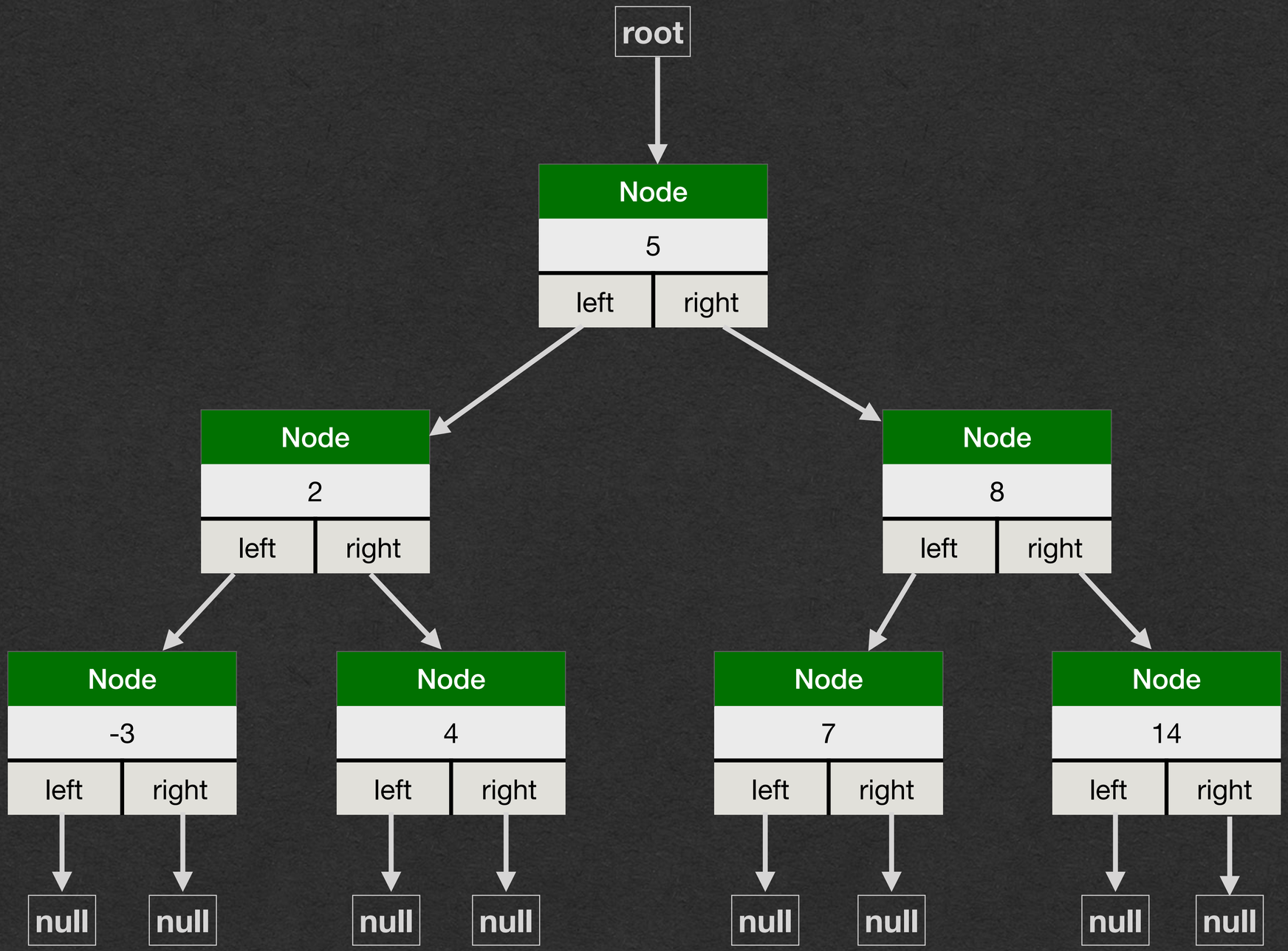
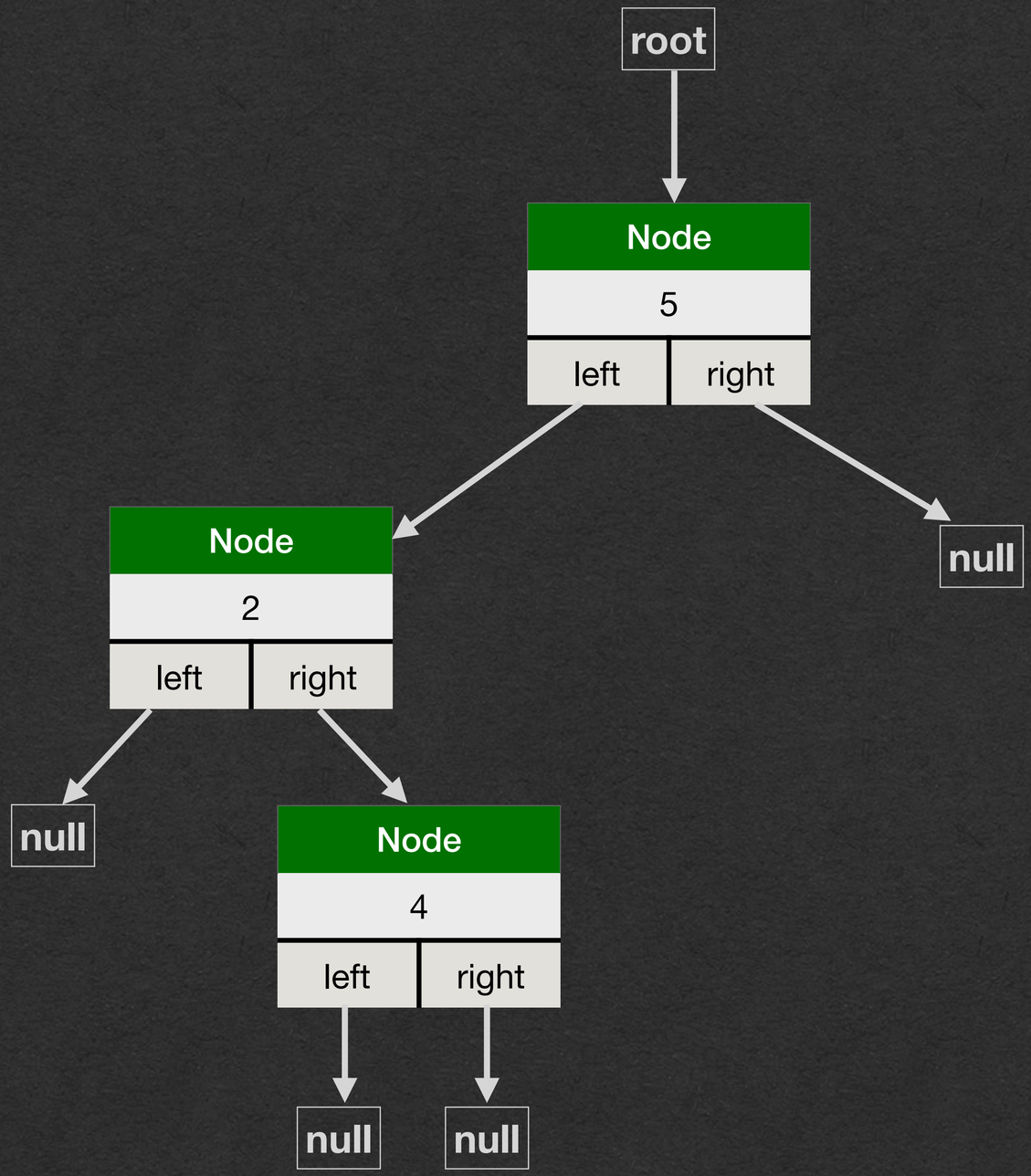
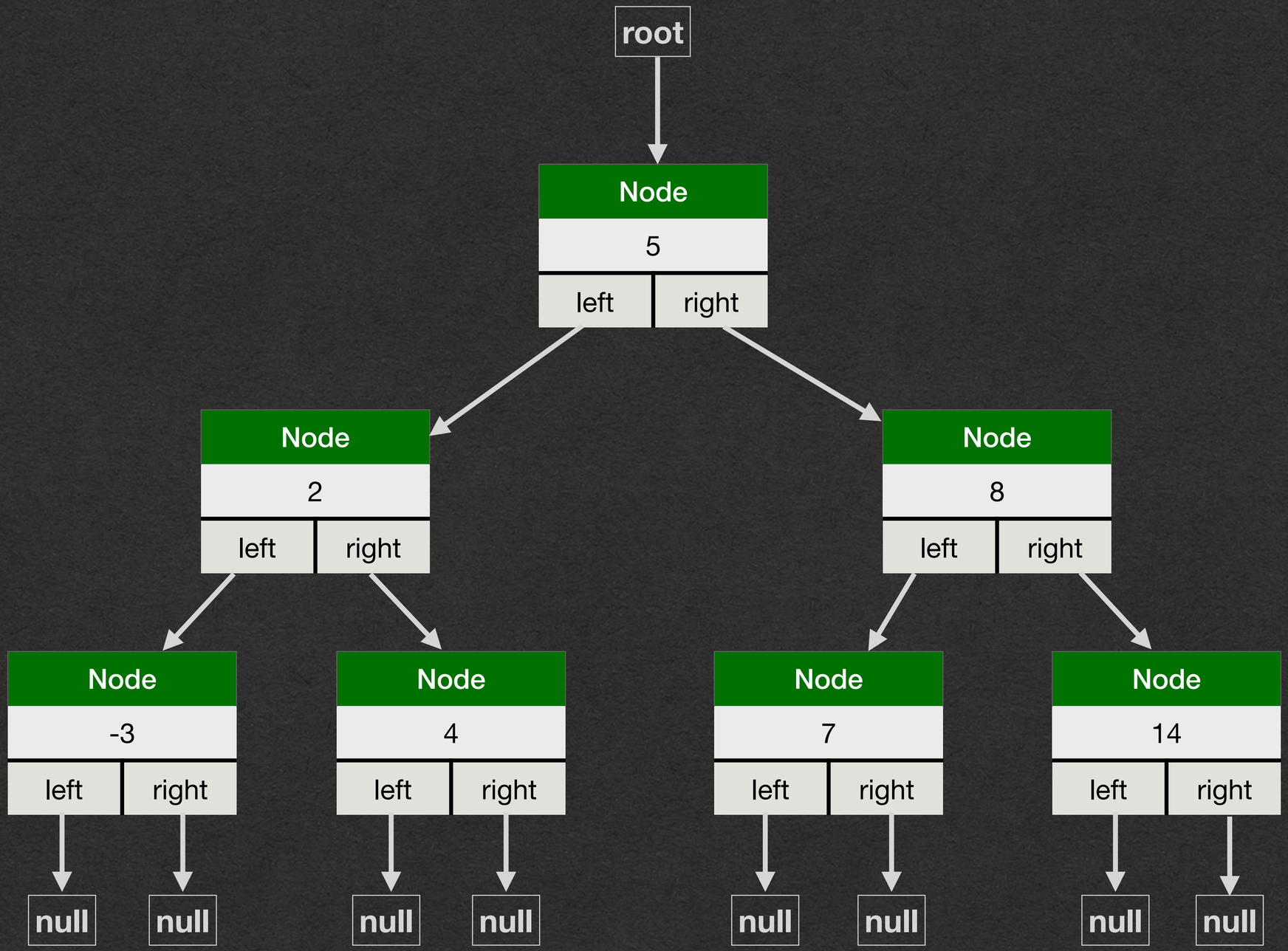
# Binary Trees and Traversals

