit left subtree
  - Visit the right subtree
  - Visit the root
- A postorder traversal of an expression tree results in an expression in reverse Polish notation.
- Use postorder traversal to remove all directories from a directory tree.
  - A directory must be empty before you can remove it



- Use preorder traversal to copy a directory tree.



- Can have pre- and post-order traversal for any tree.
- Only a binary tree has an inorder traversal.

Depth-first search

- Iterative traversal can stop when a goal has been met. **Depth-first** search uses a stack to track the nodes that it still needs to visit.
- Algorithm:

  Push the root node on a stack.
  While the stack is not empty
    Pop the stack; let n be the popped node.
    Process n.
    Push the children of n on the stack, starting with the last one.

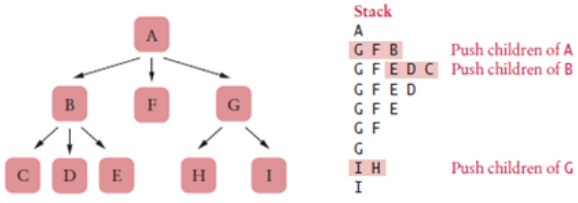


Figure 16
Depth-First Search

Stack	
A	
G F B	Push children of A
G F E D C	Push children of B
G F E D	
G F E	
G F	
G	
I H	Push children of G
I	

Breath-first search

- **Breadth-first** search first visits all nodes on the same level before visiting the children.
- Breath-first search uses a queue.