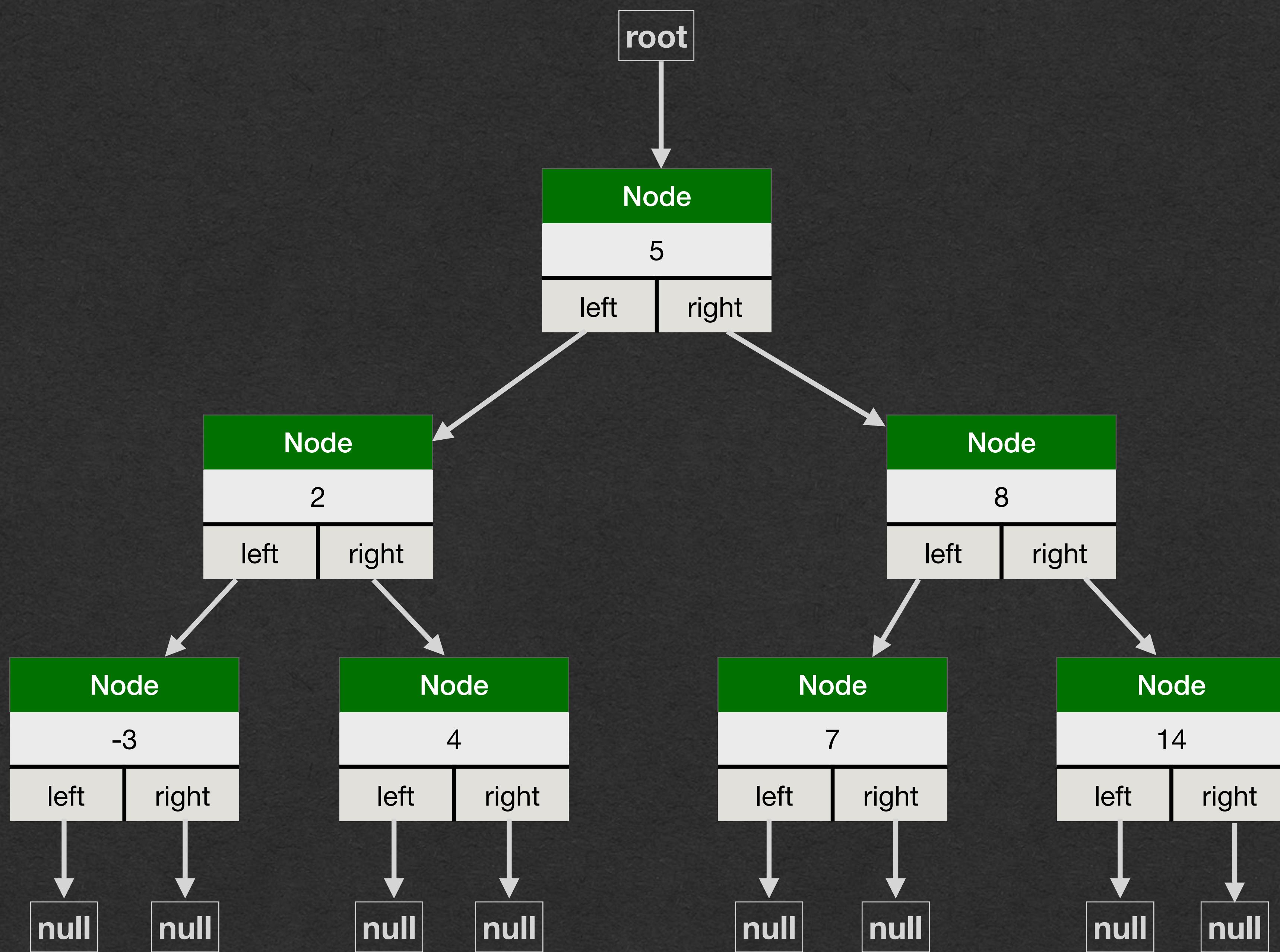
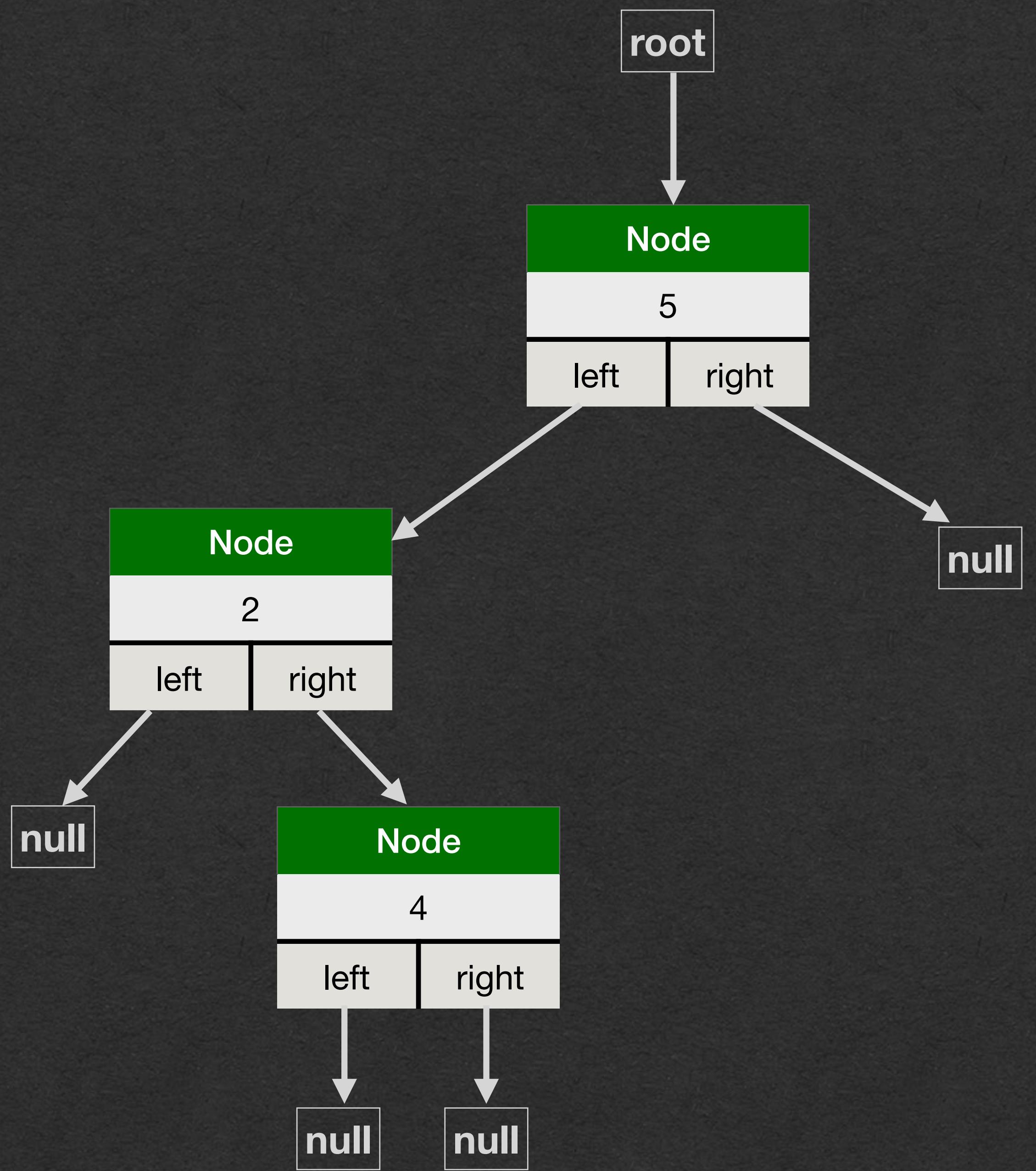


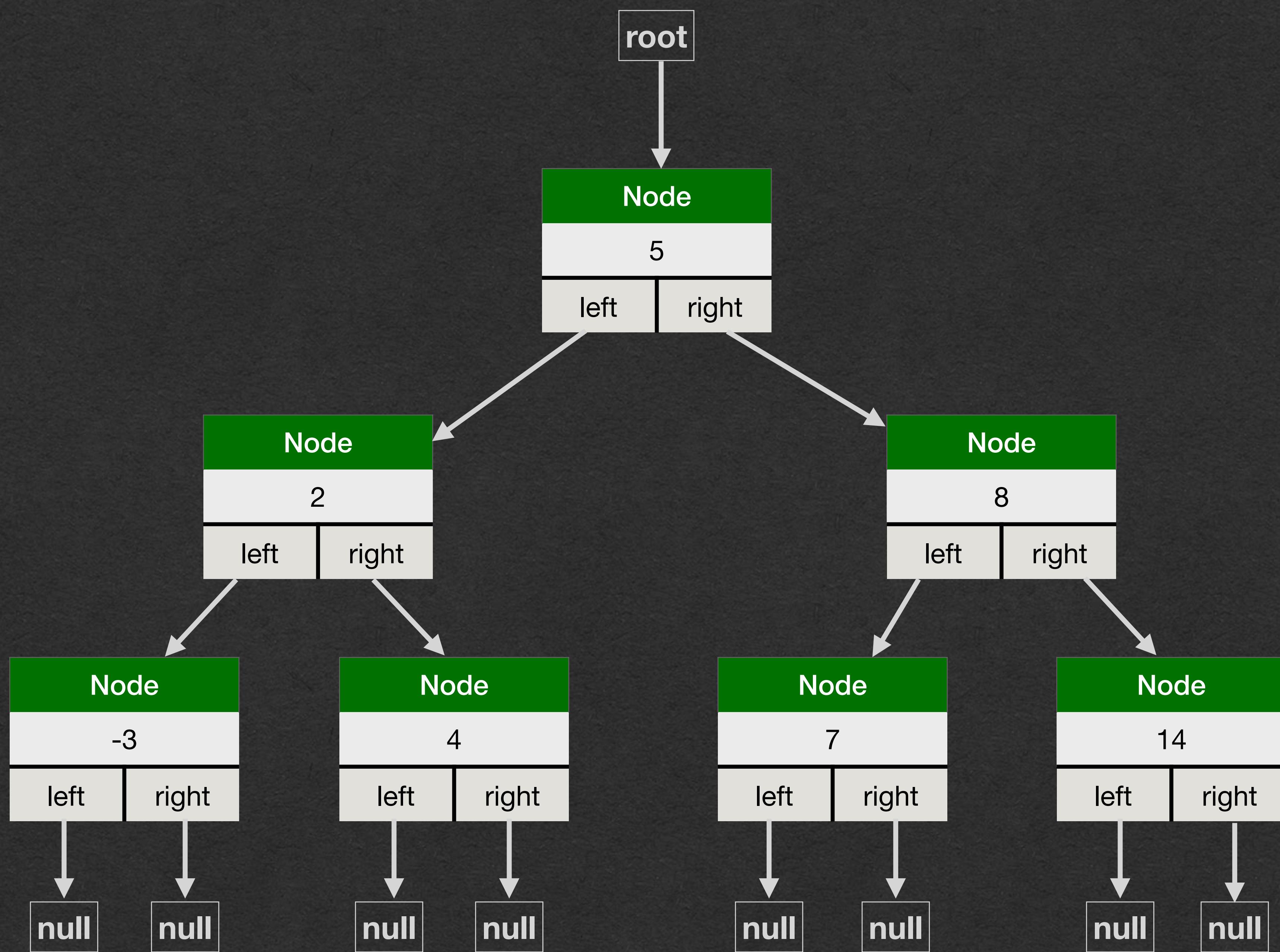
Binary Trees and Traversals

Binary Trees

- Similar in structure to Linked List
 - Consists of Nodes
 - A Tree is only a reference to the first node
(Called the root node)
 - Trees have 2 references to nodes
 - Each node has left and right reference
 - Vocab: These are called its child nodes
 - Vocab: The node is the parent to these children







```
public class BinaryTreeNode<A> {  
    private A value;  
    private BinaryTreeNode<A> left;  
    private BinaryTreeNode<A> right;  
  
    public BinaryTreeNode(A value, BinaryTreeNode<A> left, BinaryTreeNode<A> right) {  
        this.value = value;  
        this.right = right;  
        this.left = left;  
    }  
}
```

```
BinaryTreeNode<Integer> root = new BinaryTreeNode<>(5, null, null);  
root.left = new BinaryTreeNode<>(2, null, null);  
root.right = new BinaryTreeNode<>(8, null, null);  
root.left.left = new BinaryTreeNode<>(-3, null, null);  
root.left.right = new BinaryTreeNode<>(4, null, null);  
root.right.left = new BinaryTreeNode<>(7, null, null);  
root.right.right = new BinaryTreeNode<>(14, null, null);
```

- Binary Tree Nodes are very similar in structure to Linked List Nodes
- No simple prepend or append so we'll manually build a tree by setting left and right directly
 - (All this code is in the same class so accessing private instance variables is allowed)

Tree Traversals

- How do we compute with trees?
 - With linked lists we wrote several methods that recursively visited the next node to visit every value
 - With trees, how do we visit both children of each node?
 - Recursive call on **both** child nodes!
 - We'll see 3 different approaches
 - Pre-Order Traversal
 - In-Order Traversal
 - Post-Order Traversal

Tree Traversals

- In-Order Traversal
 - Call in-order on the left child
 - Visit the node's value
 - Call in-order on the right child
- This traversal converts the all values of the tree to a single string (effectively `toString`)

```
public String inOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += inOrderTraversal(node.left);  
        out += node.value.toString() + " ";  
        out += inOrderTraversal(node.right);  
        return out;  
    } else {  
        return "";  
    }  
}
```

```
System.out.println(root.inOrderTraversal(root));
```

Tree Traversals

- Pre-Order Traversal
 - Visit the node's value
 - Call pre-order on the left child
 - Call pre-order on the right child

```
public String preOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += node.value.toString() + " ";  
        out += preOrderTraversal(node.left);  
        out += preOrderTraversal(node.right);  
        return out;  
    } else {  
        return "";  
    }  
}
```

```
System.out.println(root.preOrderTraversal(root));
```

Tree Traversals

- Post-Order Traversal
 - Call post-order on the left child
 - Call post-order on the right child
 - Visit the node's value

```
public String postOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += postOrderTraversal(node.left);  
        out += postOrderTraversal(node.right);  
        out += node.value.toString() + " ";  
        return out;  
    } else {  
        return "";  
    }  
}
```

```
System.out.println(root.postOrderTraversal(root));
```

The Code

```
public String inOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += inOrderTraversal(node.left);  
        out += node.value.toString() + " ";  
        out += inOrderTraversal(node.right);  
        return out;  
    } else {  
        return "";  
    }  
}  
  
public String preOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += node.value.toString() + " ";  
        out += preOrderTraversal(node.left);  
        out += preOrderTraversal(node.right);  
        return out;  
    } else {  
        return "";  
    }  
}  
  
public String postOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += postOrderTraversal(node.left);  
        out += postOrderTraversal(node.right);  
        out += node.value.toString() + " ";  
        return out;  
    } else {  
        return "";  
    }  
}
```

The Code

- Challenge: Write these without recursion

```
public String inOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += inOrderTraversal(node.left);  
        out += node.value.toString() + " ";  
        out += inOrderTraversal(node.right);  
        return out;  
    } else {  
        return "";  
    }  
}  
  
public String preOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += node.value.toString() + " ";  
        out += preOrderTraversal(node.left);  
        out += preOrderTraversal(node.right);  
        return out;  
    } else {  
        return "";  
    }  
}  
  
public String postOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += postOrderTraversal(node.left);  
        out += postOrderTraversal(node.right);  
        out += node.value.toString() + " ";  
        return out;  
    } else {  
        return "";  
    }  
}
```

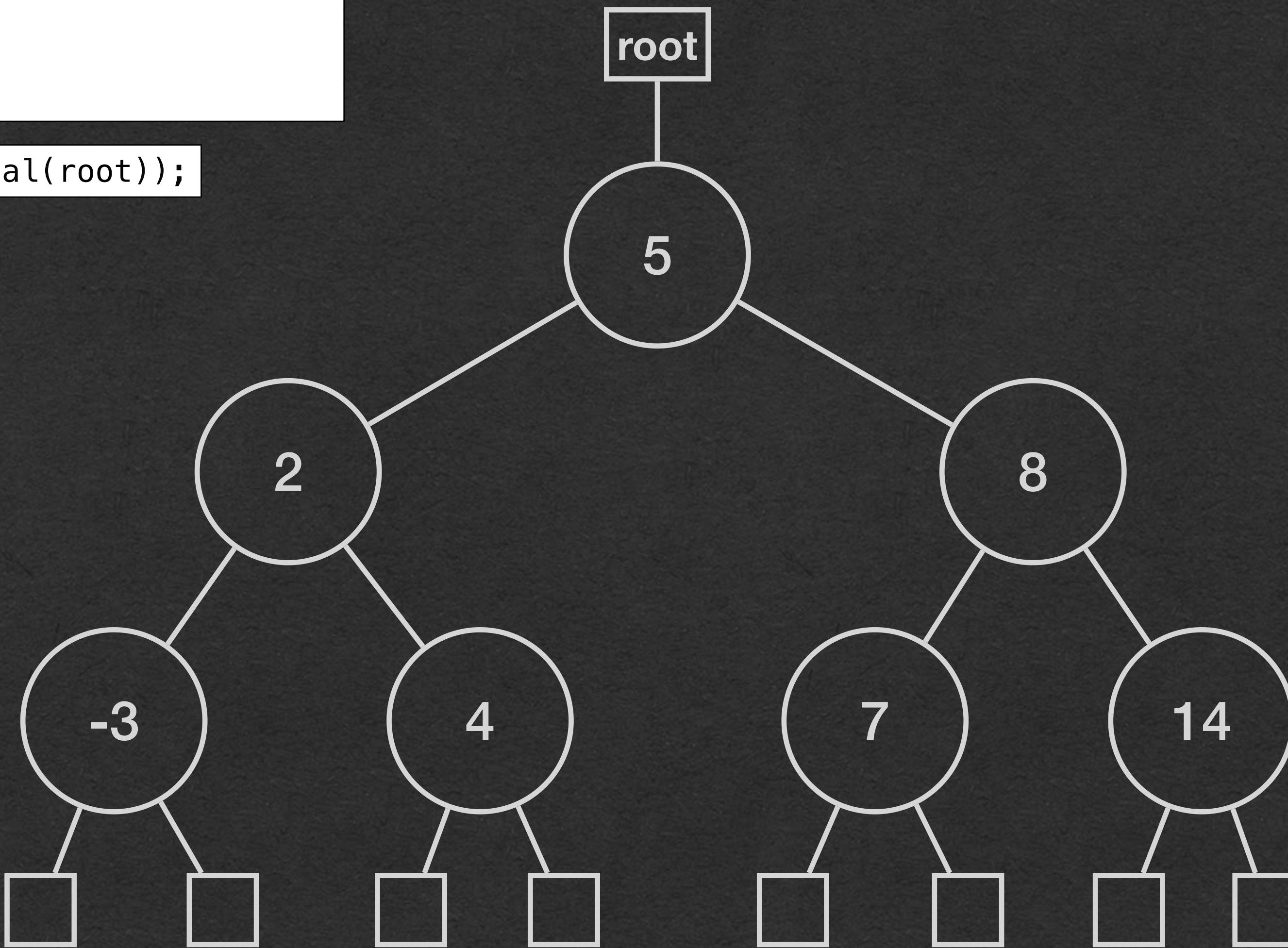
```
public String inOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += inOrderTraversal(node.left);  
        out += node.value.toString() + " ";  
        out += inOrderTraversal(node.right);  
        return out;  
    } else {  
        return "";  
    }  
}
```

```
System.out.println(root.inOrderTraversal(root));
```