# 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


root
Node
5
left
right
Node
2
left
right
Node
8
left
right
Node
-3
left
right
Node
4
left
right
Node
7
left
right
Node
14
left
right
null
null
null
null
null
null
null
null


root
Node
5
left
right
Node
2
left
right
null
null
Node
4
left
right
null
null


root
Node
5
left
right
Node
2
left
right
Node
8
left
right
Node
-3
left
right
Node
4
left
right
Node
7
left
right
Node
14
left
right
null
null
null
null
null
null
null
null

```java
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;
    }
}
```

```java
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**

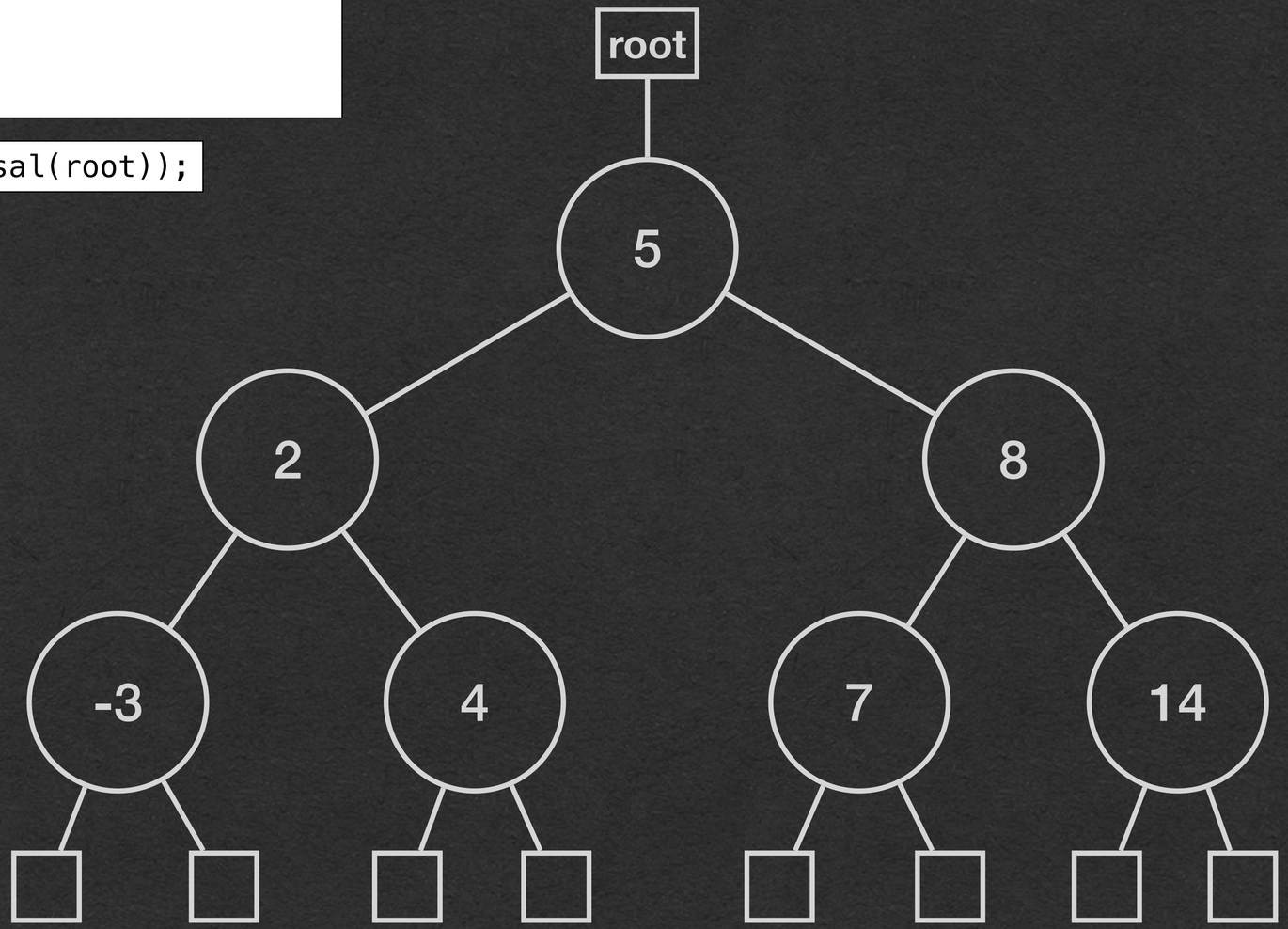
```
public String inOrderTraversal(BinaryTreeNode<A> node) {
    if (node != null) {
        String out = "";
        out += inOrderTraversal(node.left);
        out += node.value.toString() + " ";
        out += inOrderTraversal(node.right);
        return out;
    } else {
        return "";
    }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

