

204700

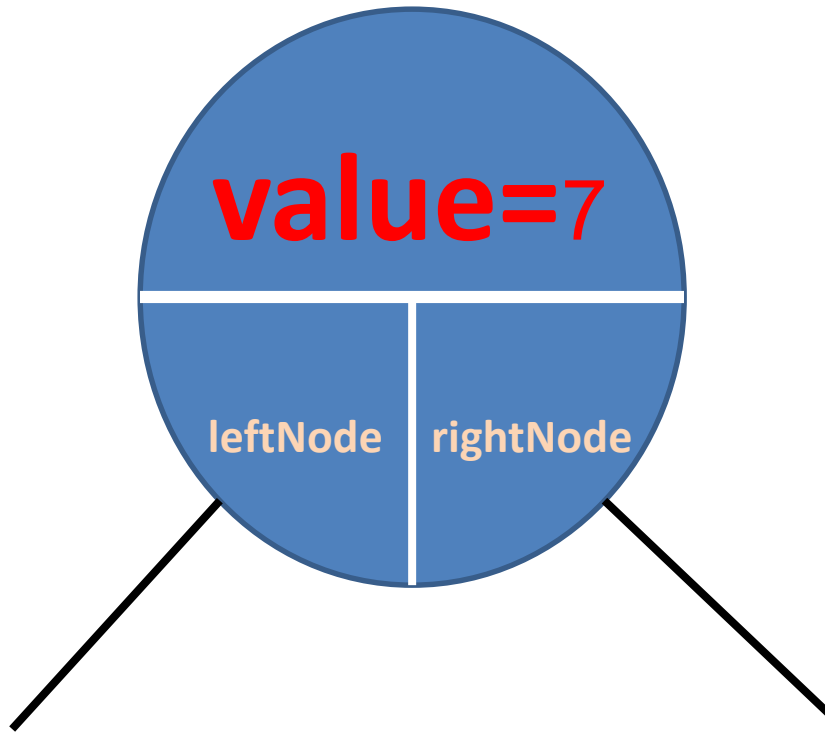
Data Structure and Programming Languages

Jakarin Chawachat

TREES

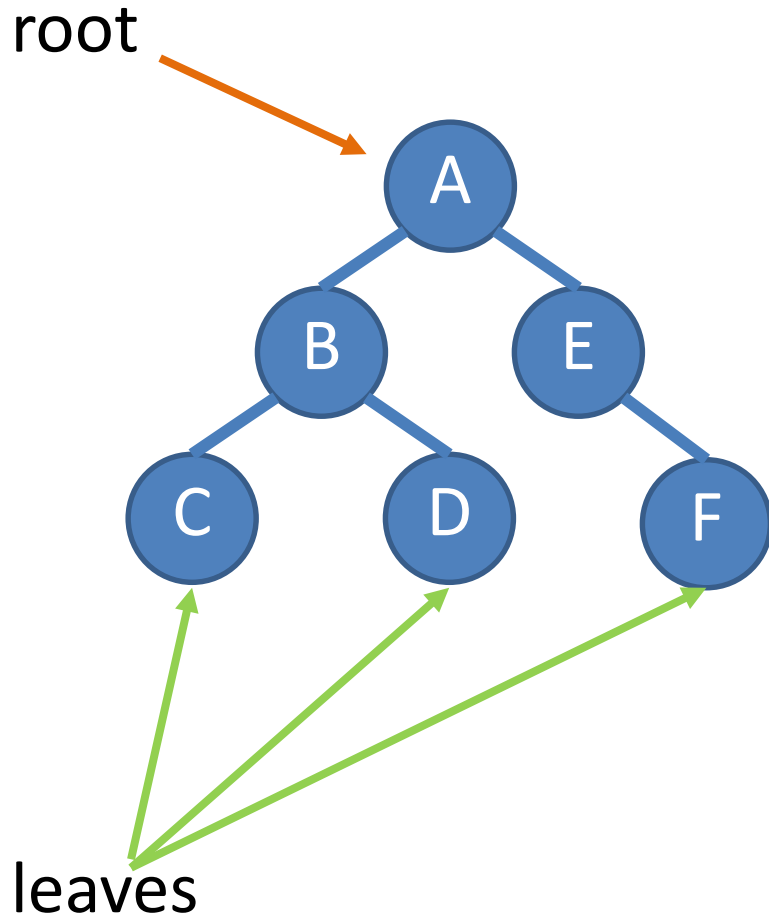
- Binary tree
 - Create a node
 - Insert a node
 - Find a key
 - Traversal

Node



```
public class Node {  
    int value;  
    Node leftNode;  
    Node rightNode;  
  
    Node() {  
    }  
  
    Node(int inputValue) {  
        value = inputValue;  
    }  
}
```