In/Out

-3 2 4 5 7 8 14

in order



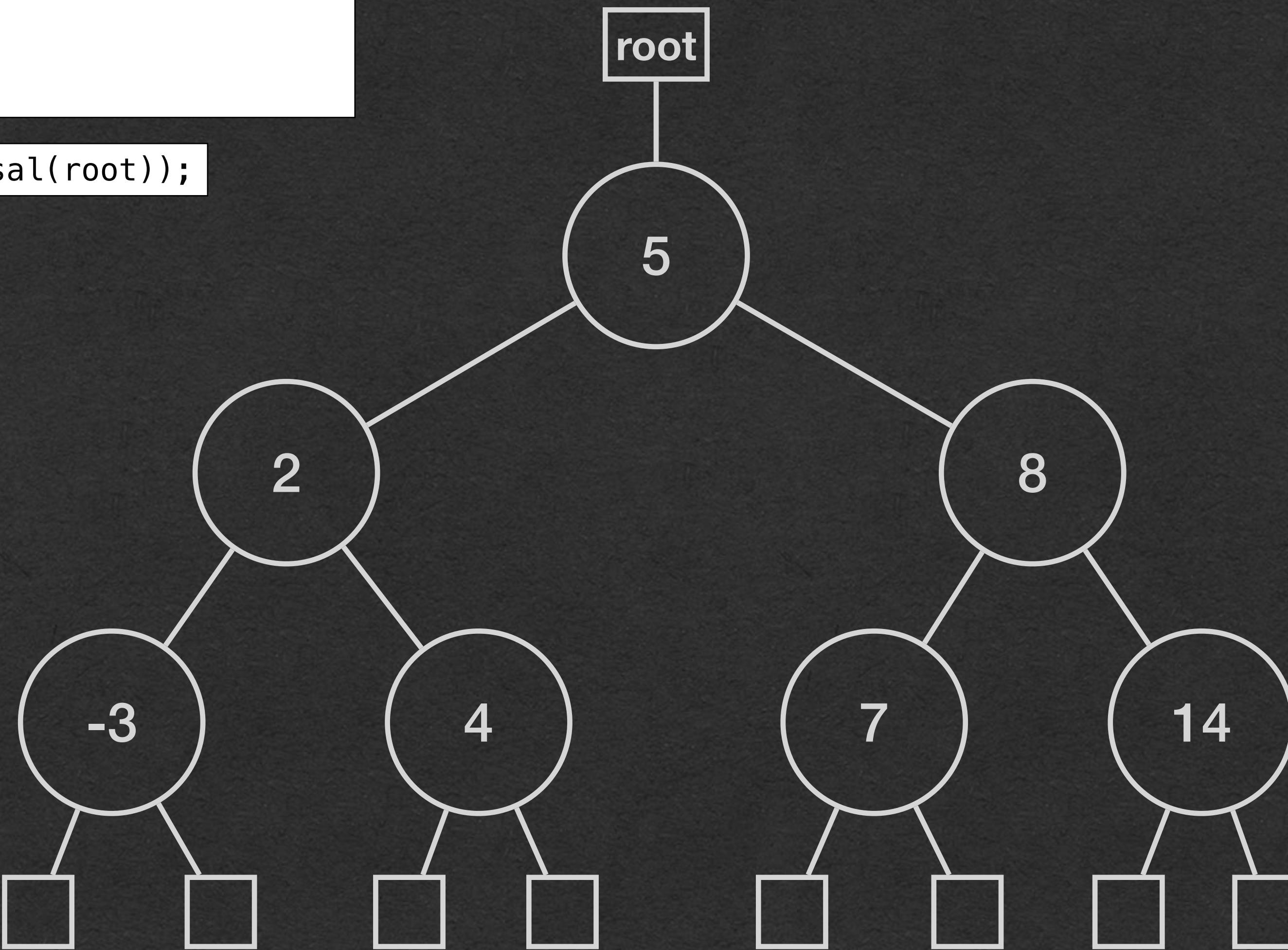
```
public String preOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += node.value.toString() + " ";  
        out += preOrderTraversal(node.left);  
        out += preOrderTraversal(node.right);  
        return out;  
    } else {  
        return "";  
    }  
}
```

```
System.out.println(root.preOrderTraversal(root));
```

In/Out

5 2 -3 4 8 7 14

pre order



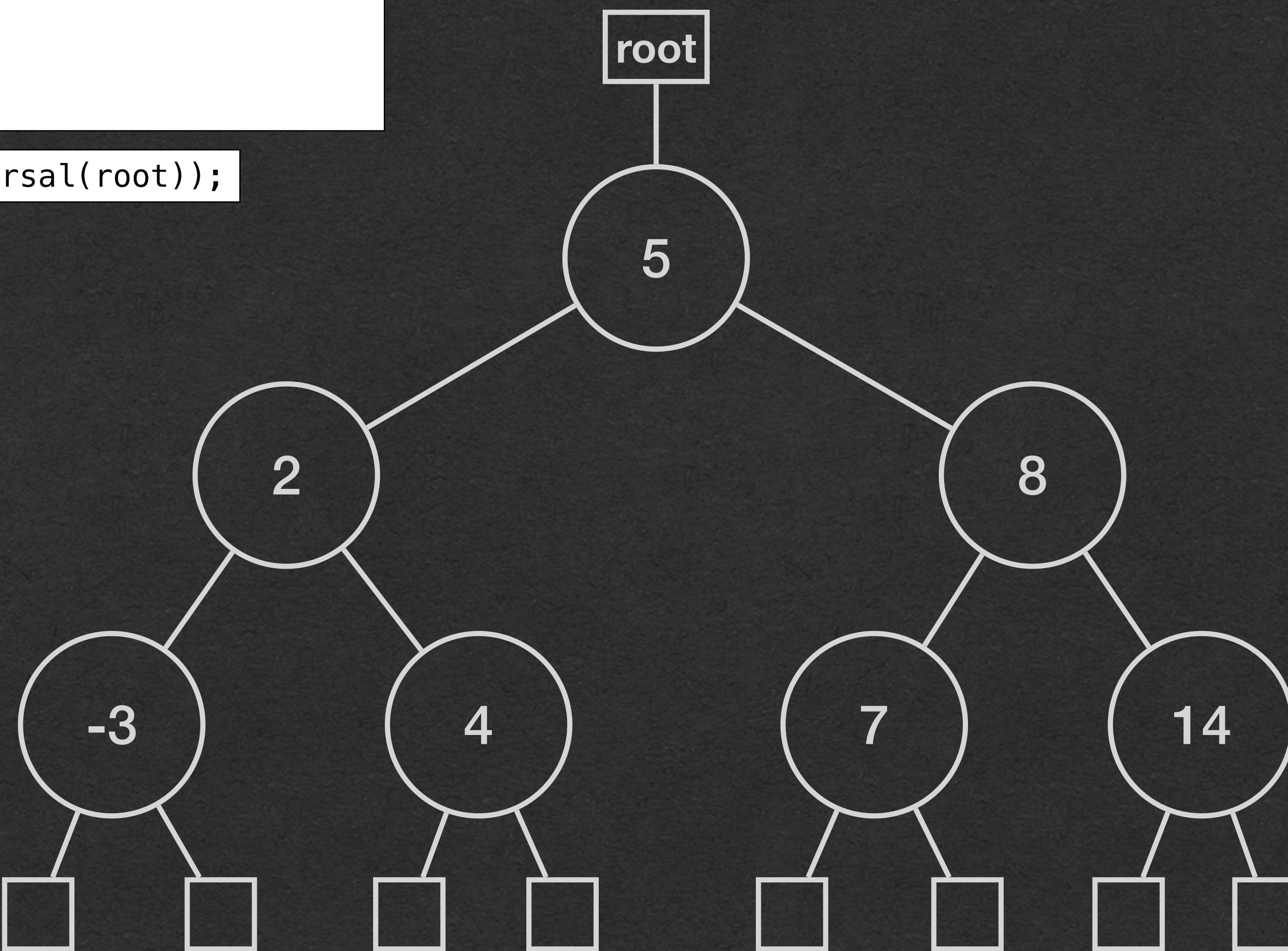
```
public String postOrderTraversal(BinaryTreeNode<A> node) {  
    if (node != null) {  
        String out = "";  
        out += postOrderTraversal(node.left);  
        out += postOrderTraversal(node.right);  
        out += node.value.toString() + " ";  
        return out;  
    } else {  
        return "";  
    }  
}
```

```
System.out.println(root.postOrderTraversal(root));
```

In/Out

-3 4 2 7 14 8 5

post order



Tree Memory Diagram

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
} → public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```

in/out

- Build a tree
- Visit every node with a post-order traversal

Stack

Name	Value
------	-------

Heap

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
}  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```

day

in/out

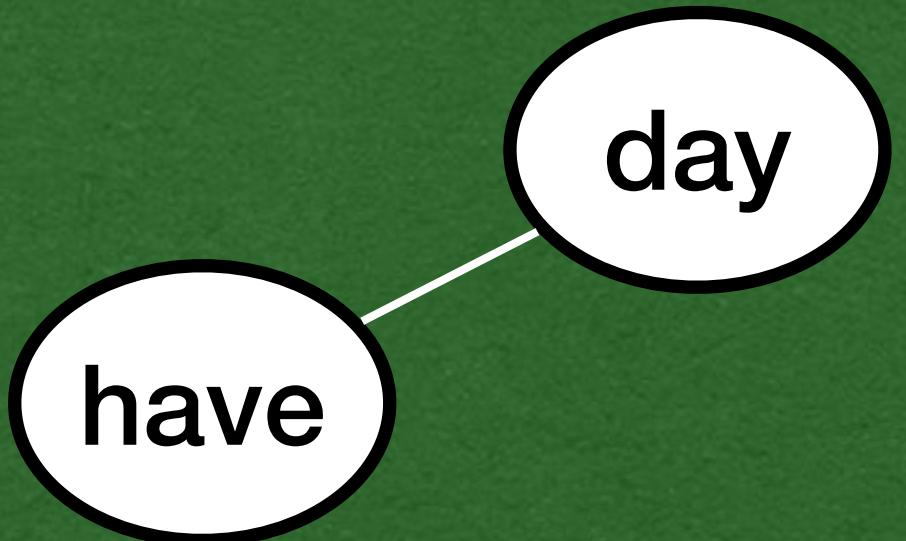
- Create the root node with the value "day"

Stack		Heap
Name	Value	
root	0x350	
this	0x350	
value	"day"	
left	null	
right	null	
BTNode		
value	"day"	
left	null	
right	null	
	0x350	

Stack		Heap
Name	Value	
root	0x350	
this	0x350	
value	"day"	
left	null	
right	null	
BTNode		
value	"day"	
left	null	
right	null	
	0x350	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



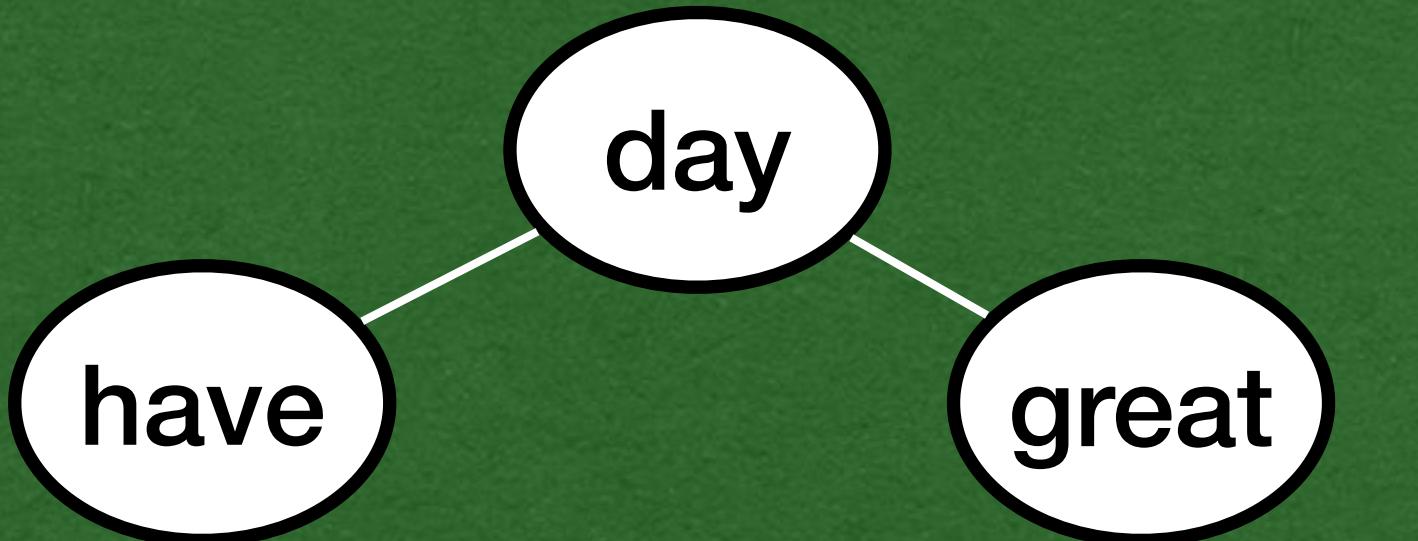
in/out

- Set the left child of the root to a node with "have"

Stack		Heap
Name	Value	
BTNode	root this value left right	0x350 0x350 "day" null null
	this value left right	0x200 "have" null null
BTNode	value left right	0x350 null null
	value left right	0x200 "have" null null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
}  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



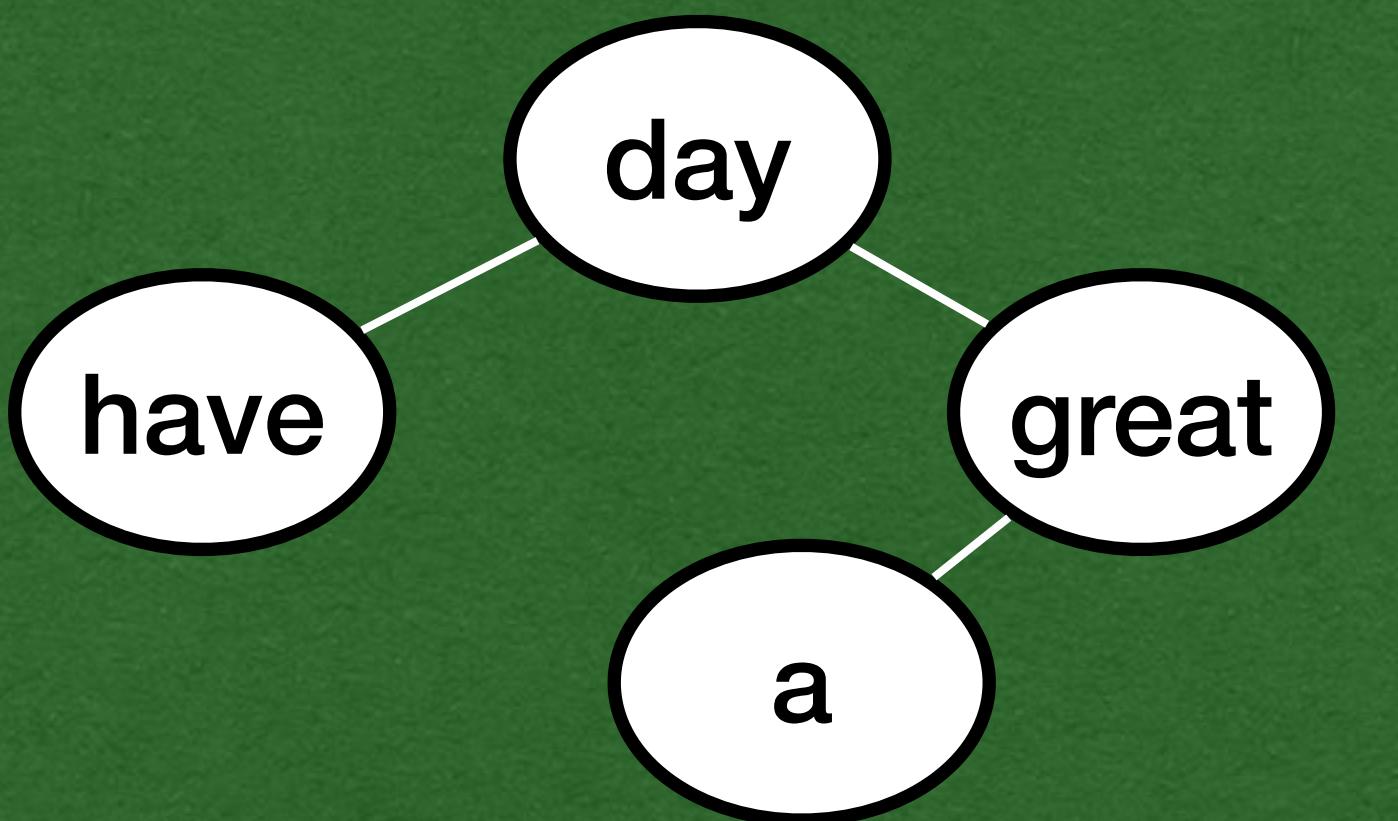
in/out

- Set the right child of the root to a node with "great"