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 "";
    }
}
```

# in order

```java
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 "";
    }
}
```

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

**In/Out**
**-3 2 4 5 7 8 14**



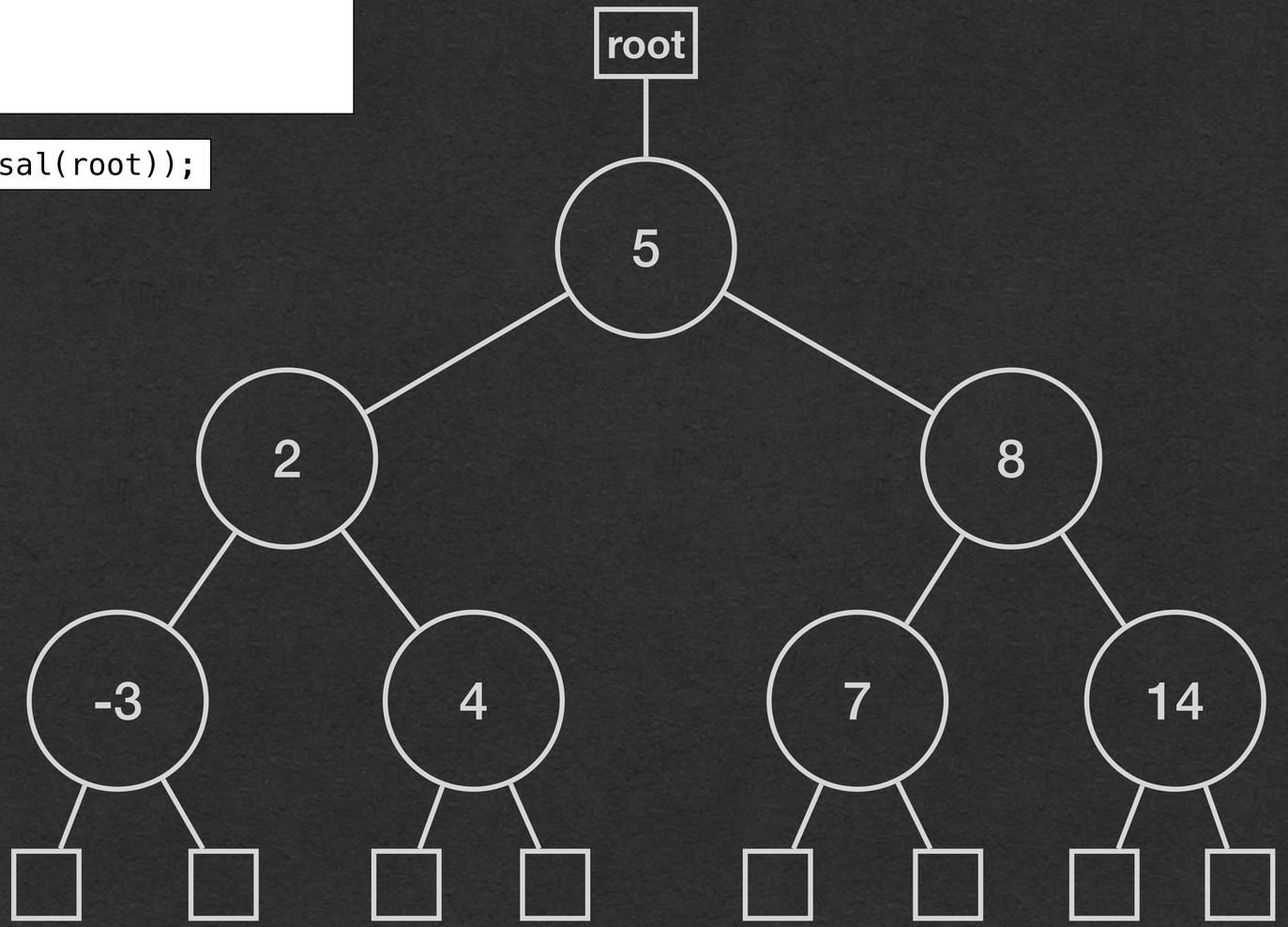
# pre order