Class Tree



```
public class Tree {  
    Node root;  
    Node pointer;
```

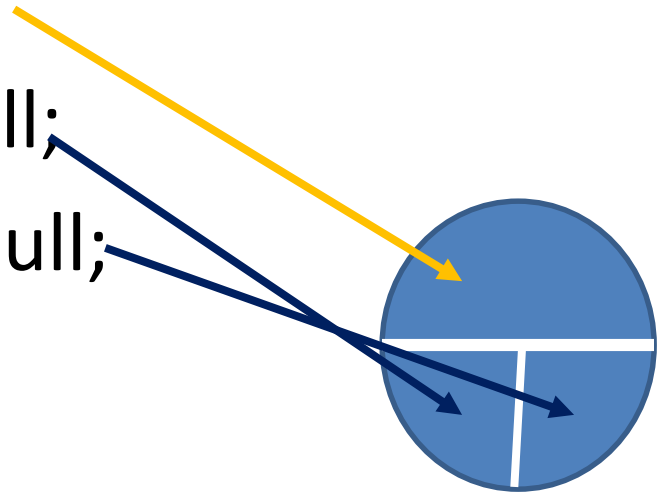
```
Tree() {  
    root=null;  
}
```

```
Tree(int item) {  
    root = new Node();  
    root.value = item;  
}
```

```
//OPERATIONS
```

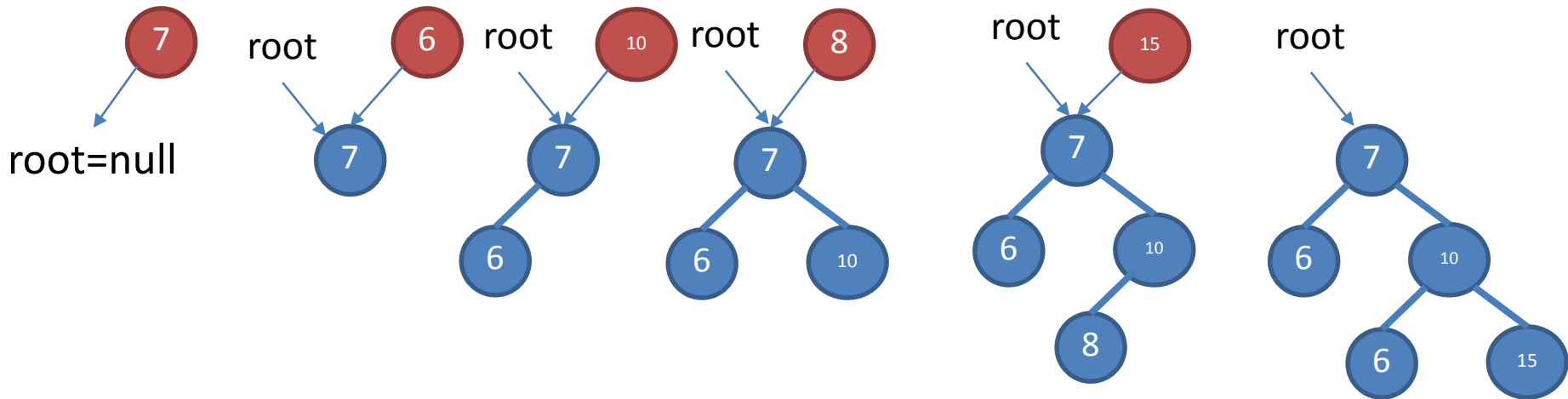
Create a node

```
Node createNode(int item) {  
    Node newNode = new Node();  
    newNode.value = item;  
    newNode.leftNode = null;  
    newNode.rightNode = null;  
    return newNode;  
}
```



รูปตัวอย่างการ insert node

- ลำดับการ insert node: 7, 6, 10, 8, 15



เริ่มต้น root = null

ผลลัพธ์สุดท้าย

```
void insertNewNode(Node newNode) {
    Node previous=null
    pointer=null;

    if (root == null) {
        root = newNode;
    } else {
        pointer=root;
        while(pointer!=null){
            previous = pointer;
            if(pointer.value>newNode.value){
                pointer=pointer.leftNode;
            }else{
                pointer=pointer.rightNode;
            }
        }

        if(previous.value>newNode.value){
            previous.leftNode=newNode;
        }else{
            previous.rightNode=newNode;
        }
    }
}
```

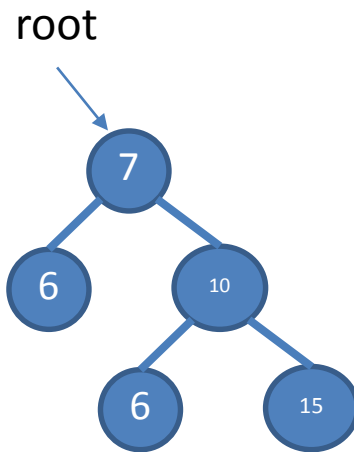
← หาโหนดสุดท้ายแล้วเก็บไว้ที่ **previous**