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
	0x200	
	0x350	
	0x480	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
}  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



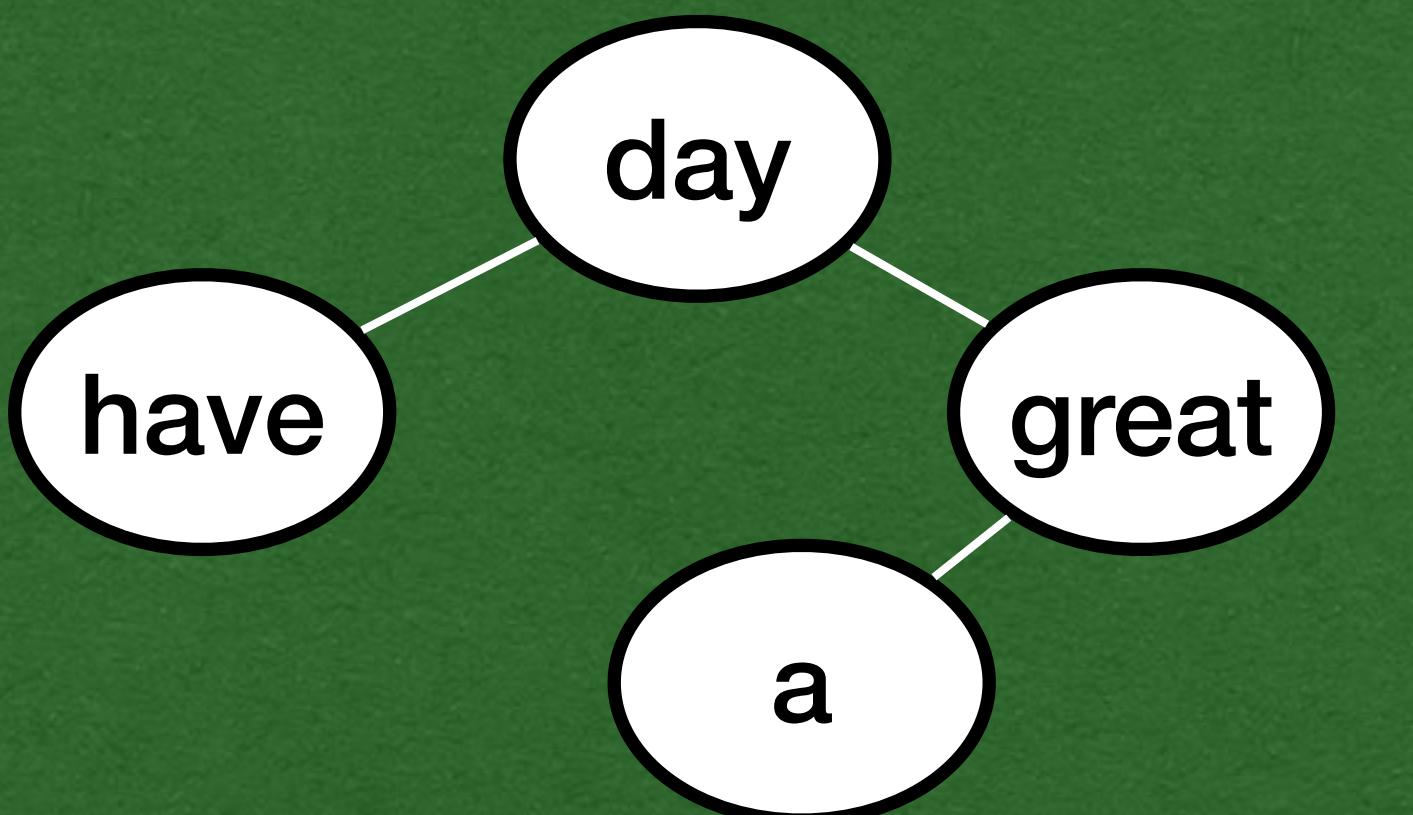
in/out

- Set the left child of the right child of the root to a node with "a"

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
	0x200	
	0x480	
	0x936	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
}  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



in/out

- Time to start the post-order traversal

Stack

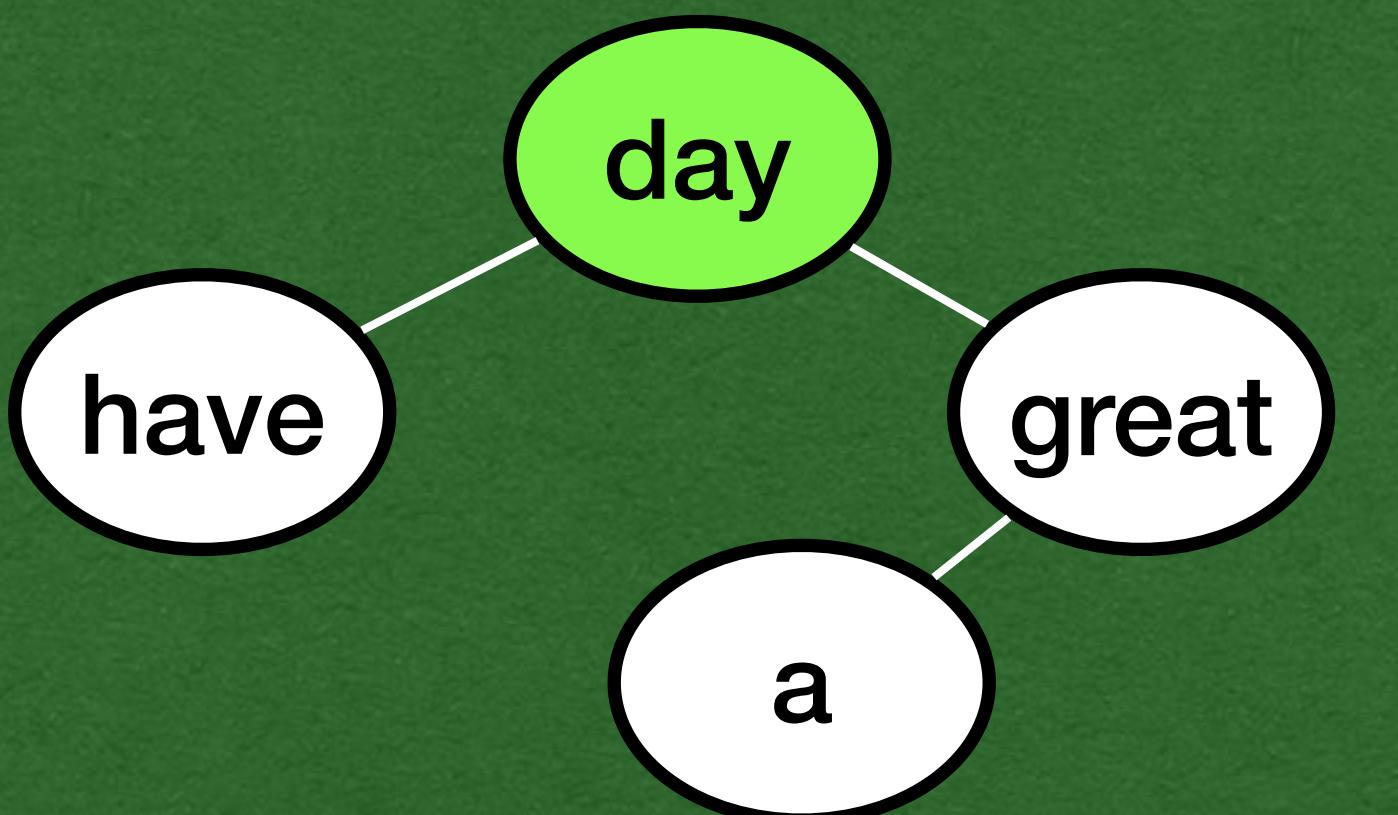
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null

Heap

BTNode
value "day"
left null 0x200
right null 0x480 0x350
BTNode
value "have"
left null
right null
0x200
BTNode
value "great"
left null
right null
0x480
BTNode
value "a"
left null 0x936
right null 0x480 0x936
0x936

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
}  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



in/out

- Initial call of traversal is on the root node (0x350)

Stack

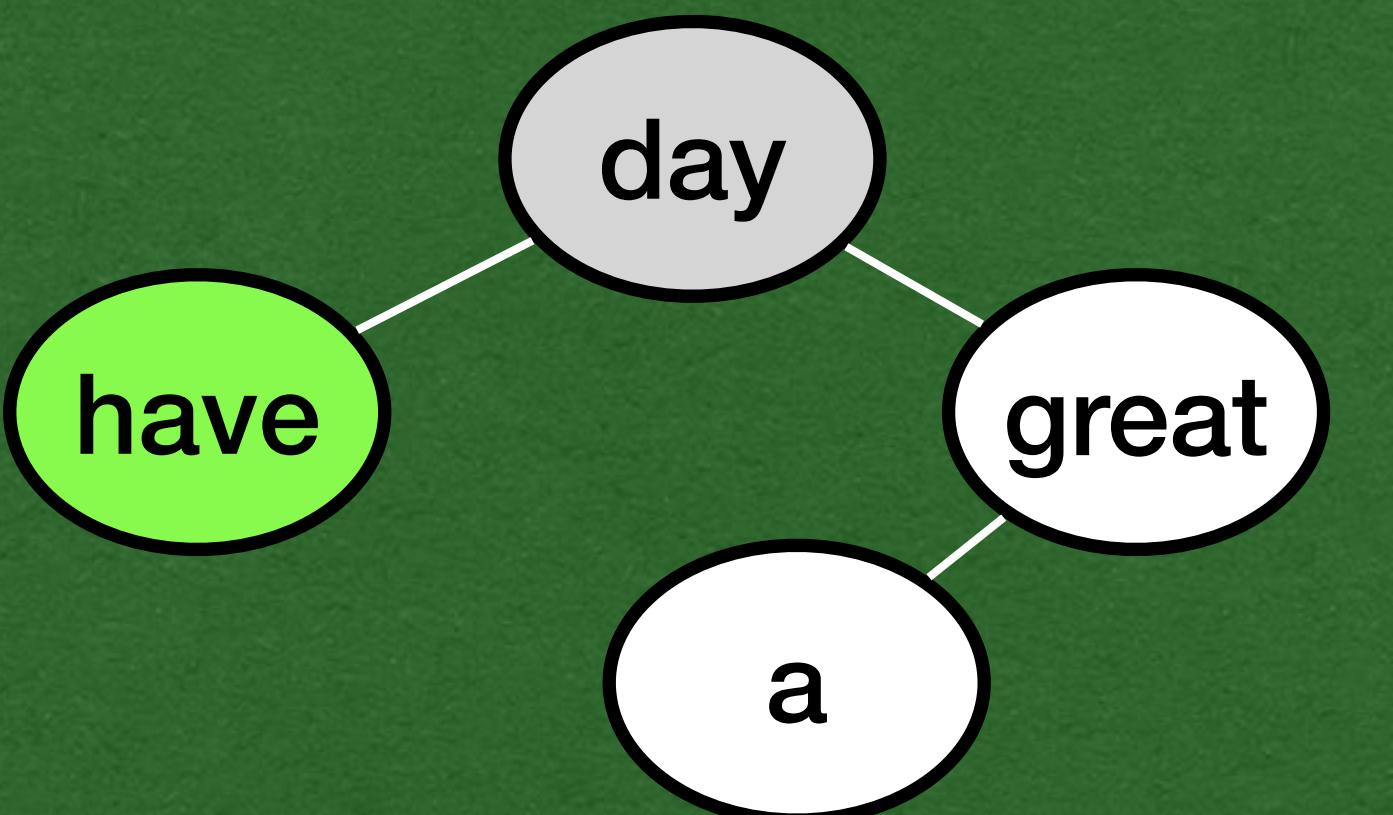
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
traversal	0x350
node	0x350

Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"great"
left	null
right	null
0x480	
BTNode	
value	"a"
left	null 0x936
right	null
0x936	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



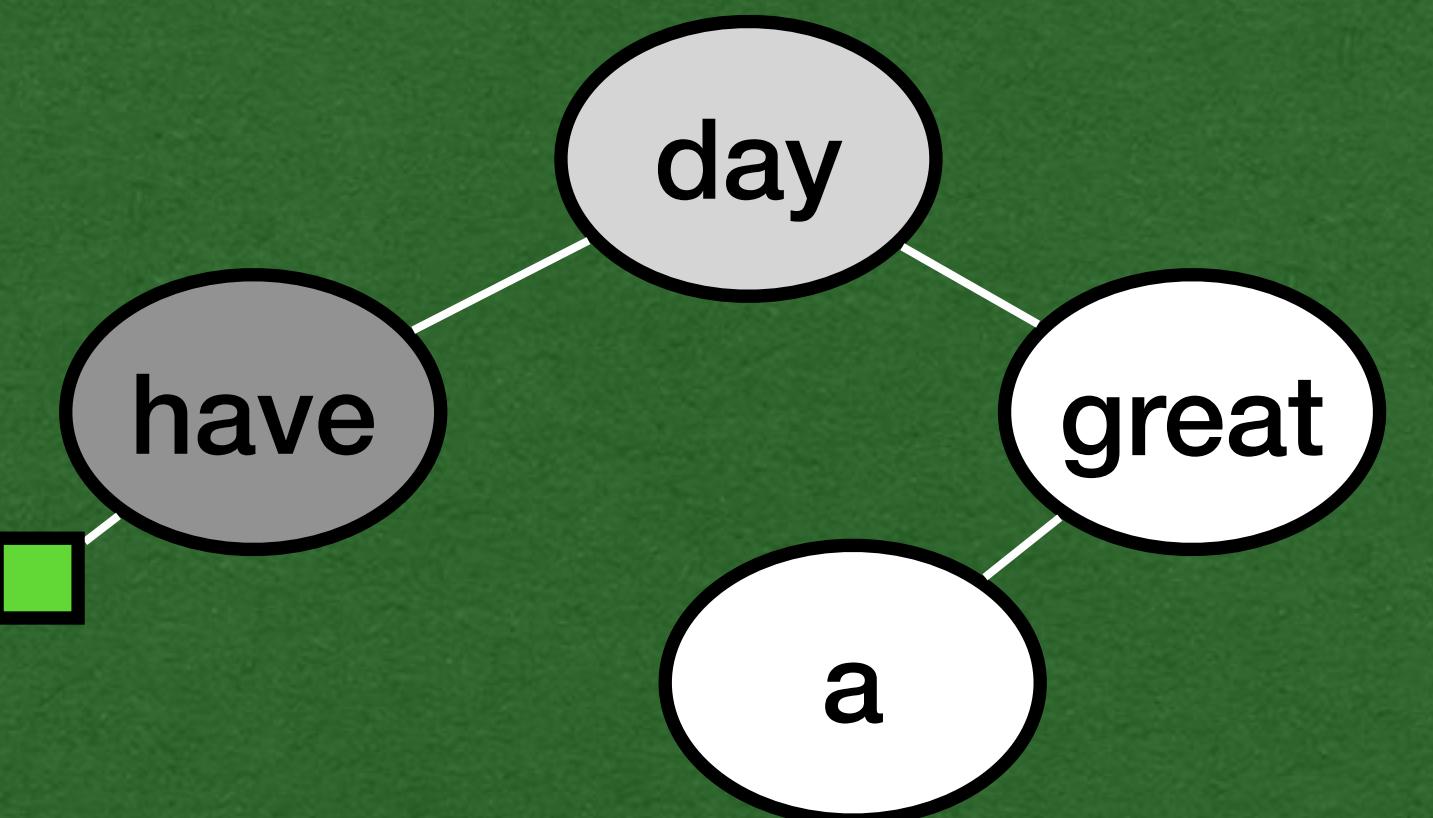
in/out

- Make a recursive call on the left child first

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "great" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "a" left null right null
this	0x200	traversal
value	"have"	traversal
left	null	node
right	null	node
traversal	0x350	BTNode value "great" left null right null
traversal	0x200	0x480
node	0x350	BTNode value "a" left null right null
node	0x200	0x936