```java
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 "";
    }
}
```

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

**In/Out**

**5 2 -3 4 8 7 14**

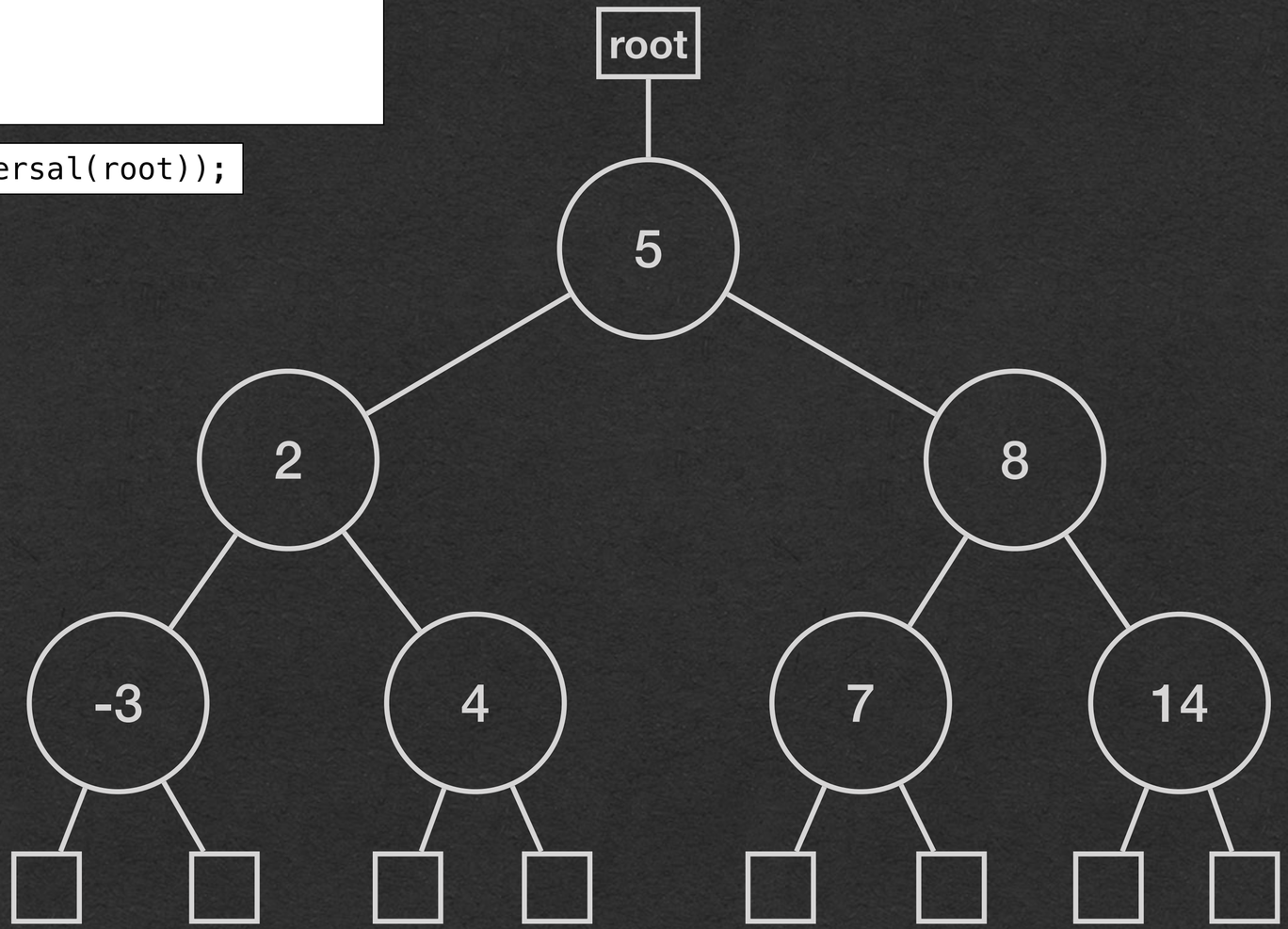


# post order

```java
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 "";
    }
}
```

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

**In/Out**
**-3 4 2 7 14 8 5**



# 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);
    traversal(root);
}
```