← เลือกตัวชี้ที่จะต้อง **link** ไปโหนดใหม่

Class Test

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    Tree t = new Tree(10);  
    Node x = new Node(5);  
    Node y = new Node(15);  
    t.insertNewNode(x);  
    t.insertNewNode(y);  
    System.out.println(t.root.value);  
    System.out.print(t.root.leftNode.value);  
    System.out.print(t.root.rightNode.value);  
}
```

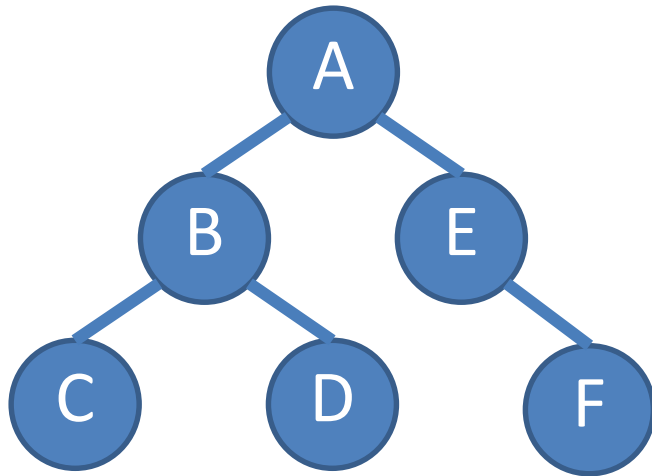

FindKey



FindKey(20)
FindKey(10)

```
void FindKey(int key,Node v){
    if(v==null){
        System.out.println("Not found.");
    }
    if(key<v.value){
        if(v.leftNode==null){
            System.out.println("Not found.");
        }else{
            FindKey(key,v.leftNode);
        }
    }else if(key==v.value){
        System.out.println("Found.");
    }else{
        if(v.rightNode==null){
            System.out.println("Not found.");
        }else{
            FindKey(key,v.rightNode);
        }
    }
}
```

Binary Tree Traversal



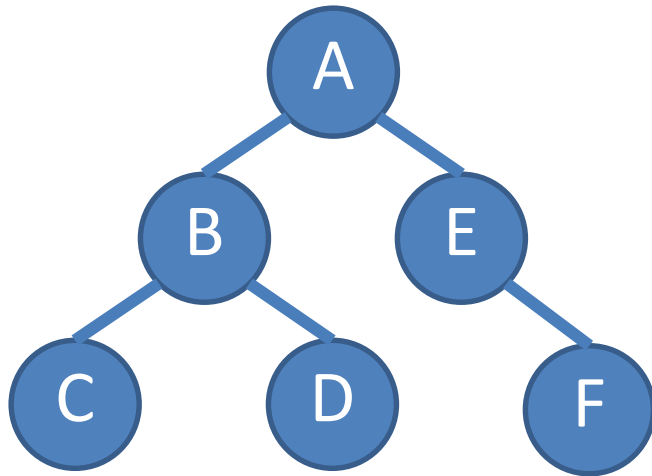
preOrder

```
void preOrder(Node p){  
    System.out.println(p.value);  
    if(p.leftNode!=null){  
        preOrder(p.leftNode);  
    }  
    if(p.rightNode!=null){  
        preOrder(p.rightNode);  
    }  
}
```

ผลลัพธ์: $RT_L T_R$

A B C D E F

Binary Tree Traversal

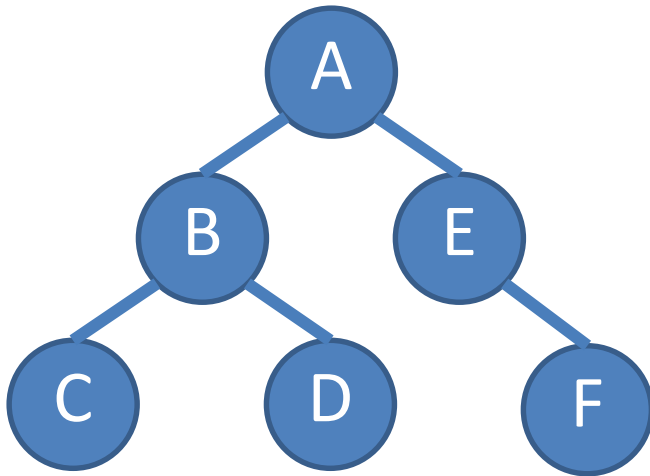


postOrder

```
void postOrder(Node p){  
    if(p.leftNode!=null){  
        postOrder(p.leftNode);  
    }  
    if(p.rightNode!=null){  
        postOrder(p.rightNode);  
    }  
    System.out.println(p.value);  
}
```

ผลลัพธ์: $T_L T_R R$
C D B F E A

Binary Tree Traversal



inOrder

InOrder ??

ผลลัพธ์: $T_L R T_R$

C B D A E F