

# Java

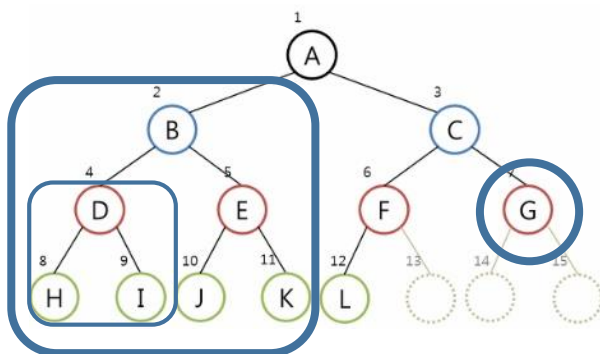
```
public class BST {  
    private Node root=null;  
  
    private class Node{  
        private int key;  
        private Node left_child;  
        private Node rightchild;  
        private Node parent;  
        Node(int key){  
            this.key = key;  
            this.left_child=null;  
            this.right_child=null;  
        }  
    }  
}
```

**C**

```
typedef struct _bTreeNode
{
    BTData data;

    struct _bTreeNode * left;

    struct _bTreeNode * right;
} BTreeNode;
```



## Function modulation

```
BTreeNode * GetLeftSubTree(BTreeNode * bt);
```

```
BTreeNode * GetRightSubTree(BTreeNode * bt);
```

```
void MakeLeftSubTree(BTreeNode * main, BTreeNode * sub);
```

```
void MakeRightSubTree(BTreeNode * main, BTreeNode * sub);
```

## ex) Insert function

```
if(GetData(cNode) > data)
```

```
    cNode = GetLeftSubTree(cNode);
```

```
else
```

```
    cNode = GetRightSubTree(cNode);
```

```
}
```

```
nNode = MakeBTreeNode();
```

```
SetData(nNode, data);
```

```
if(pNode != NULL)
```

```
{
```

```
    if(data < GetData(pNode))
```

```
        MakeLeftSubTree(pNode, nNode);
```

```
    else
```

```
        MakeRightSubTree(pNode, nNode);
```

```
}
```