Stack

| Name | Value |
|---|---|

Heap

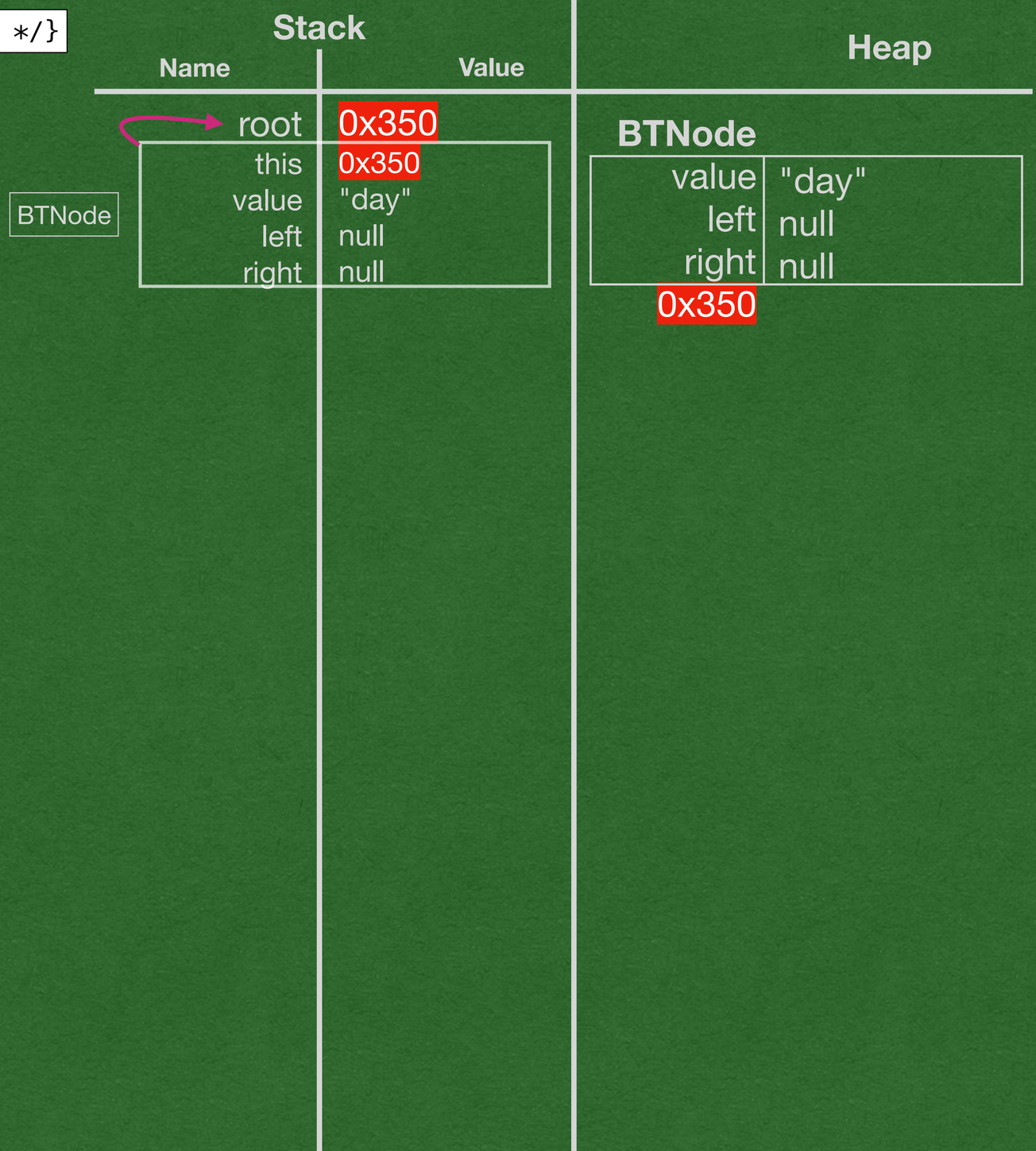
**in/out**

- Build a tree
- Visit every node with a post-order 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);
    traversal(root);
}
```

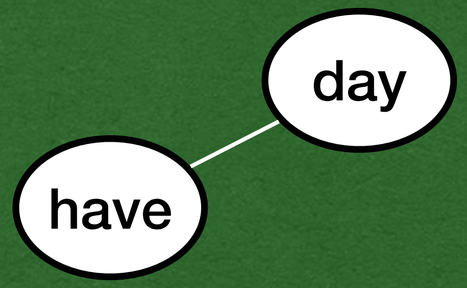

**in/out**



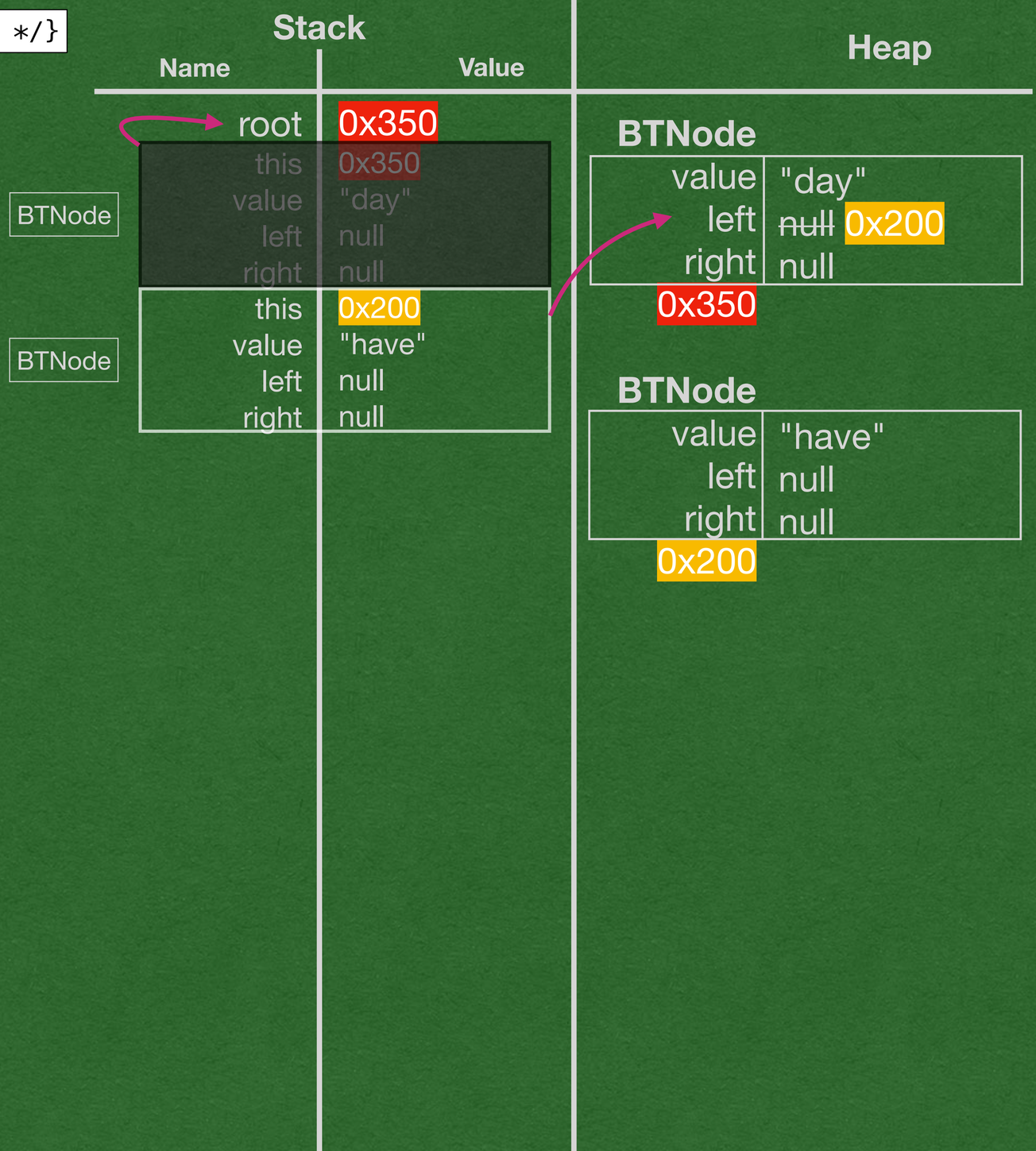

- Create the root node with the value "day"

```
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);
    traversal(root);
}
```



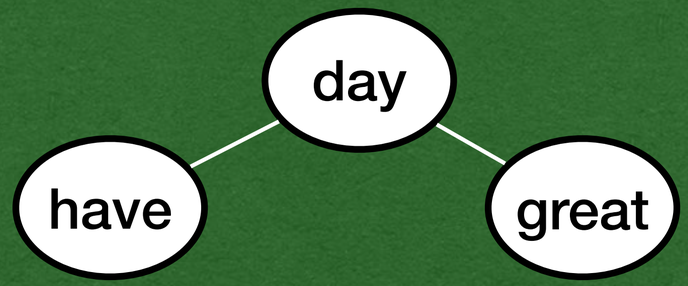

**in/out**



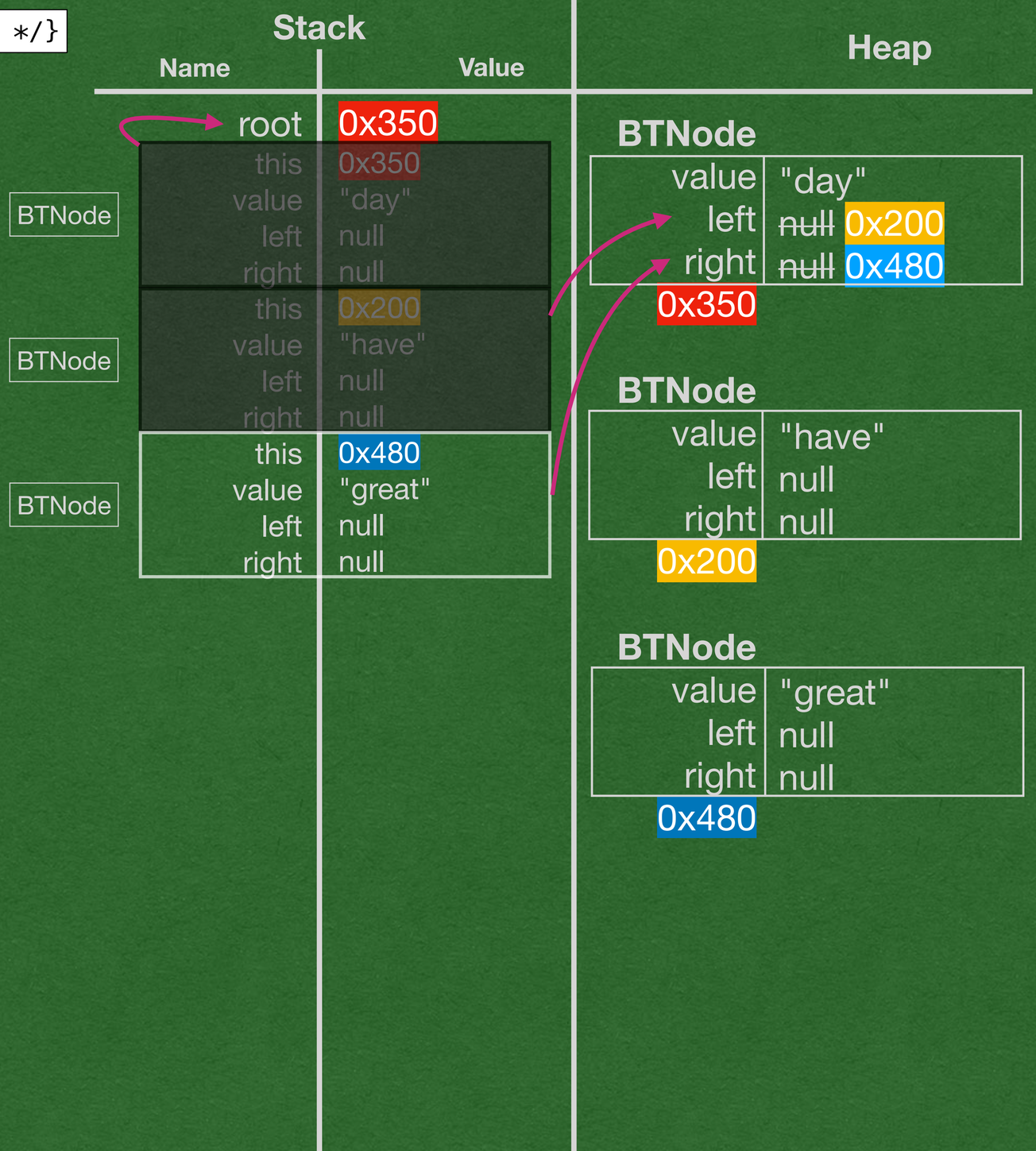

- Set the left child of the root to a node with "have"