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
    public static void main(String[] args) {  
        BTNode<String> root = new BTNode<>("day", null, null);  
        root.left = new BTNode<>("have", null, null);  
        root.right = new BTNode<>("great", null, null);  
        root.right.left = new BTNode<>("a", null, null);  
        root.traversal(root);  
    }  
}
```



in/out

- Left node makes another recursive call on its left child

Stack

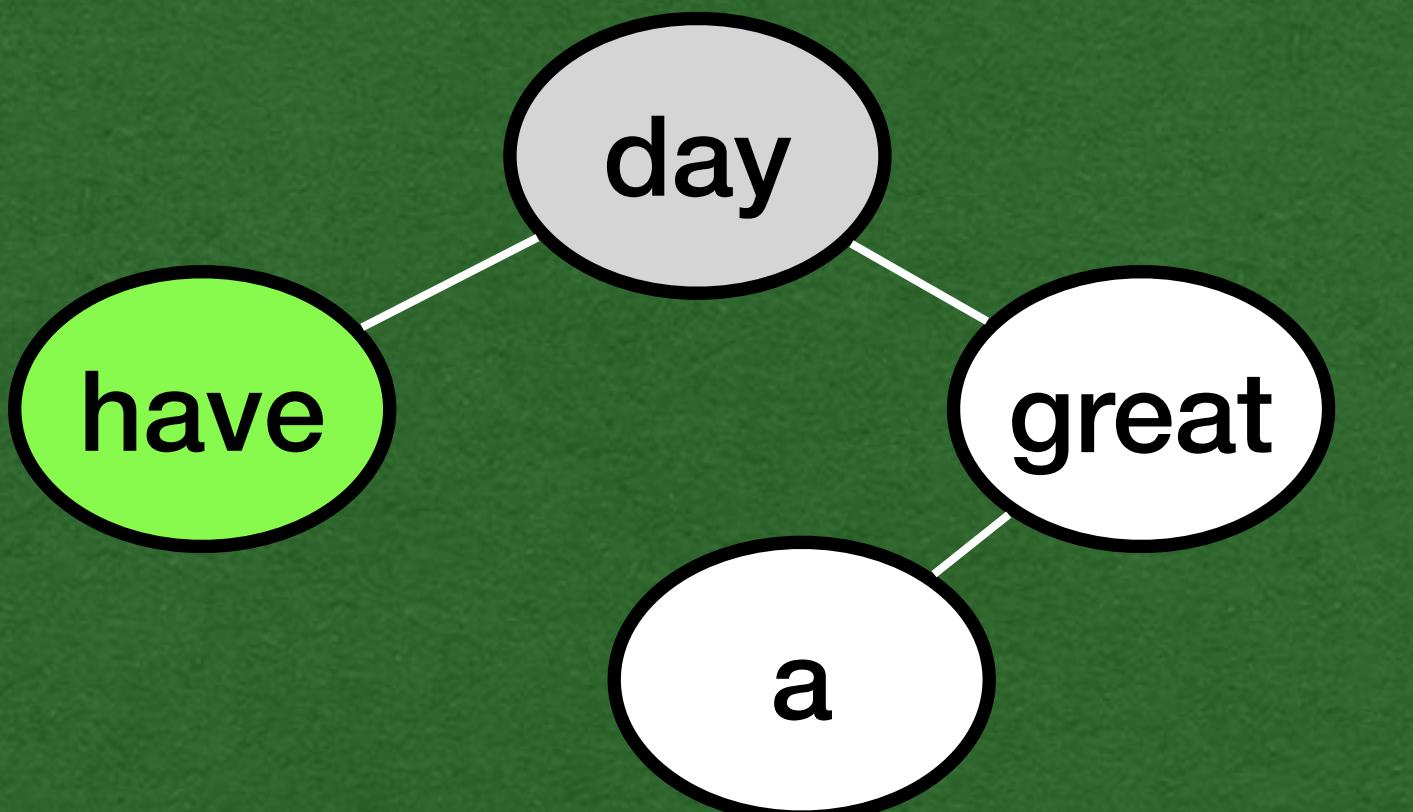
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null

Heap

BTNode
value "day"
left null 0x200
right null 0x480 0x350
BTNode
value "have"
left null
right null
0x200
BTNode
value "great"
left null
right null
0x480
BTNode
value "a"
left null 0x936
right null 0x480 0x200 0x350
0x936

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
}  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



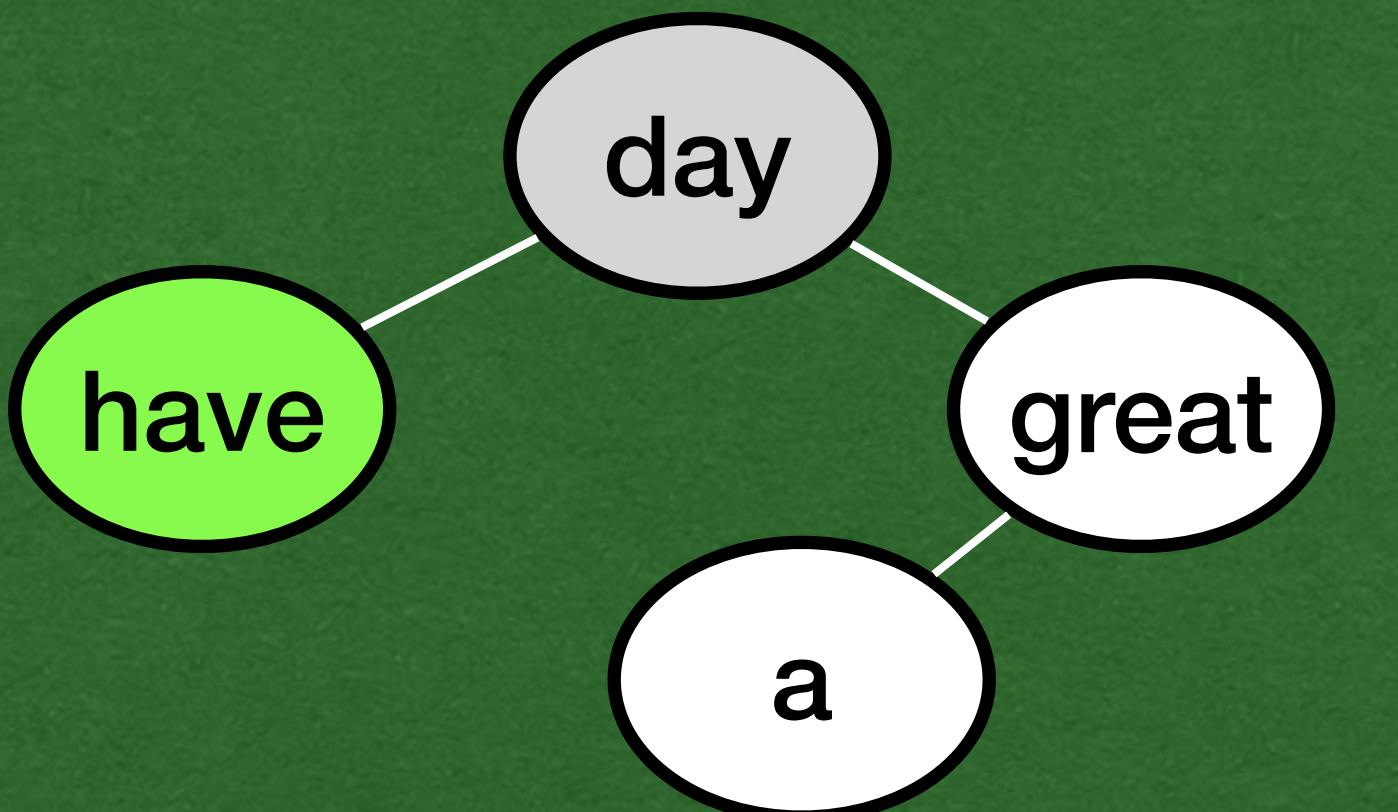
in/out

- A null node is our base case
- Do nothing and return to the previous frame on the stack

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "great" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "great" left null right null
this	0x200	BTNode value "a" left null right null
value	"have"	BTNode value "a" left null right null
left	null	BTNode value "great" left null right null
right	null	BTNode value "great" left null right null
traversal	0x936	BTNode value "a" left null right null
node	0x350	BTNode value "a" left null right null
node	0x200	BTNode value "great" left null right null
traversal	0x480	BTNode value "a" left null right null
node	null	BTNode value "a" left null right null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



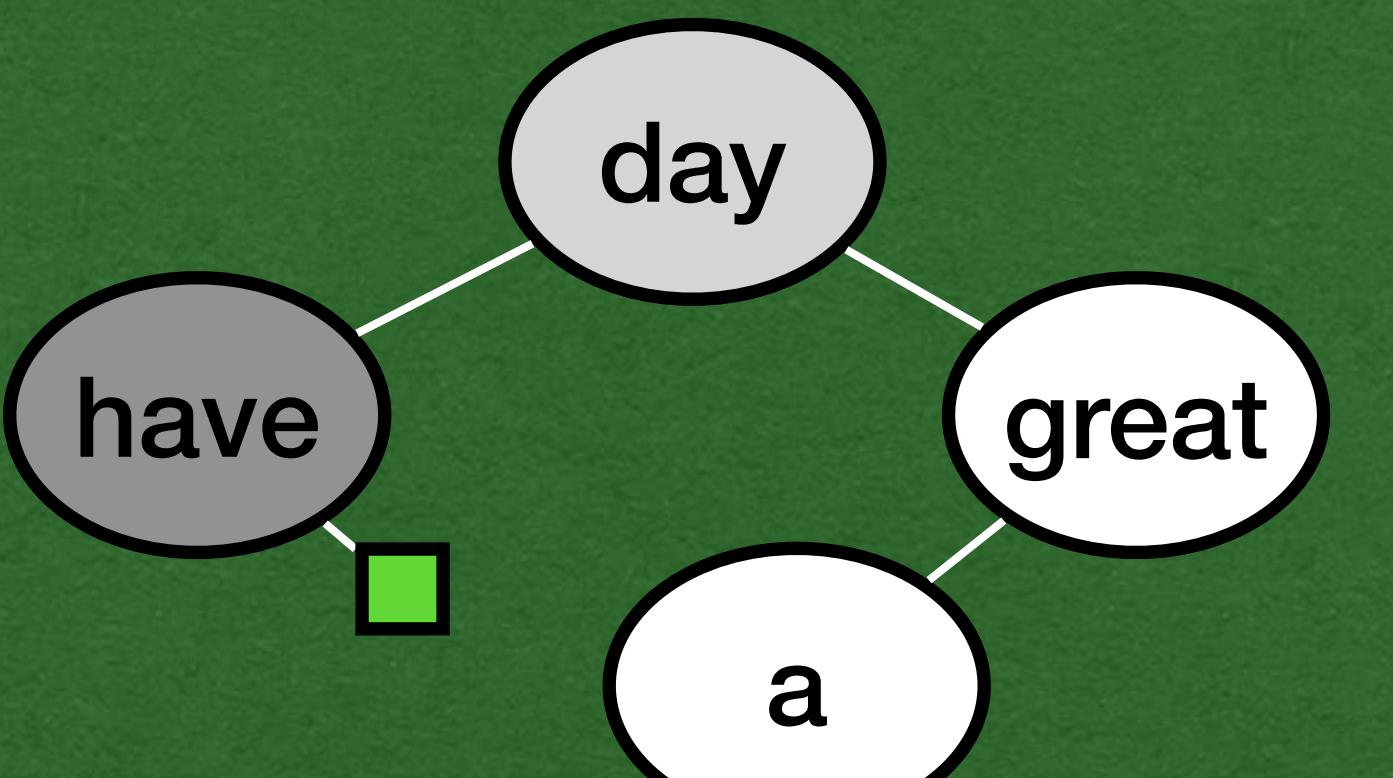
in/out

- This frame "remembers" that it made a left recursive call and needs to make the right recursive call

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "great" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "great" left null right null
this	0x200	BTNode value "a" left null right null
value	"have"	BTNode value "a" left null right null
left	null	BTNode value "great" left null right null
right	null	BTNode value "a" left null right null
traversal	0x350	BTNode value "a" left null right null
traversal	0x200	BTNode value "great" left null right null
traversal	null	BTNode value "a" left null right null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



in/out

- Right recursive call is also for a null node

Stack

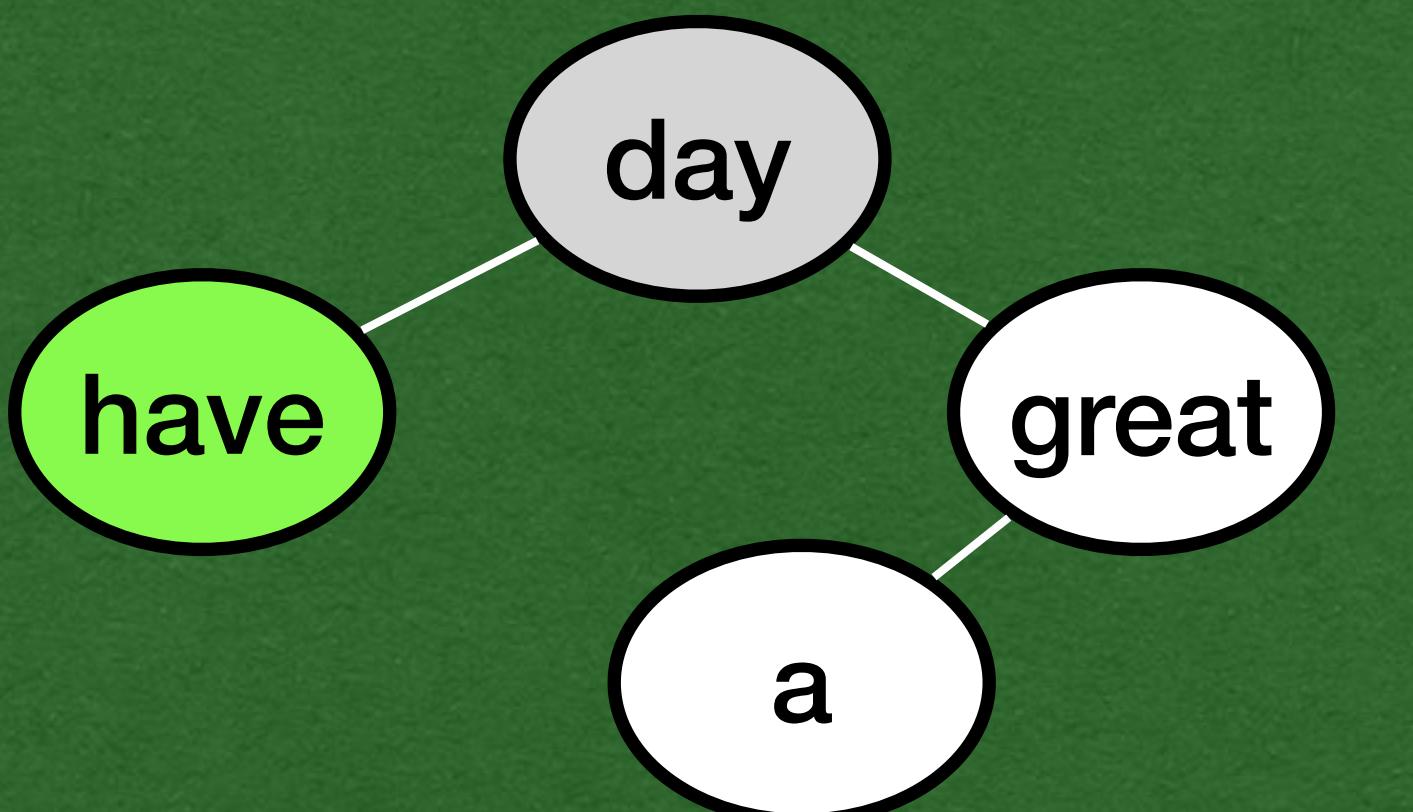
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null

Heap

BTNode
value "day"
left null 0x200
right null 0x480
0x350
BTNode
value "have"
left null
right null
0x200
BTNode
value "great"
left null
right null
0x480
BTNode
value "a"
left null 0x936
right null
0x936
BTNode
value "a"
left null
right null
0x936

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



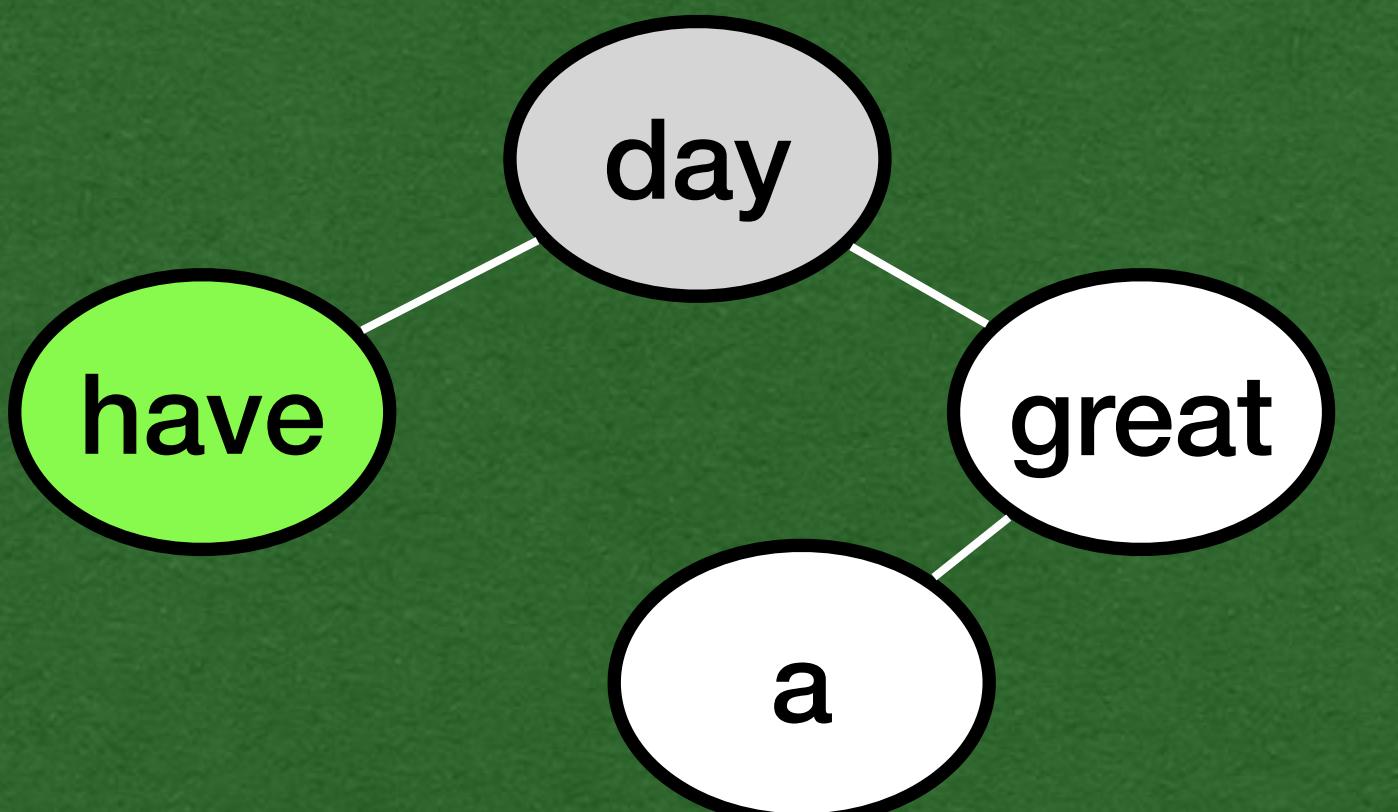
in/out

- Return to the previous frame again
- "Remembers" that it made the right recursive call

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	
node	0x200	BTNode
node	null	value "have"
node	null	left null
traversal	0x480	right null
traversal	0x936	
traversal		BTNode
traversal		value "great"
traversal		left null
traversal		right null
traversal		

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