```
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);
    traversal(root);
}
```



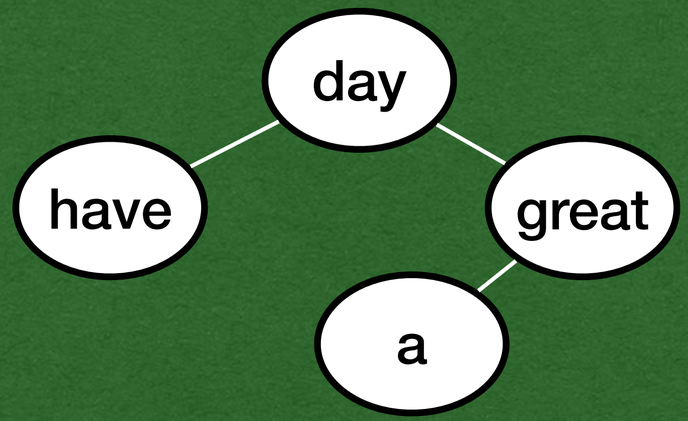

in/out




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

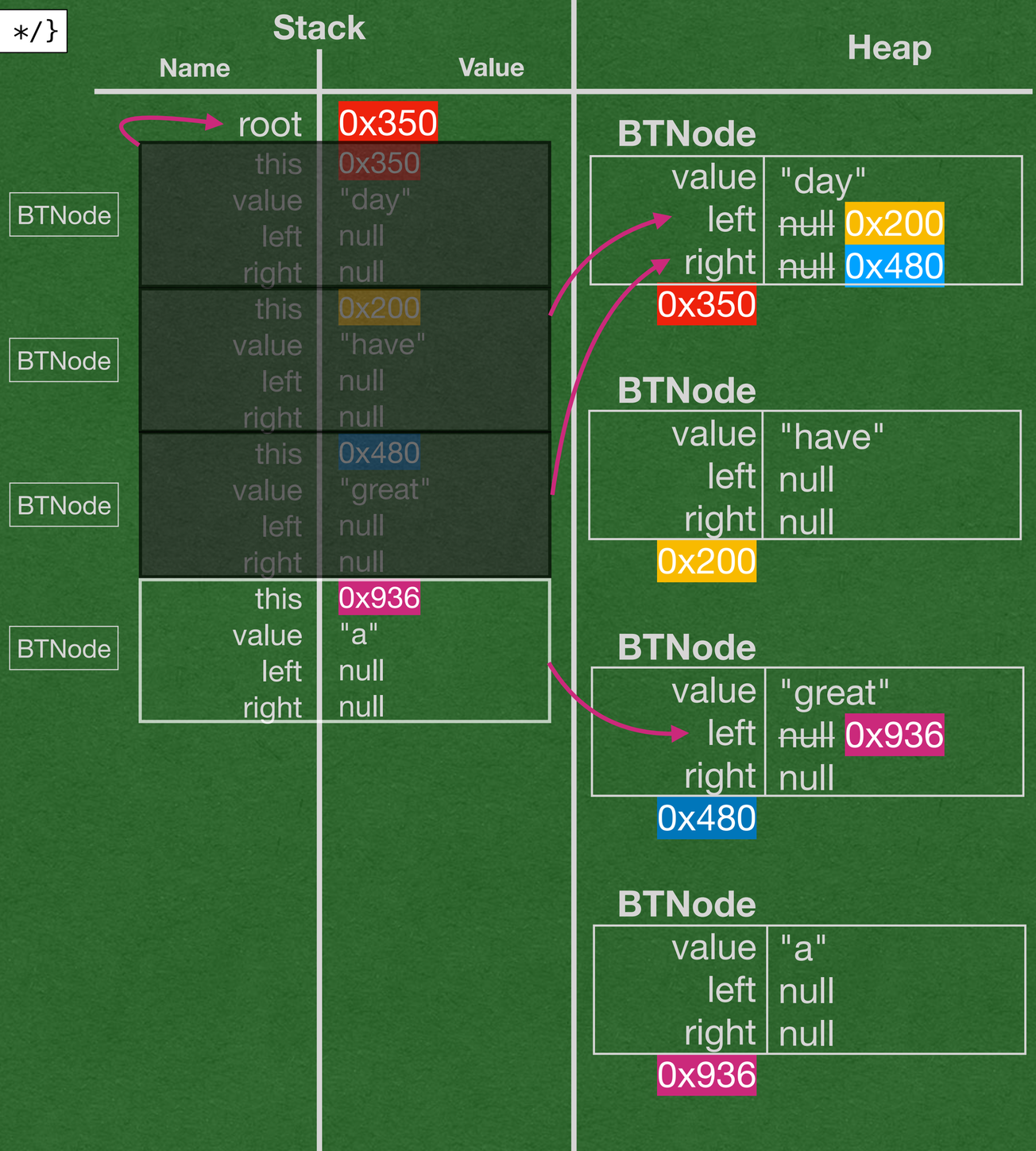
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```



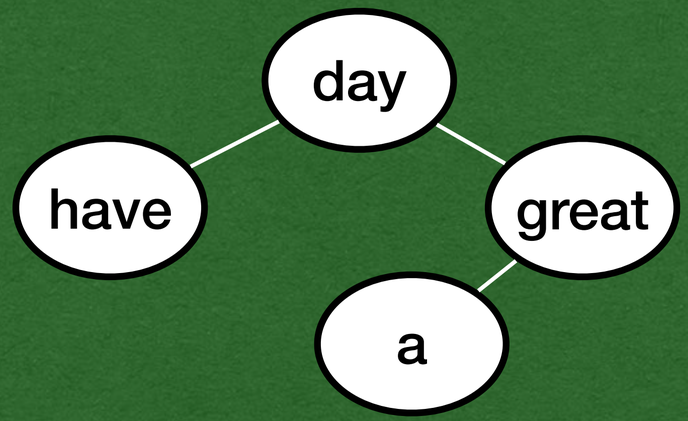

**in/out**

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