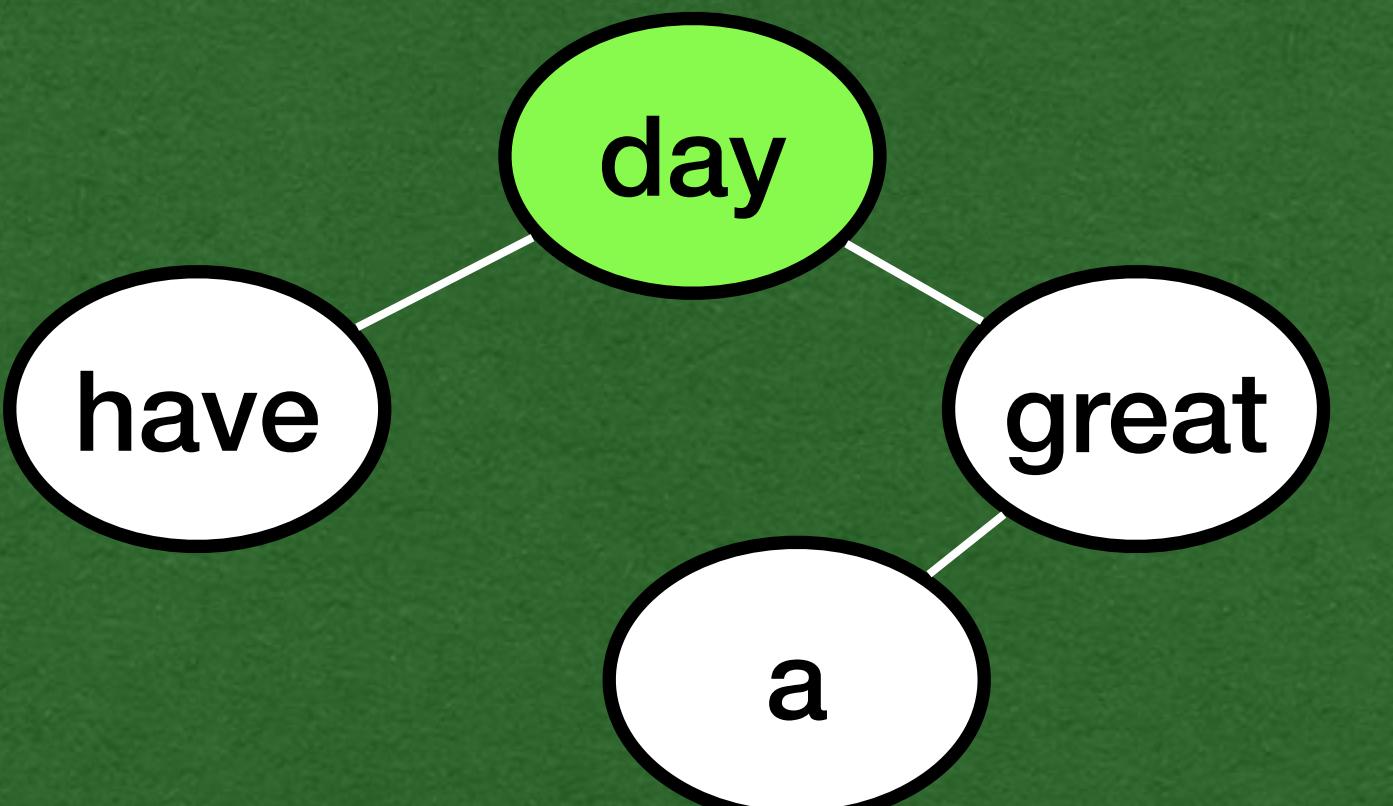
in/out
have

- Print "have" to the screen
- This stack frame is done and returns

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	0x480
node	null	
node	null	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
}  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



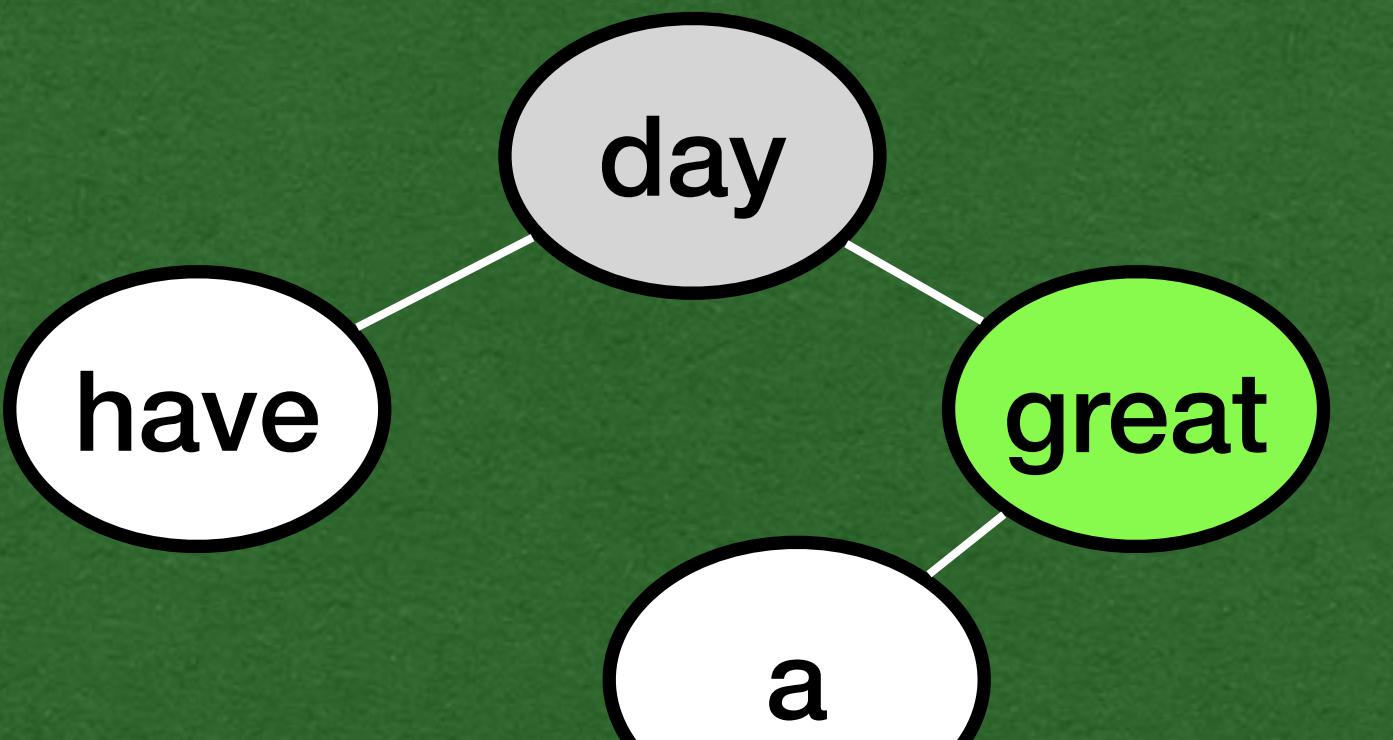
in/out
have

- First recursive call finally regains control (is at the top of the stack)

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	BTNode
node	null	value "have"
node	null	left null
traversal	0x350	right null
traversal	0x200	0x480
traversal	null	BTNode
traversal	null	value "great"
traversal	0x936	left null
traversal	0x936	right null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



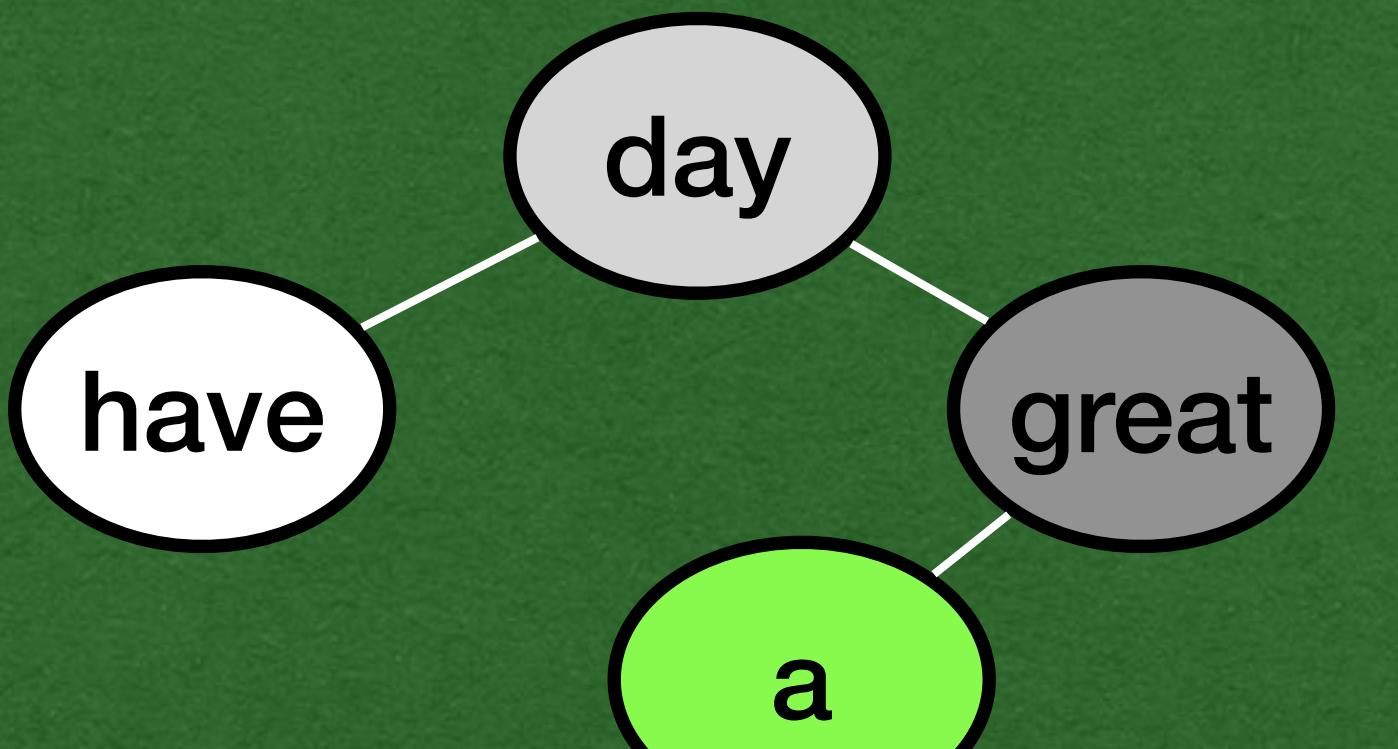
in/out
have

- Make a recursive call to the right of the root

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	traversal
node	0x200	traversal
node	null	traversal
node	null	traversal
node	0x480	traversal
	0x350	BTNode
	0x200	value "have"
	0x480	left null
	0x936	right null
	0x936	BTNode
	0x200	value "great"
	0x936	left null
	0x480	right null
	0x936	BTNode
	0x200	value "a"
	0x936	left null
	0x480	right null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



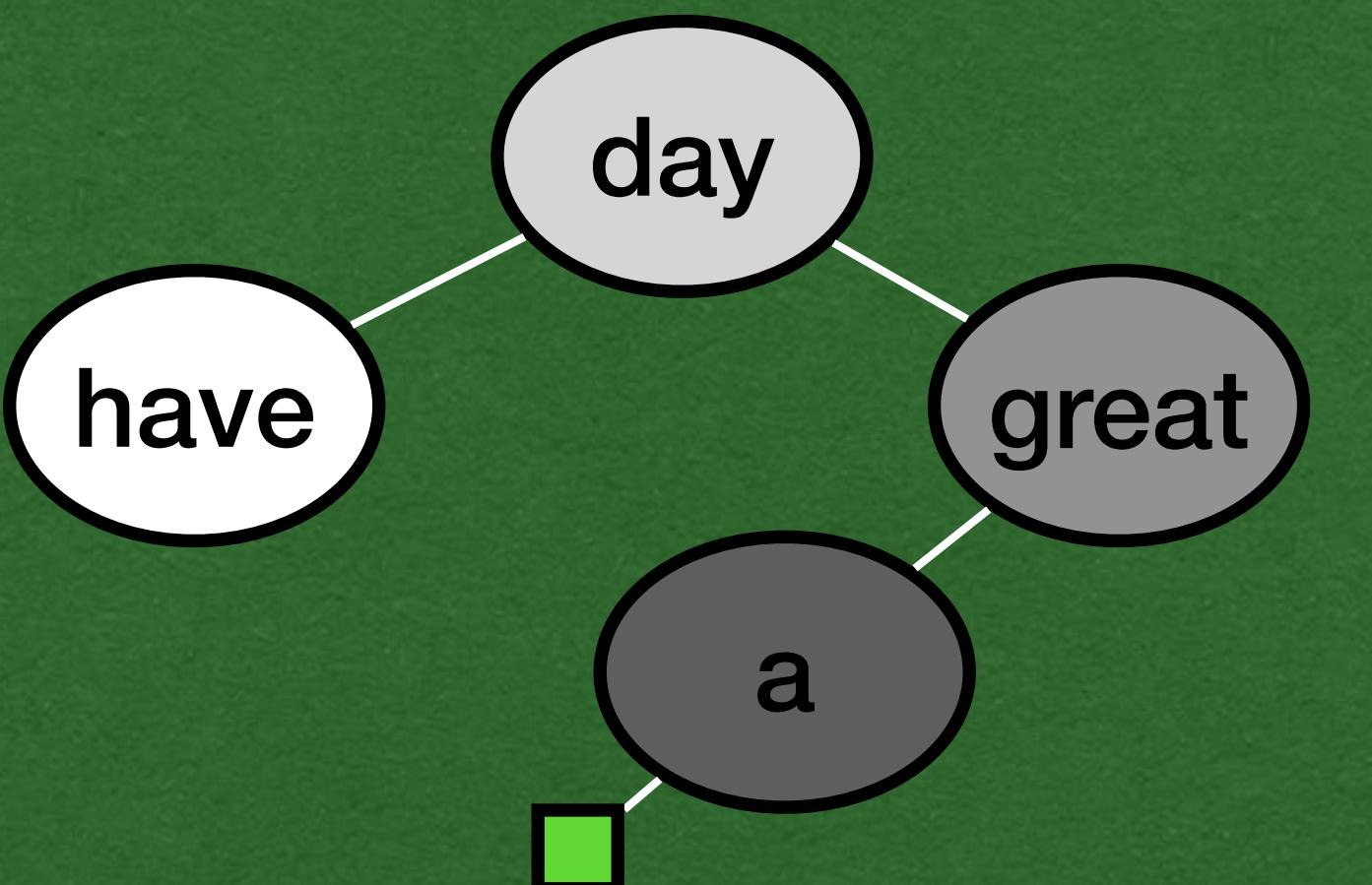
in/out
have

- That node makes a recursive call to its left

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	traversal
node	null	value "have"
node	null	left null
node	0x480	right null
node	0x936	0x480
traversal	0x936	BTNode
traversal		value "great"
traversal		left null
traversal		right null
traversal		0x936
traversal		BTNode
traversal		value "a"
traversal		left null
traversal		right null
traversal		0x936

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```

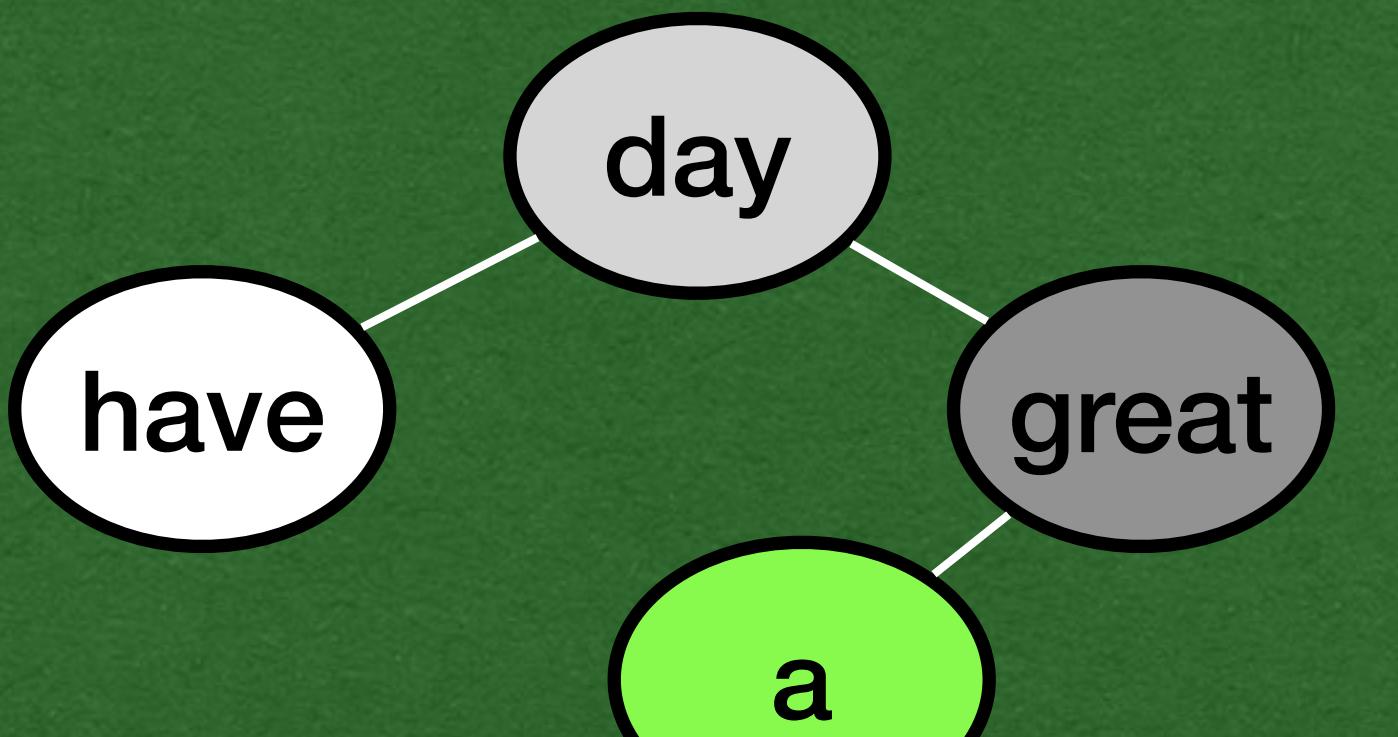


- Recursive call of null to the left
- Return

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "have" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "have" left null right null
this	0x200	BTNode value "great" left null right null
value	"have"	BTNode value "great" left null right null
left	null	BTNode value "great" left null right null
right	null	BTNode value "great" left null right null
this	0x480	BTNode value "a" left null right null
value	"great"	BTNode value "a" left null right null
left	null	BTNode value "a" left null right null
right	null	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
node	0x350	BTNode value "a" left null right null
traversal	0x200	BTNode value "a" left null right null
traversal	null	BTNode value "a" left null right null
traversal	null	BTNode value "a" left null right null
traversal	0x480	BTNode value "a" left null right null
traversal	0x936	BTNode value "a" left null right null
traversal	null	BTNode value "a" left null right null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```

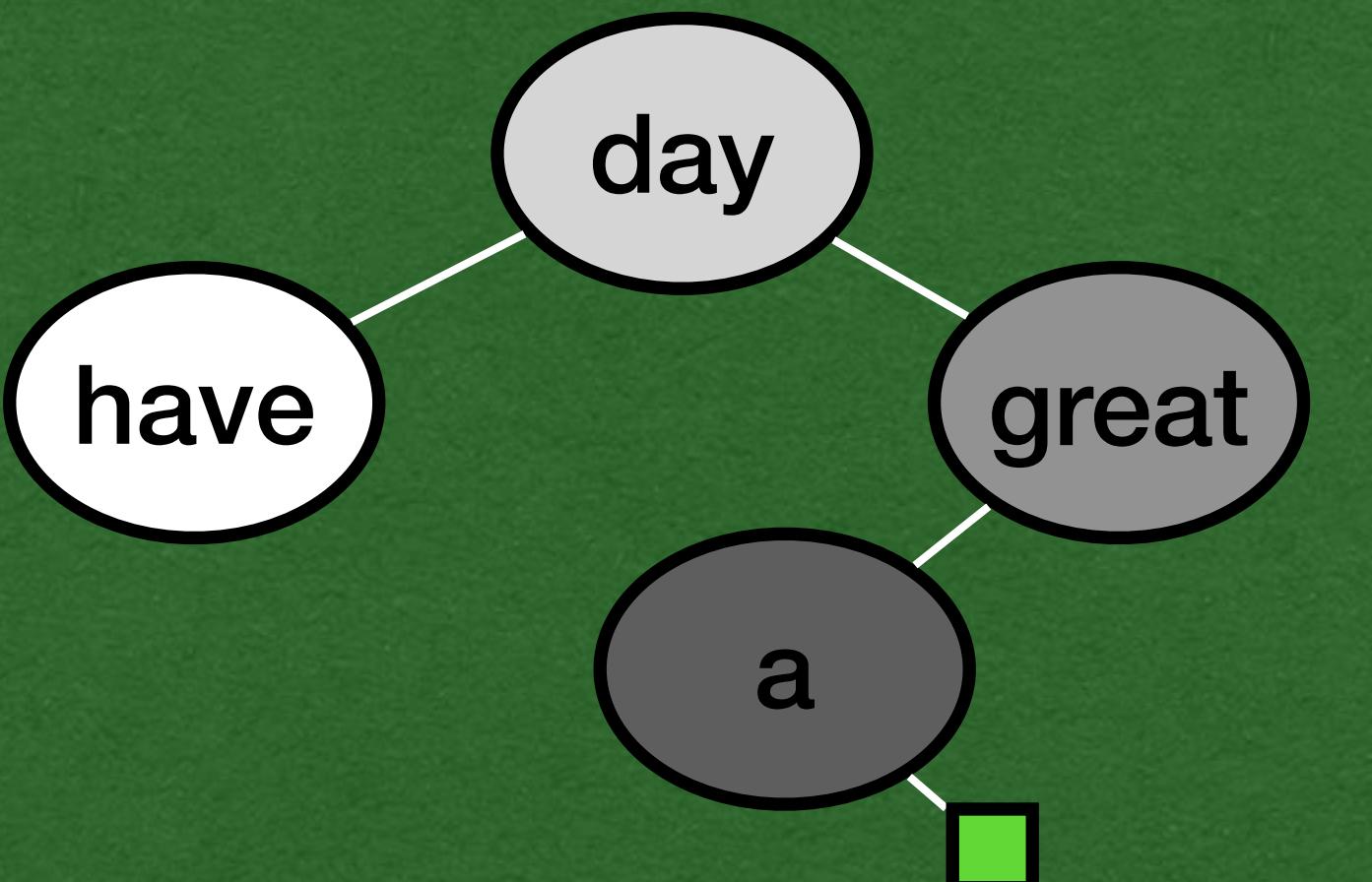


- Return to the previous frame and make the right recursive call

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "have"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	BTNode
node	null	value "have"
node	null	left null
node	null	right null
node	0x480	BTNode
node	0x936	value "great"
node	null	left null
node	null	right null
traversal	0x350	0x480
traversal	0x200	BTNode
traversal	null	value "have"
traversal	null	left null
traversal	null	right null
traversal	0x480	BTNode
traversal	0x936	value "great"
traversal	null	left null
traversal	null	right null
traversal	0x936	0x936

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



- Make the right recursive call of null

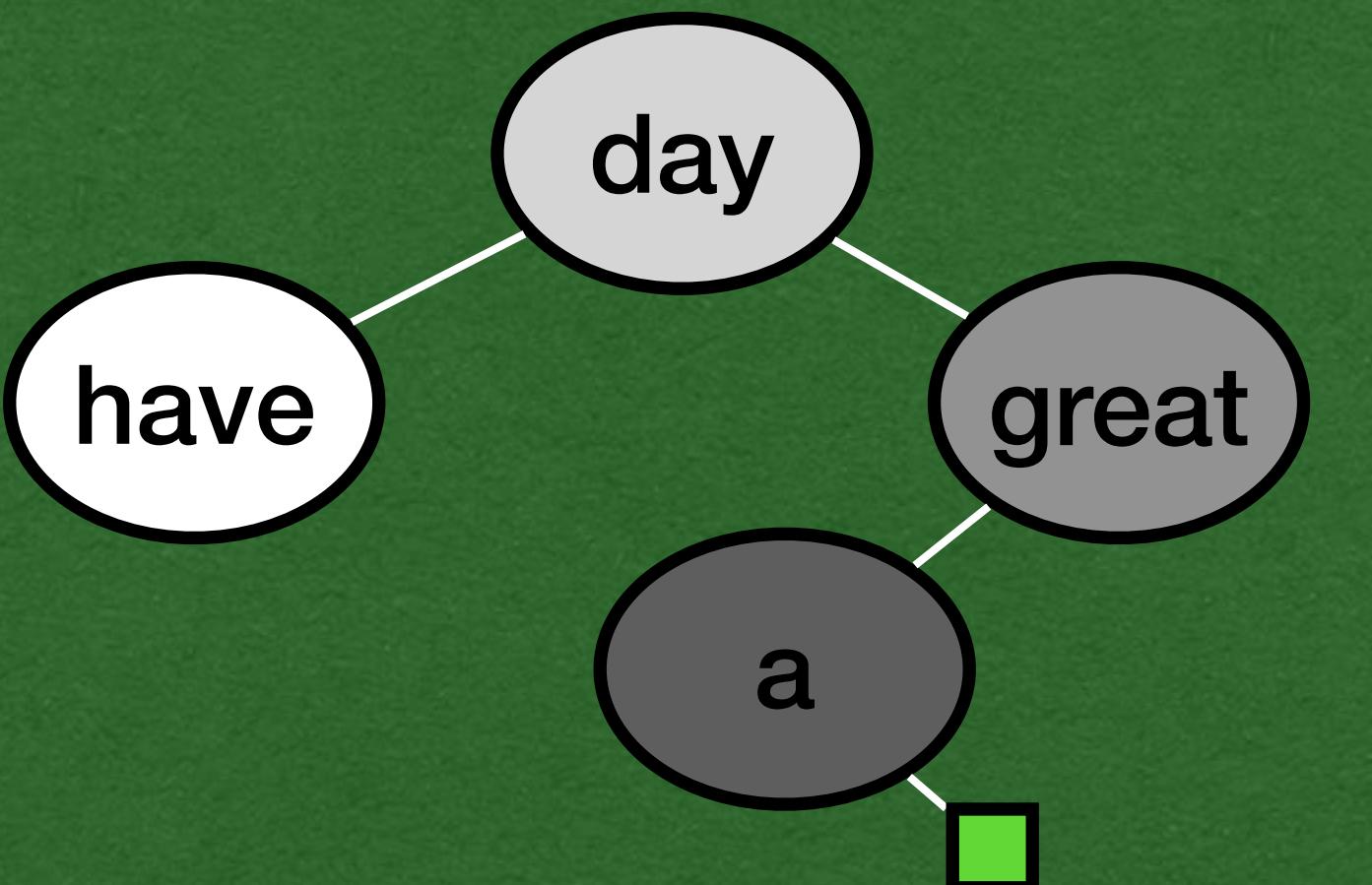
in/out
have

Stack	
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

Heap	
BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"have"
left	null
right	null
0x480	
BTNode	
value	"great"
left	null 0x936
right	null
0x936	
BTNode	
value	"a"
left	null
right	null
0x936	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



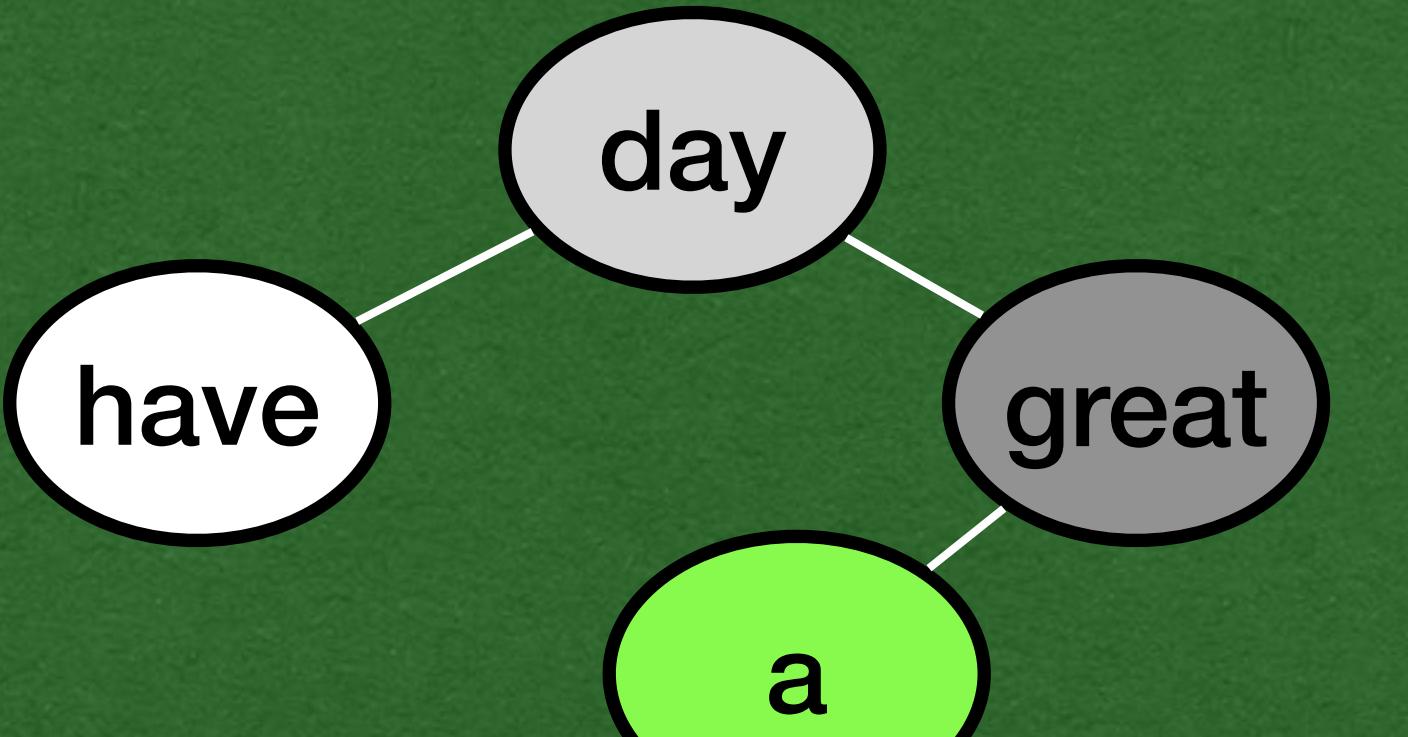
in/out
have

- Notice that all 4 stack frames remember what they need to do next when they regain control

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	traversal
node	null	value "have"
node	null	left null
node	0x480	right null
node	0x936	traversal
node	null	value "great"
node	null	left null
node	0x936	traversal
node	null	value "a"
node	null	left null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



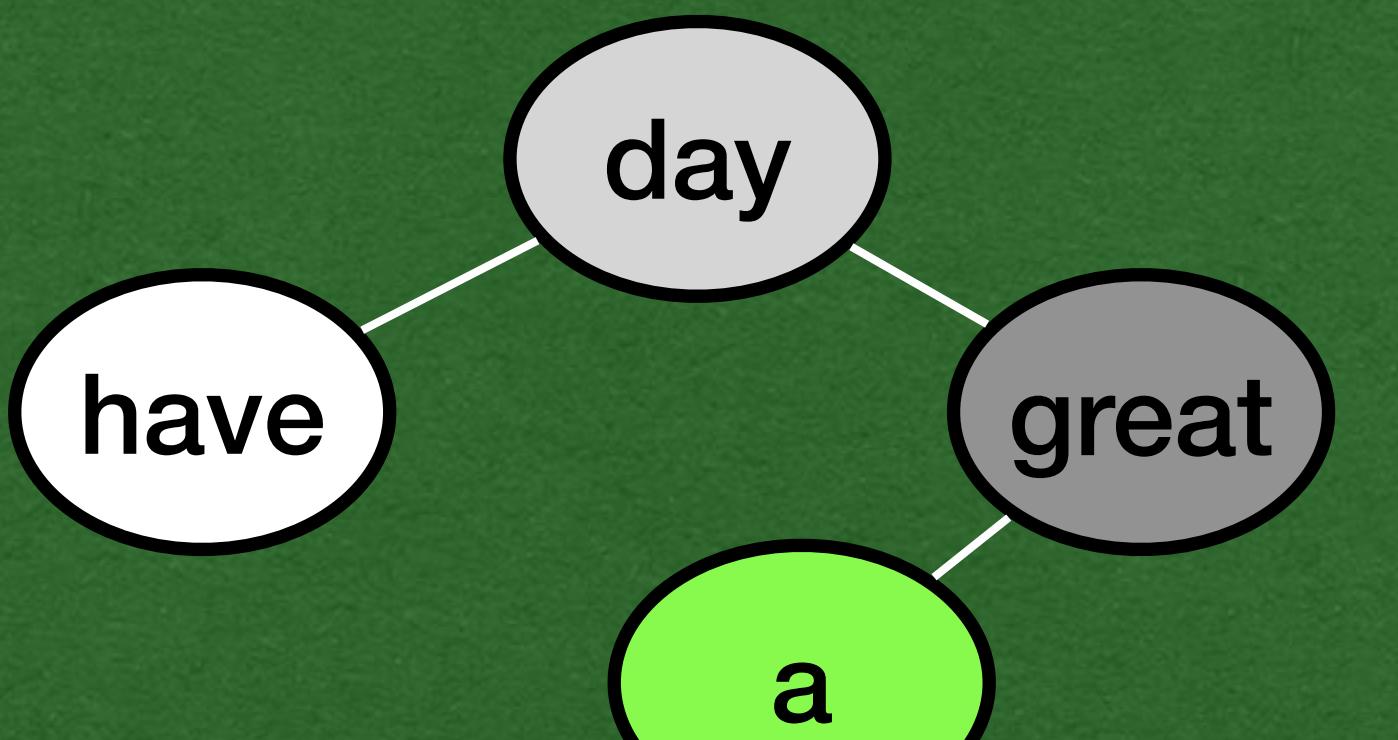
in/out
have

- This stack frame already made both recursive calls

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	0x480
node	null	
node	null	
node	0x480	BTNode
node	0x936	value "great"
left	0x936	left null
right	null	right null
traversal	0x350	BTNode
traversal	0x200	value "have"
traversal	null	left null
traversal	null	right null
traversal	0x480	BTNode
traversal	0x936	value "great"
traversal	null	left null
traversal	null	right null
traversal	0x936	BTNode
traversal	0x936	value "a"
traversal	null	left null
traversal	null	right null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



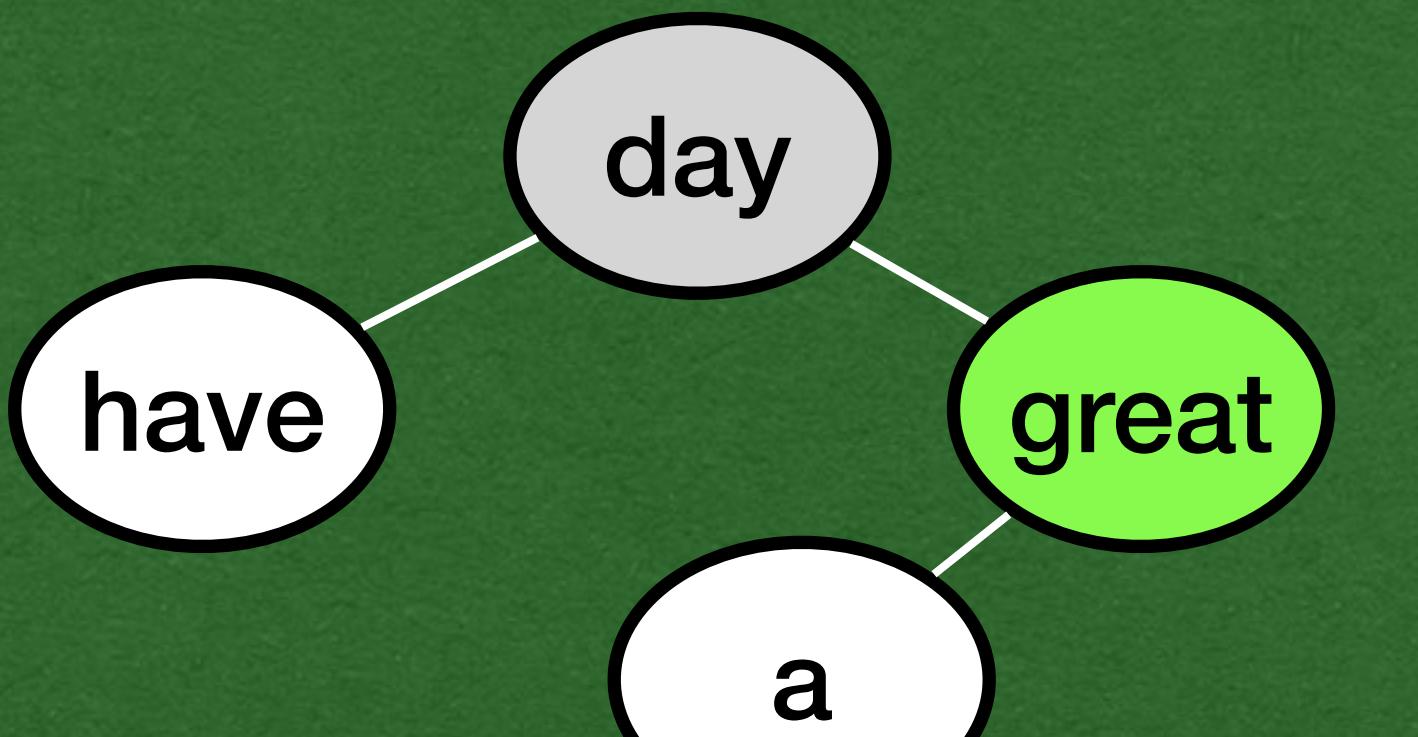
- Print "a" to the screen

in/out
have a

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	traversal
node	null	node 0x350
node	null	node 0x200
node	0x480	traversal
node	0x936	node 0x480
node	null	traversal
node	null	node 0x936
traversal		traversal
traversal		value "a"
traversal		left null
traversal		right null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



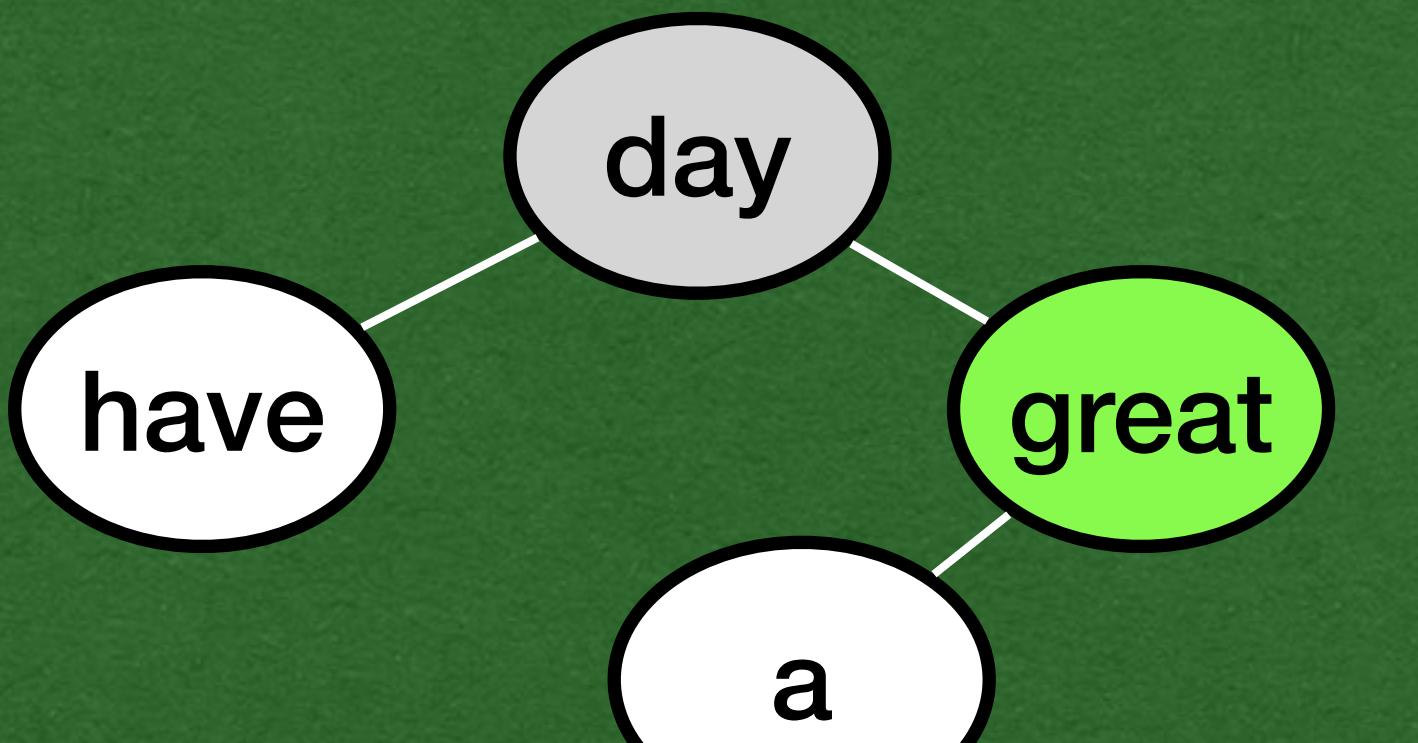
in/out
have a

- Return back to the previous stack frame

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	0x480
node	null	
node	null	
node	0x480	BTNode
node	0x936	value "great"
node	null	left null
node	null	right null
traversal	0x350	0x480
traversal	0x200	
traversal	null	
traversal	null	
traversal	0x480	BTNode
traversal	0x936	value "a"
traversal	null	left null
traversal	null	right null
traversal	0x936	0x936

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```

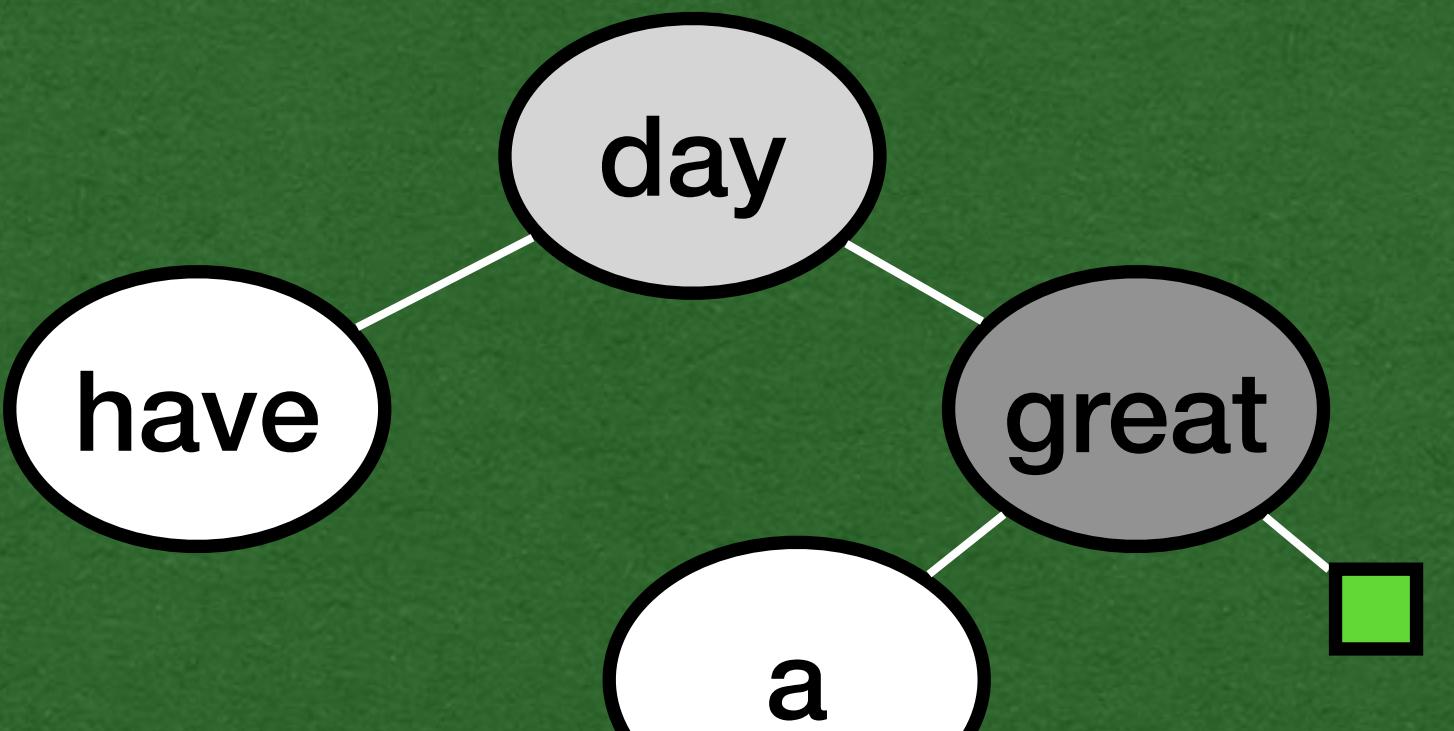


- Make the right recursive call

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "great" left null right null
left	null	BTNode value "a" left null right null
right	null	BTNode value "have" left null right null
this	0x200	traversal node 0x350
value	"have"	traversal node 0x200
left	null	traversal node null
right	null	traversal node null
this	0x480	traversal node 0x480
value	"great"	traversal node 0x936
left	null	traversal node null
right	null	traversal node null
this	0x936	traversal node 0x936
value	"a"	traversal node null
left	null	traversal node null
right	null	traversal node null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



- Base case of null