```
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);
    traversal(root);
}
```




**in/out**

- Time to start the post-order traversal

**Stack**

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

**Heap**

**BTNode** 0x350

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** 0x200

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** 0x480

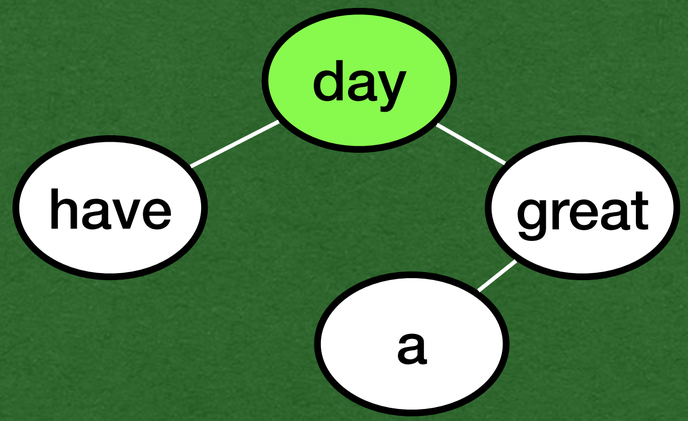
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** 0x936

| | |
|---|---|
| value | "a" |
| left | null |
| right | null |

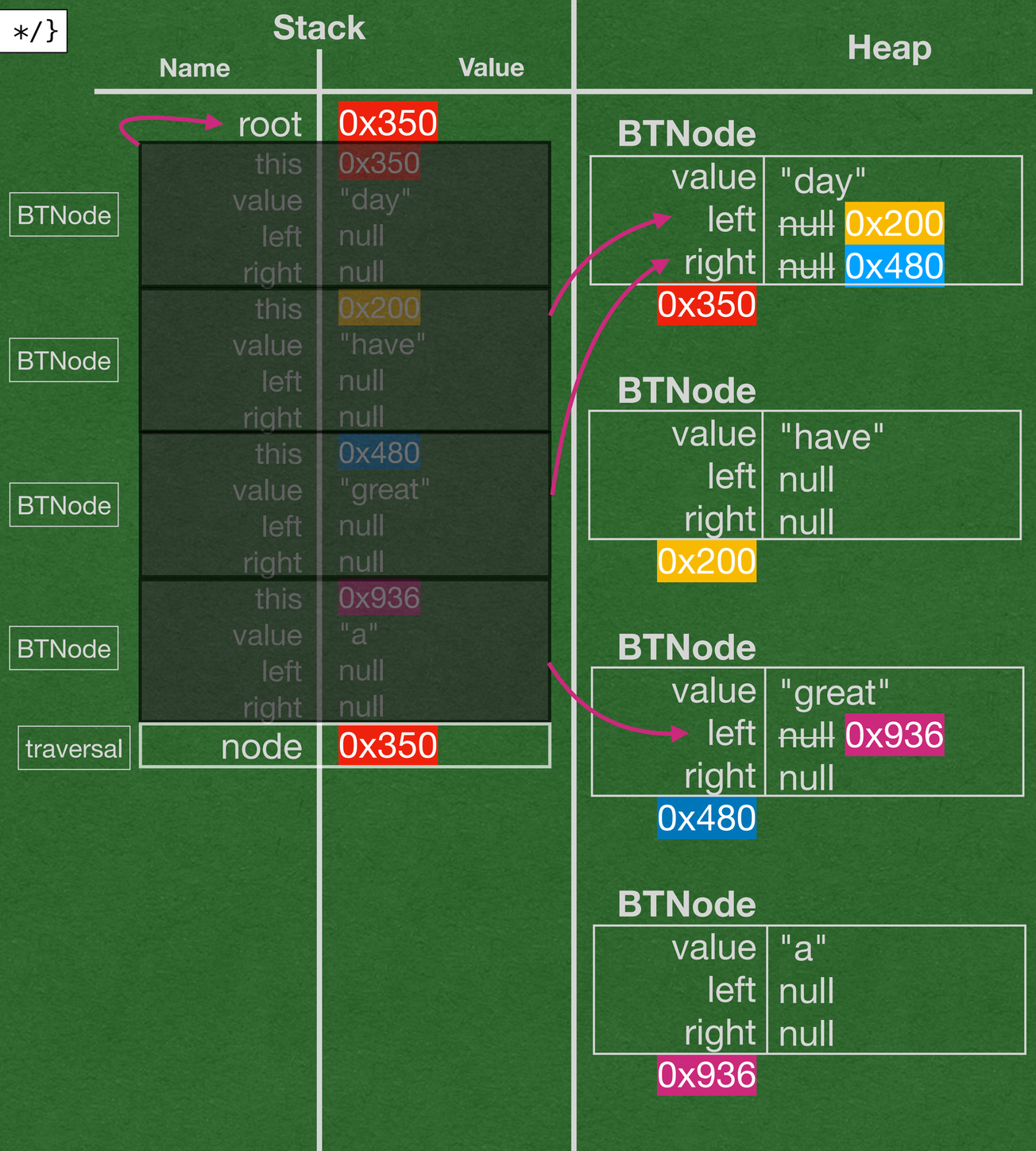