- Just return

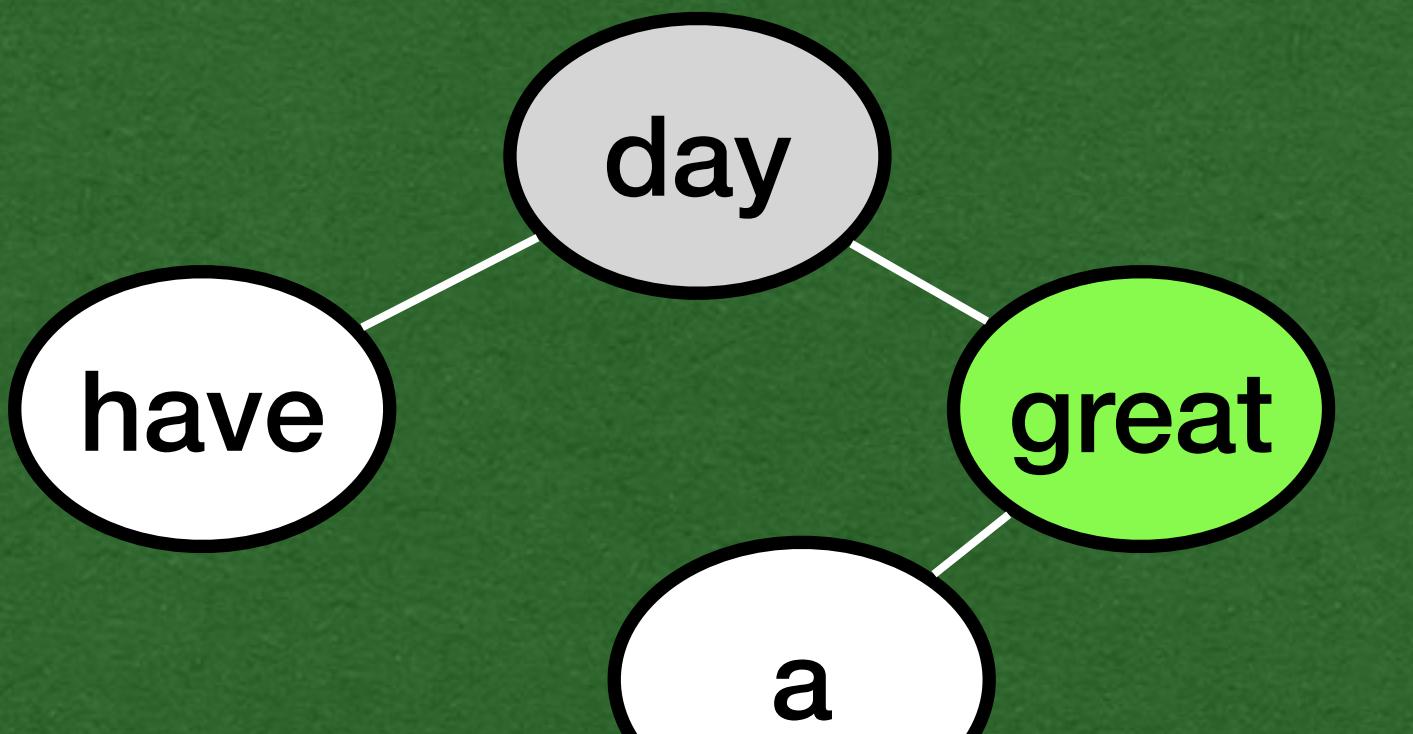
in/out
have a

Stack	
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null
node	0x936

Heap	
BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"great"
left	null
right	null
0x480	
BTNode	
value	"a"
left	null
right	null
0x936	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```

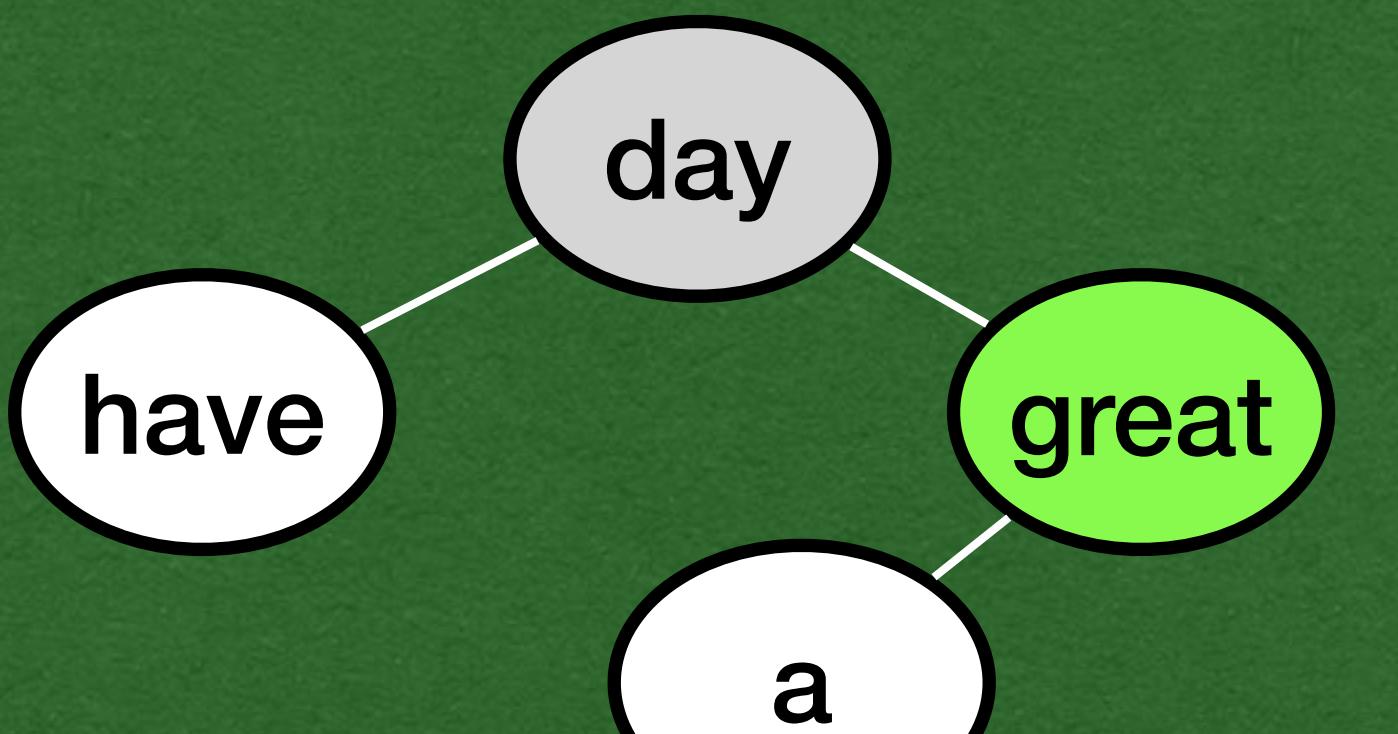


- Stack frame done with both recursive calls

Stack		Heap
Name	Value	
root	0x350	BTNode value "day" left null right null
this	0x350	BTNode value "have" left null right null
value	"day"	BTNode value "great" left null right null
left	null	BTNode value "have" left null right null
right	null	BTNode value "great" left null right null
this	0x200	traversal node 0x350
value	"have"	traversal node 0x200
left	null	traversal node null
right	null	traversal node null
this	0x480	traversal node 0x480
value	"great"	traversal node 0x936
left	null	traversal node null
right	null	traversal node null
this	0x936	traversal node 0x936
value	"a"	traversal node null
left	null	traversal node null
right	null	traversal node null

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



in/out
have a great

- Print "great" to the screen

Stack

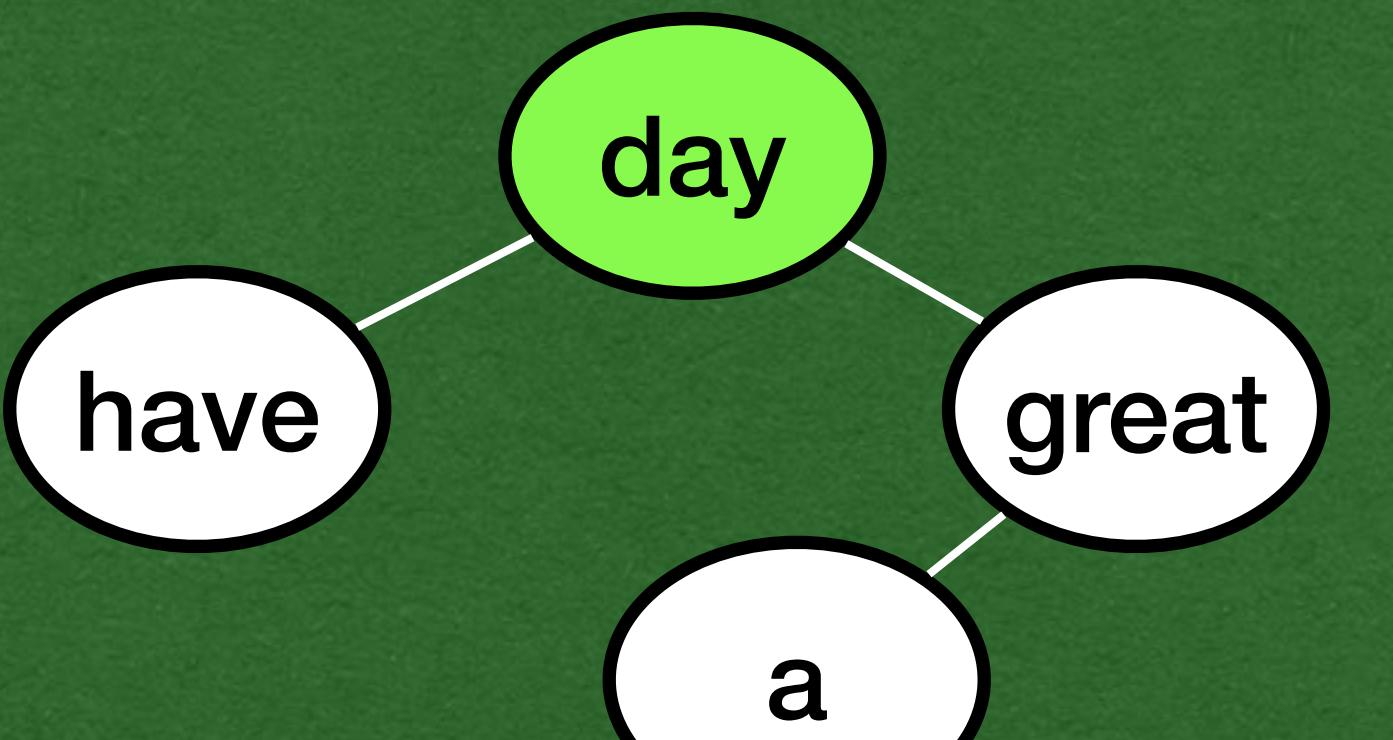
Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null
node	0x936

Heap

BTNode	
value	"day"
left	null 0x200
right	null 0x480
0x350	
BTNode	
value	"have"
left	null
right	null
0x200	
BTNode	
value	"great"
left	null
right	null
0x480	
BTNode	
value	"a"
left	null
right	null
0x936	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



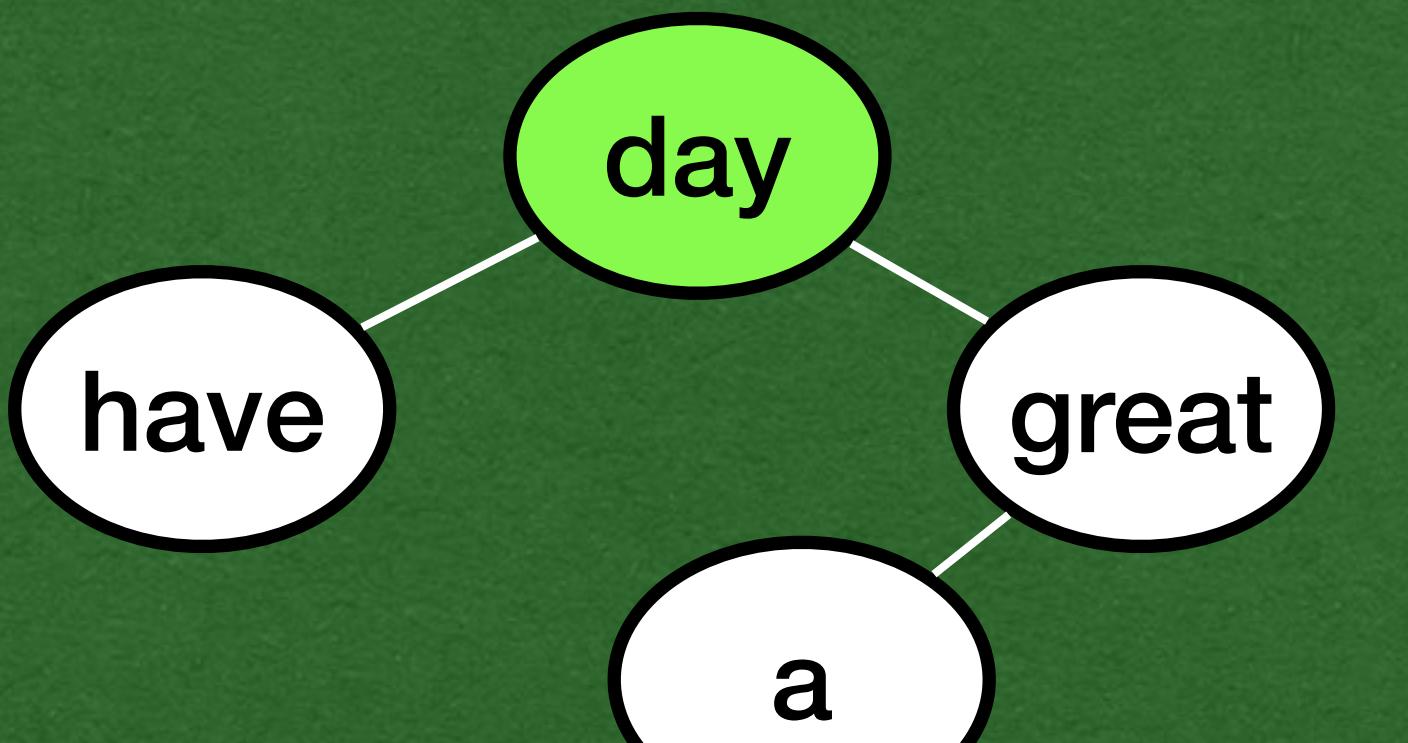
in/out
have a great

- Root node is now done with both recursive calls

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	BTNode
node	null	value "have"
node	null	left null
node	0x480	right null
node	0x480	
node	0x936	BTNode
node	0x936	value "great"
node	null	left null
node	null	right null
traversal	0x480	0x936
traversal	0x936	BTNode
traversal	null	value "a"
traversal	null	left null
traversal	null	right null
traversal	0x936	
traversal	0x936	BTNode
traversal	null	value "great"
traversal	null	left null
traversal	null	right null
traversal	0x936	0x936

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



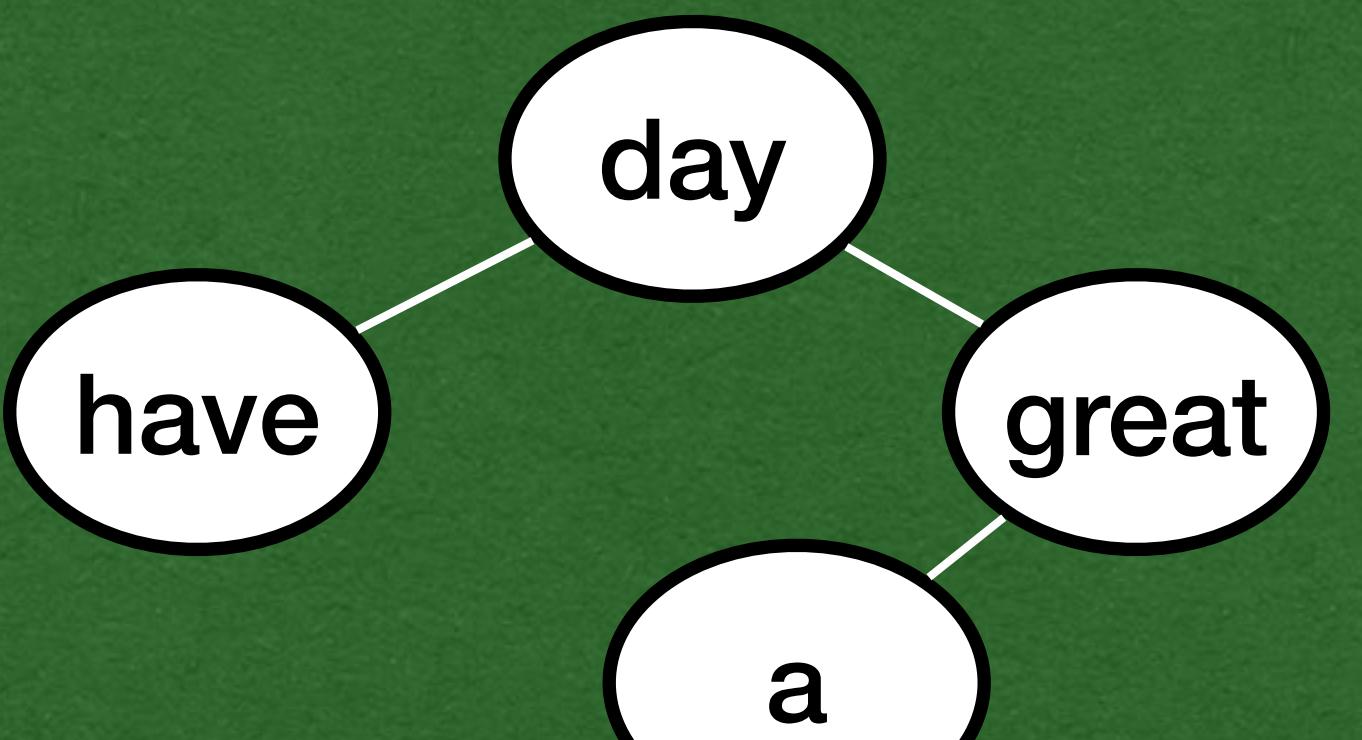
in/out
have a great day

- Print "day" to the screen

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left null
right	null	right null
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "a"
left	null	left null
right	null	right null
node	0x350	0x200
node	0x200	BTNode
node	null	value "have"
node	null	left null
node	0x480	right null
node	0x480	
node	0x936	BTNode
node	0x936	value "great"
node	null	left null
node	null	right null
traversal	0x480	0x936
traversal	0x936	BTNode
traversal	null	value "a"
traversal	null	left null
traversal	0x936	right null
traversal	0x936	
traversal	null	BTNode
traversal	null	value "day"
traversal	0x350	left null
traversal	0x350	right null
traversal	null	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
}  
  
public static void main(String[] args) {  
    BTNode<String> root = new BTNode<>("day", null, null);  
    root.left = new BTNode<>("have", null, null);  
    root.right = new BTNode<>("great", null, null);  
    root.right.left = new BTNode<>("a", null, null);  
    root.traversal(root);  
}
```



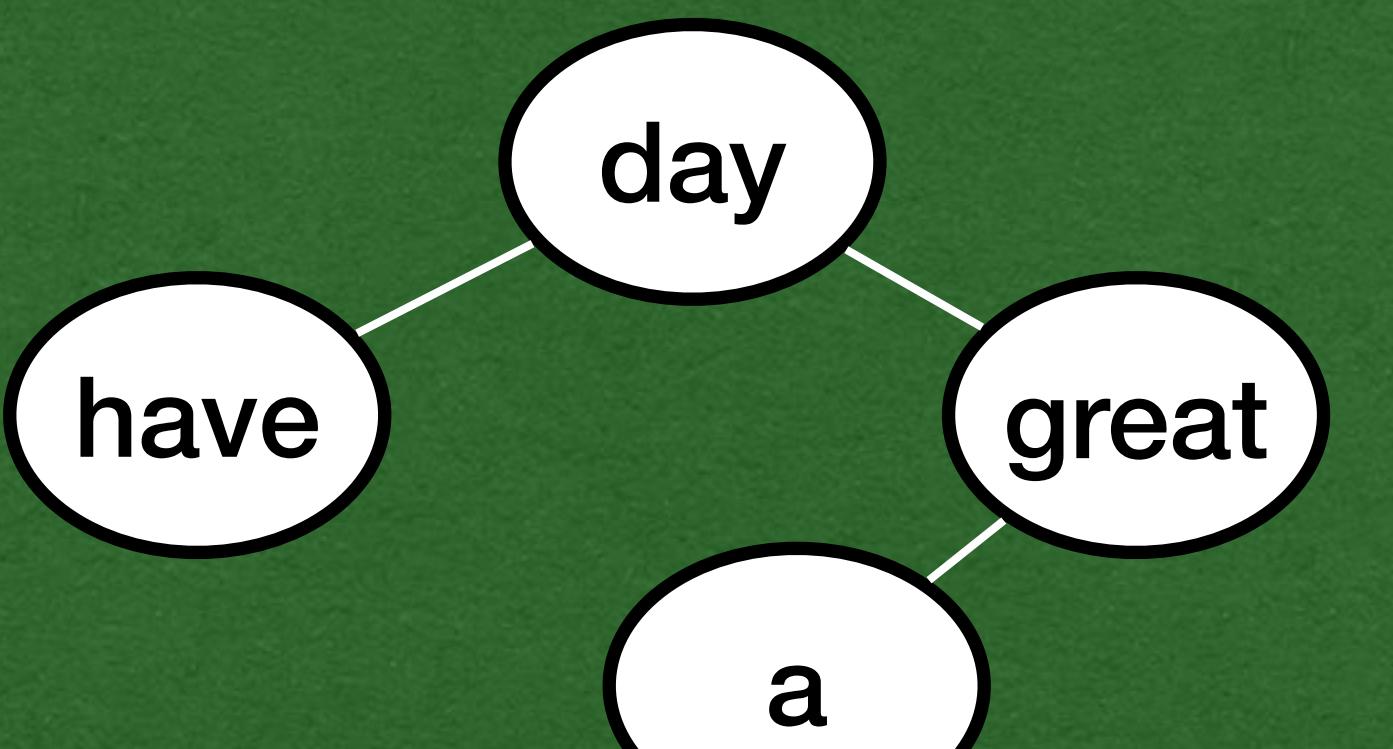
in/out
have a great day

- The traversal is over
- All stack frames have returned

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "day"
value	"day"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"have"	value "have"
left	null	left 0x200
right	null	right 0x480
this	0x480	BTNode
value	"great"	value "great"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"a"	value "have"
left	null	left null
right	null	right null
traversal		0x200
node	0x350	BTNode
traversal	0x200	value "have"
node	null	left null
traversal	null	right null
node	0x480	BTNode
traversal	0x480	value "great"
node	0x936	left 0x936
traversal	null	right null
node	null	0x480
traversal		BTNode
node	0x936	value "a"
traversal		left null
node	null	right null
traversal		0x936
node	null	

```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
public static void traversal(BTNode<A> node) {  
    if (node != null) {  
        traversal(node.left);  
        traversal(node.right);  
        System.out.print(node.value + " ");  
    }  
  
    public static void main(String[] args) {  
        BTNode<String> root = new BTNode<>("day", null, null);  
        root.left = new BTNode<>("have", null, null);  
        root.right = new BTNode<>("great", null, null);  
        root.right.left = new BTNode<>("a", null, null);  
        root.traversal(root);  
    }  
}
```



in/out
have a great day

• Program ends

Stack

Name	Value
root	0x350
this	0x350
value	"day"
left	null
right	null
this	0x200
value	"have"
left	null
right	null
this	0x480
value	"great"
left	null
right	null
this	0x936
value	"a"
left	null
right	null
traversal	
node	0x350
traversal	
node	0x200
traversal	
node	null
traversal	
node	0x480
traversal	
node	0x936
traversal	
node	null
traversal	
node	0x936
traversal	
node	null

Heap

BTNode	
value "day"	
left	null 0x200
right	null 0x480
0x350	
BTNode	
value "have"	
left	null
right	null
0x200	
BTNode	
value "great"	
left	null
right	null
0x480	
BTNode	
value "a"	
left	null
right	null
0x936	