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```



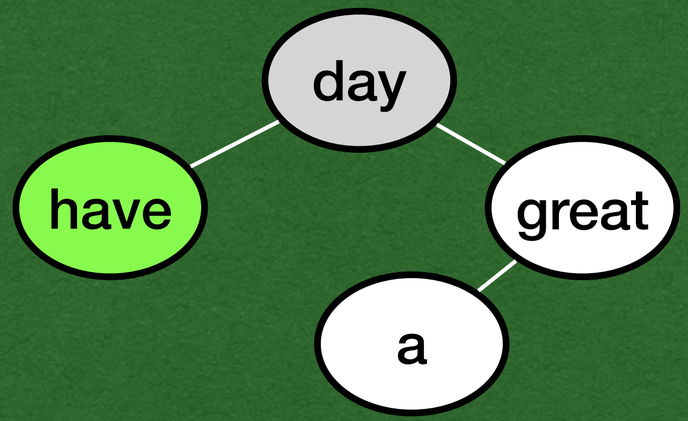

**in/out**

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



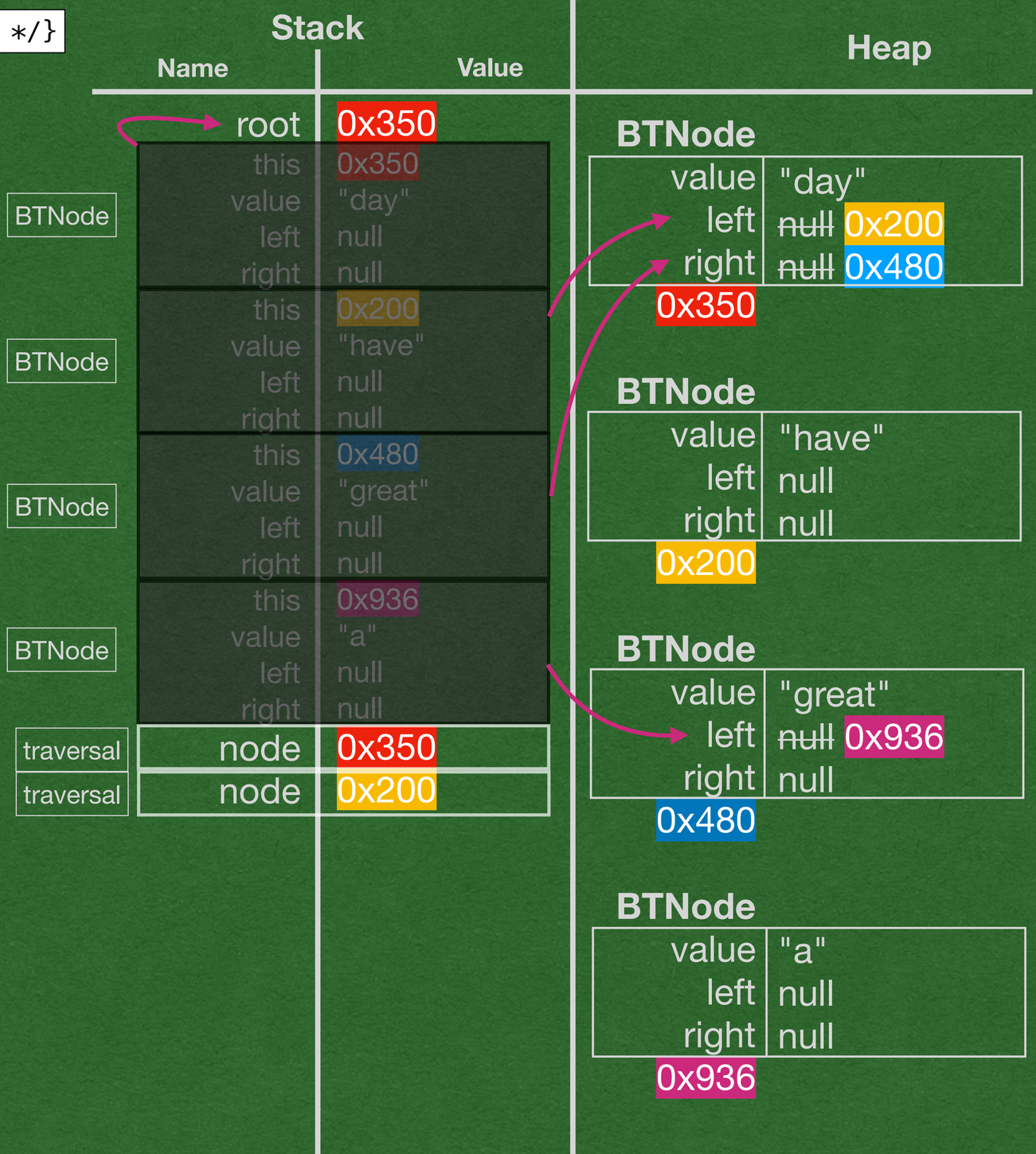
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```



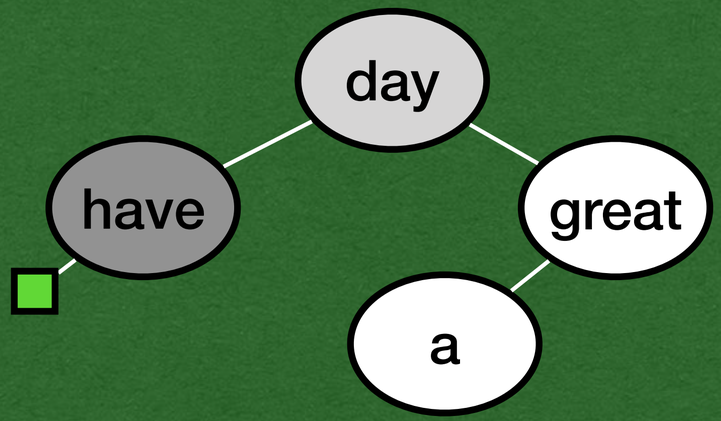

**in/out**

- Make a recursive call on the left child first



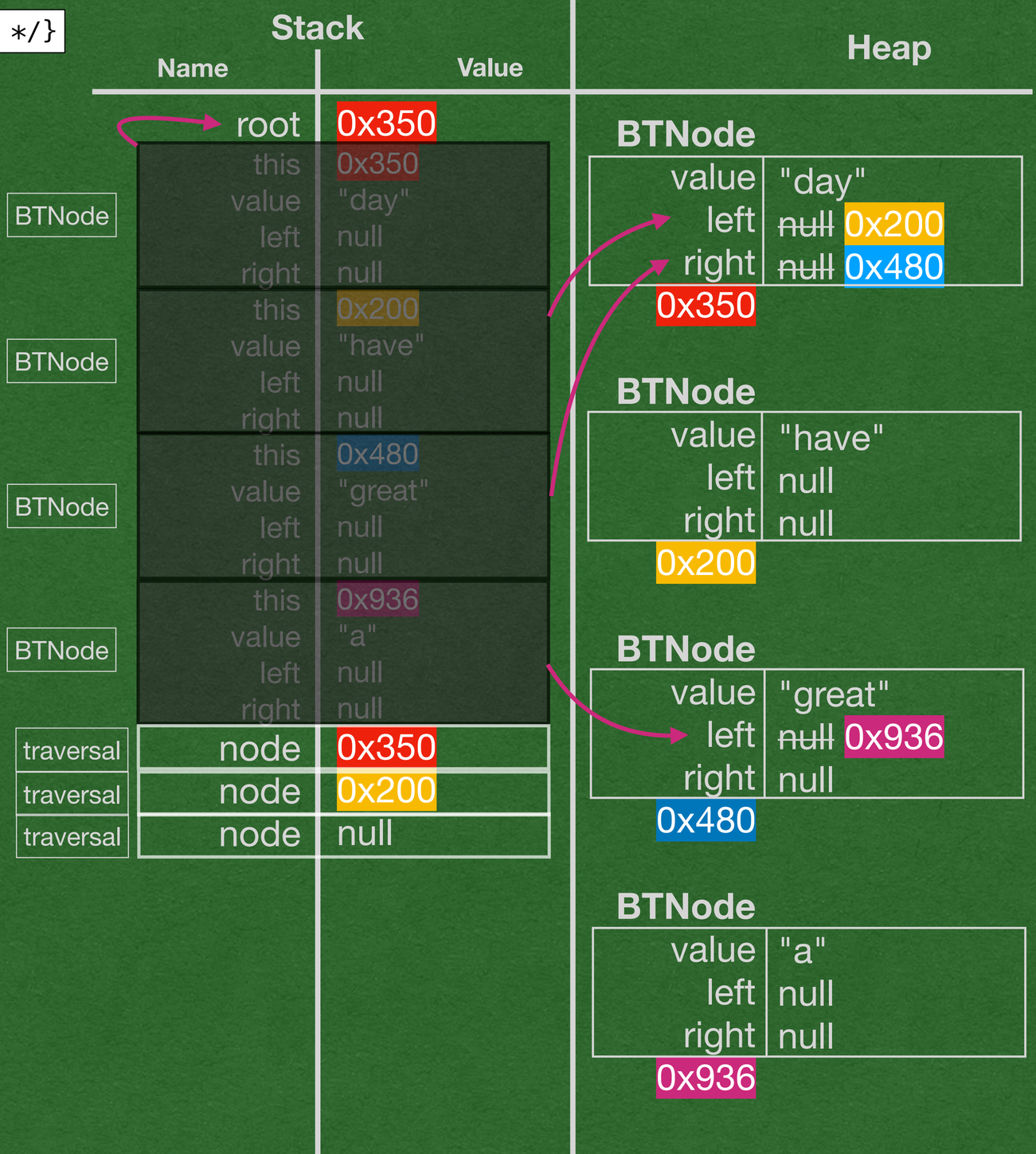
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```



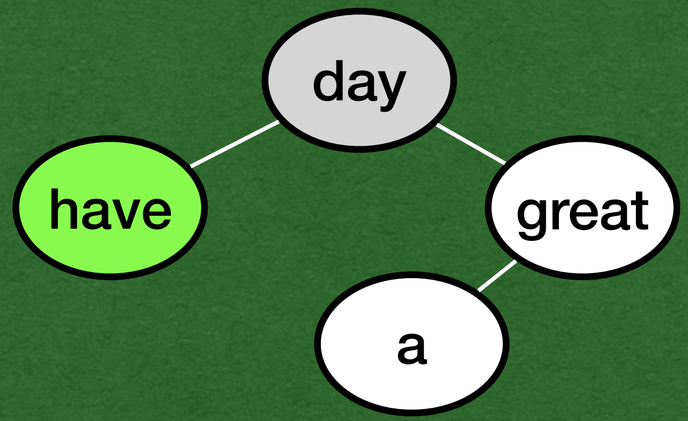

**in/out**

- Left node makes another recursive call on its left child



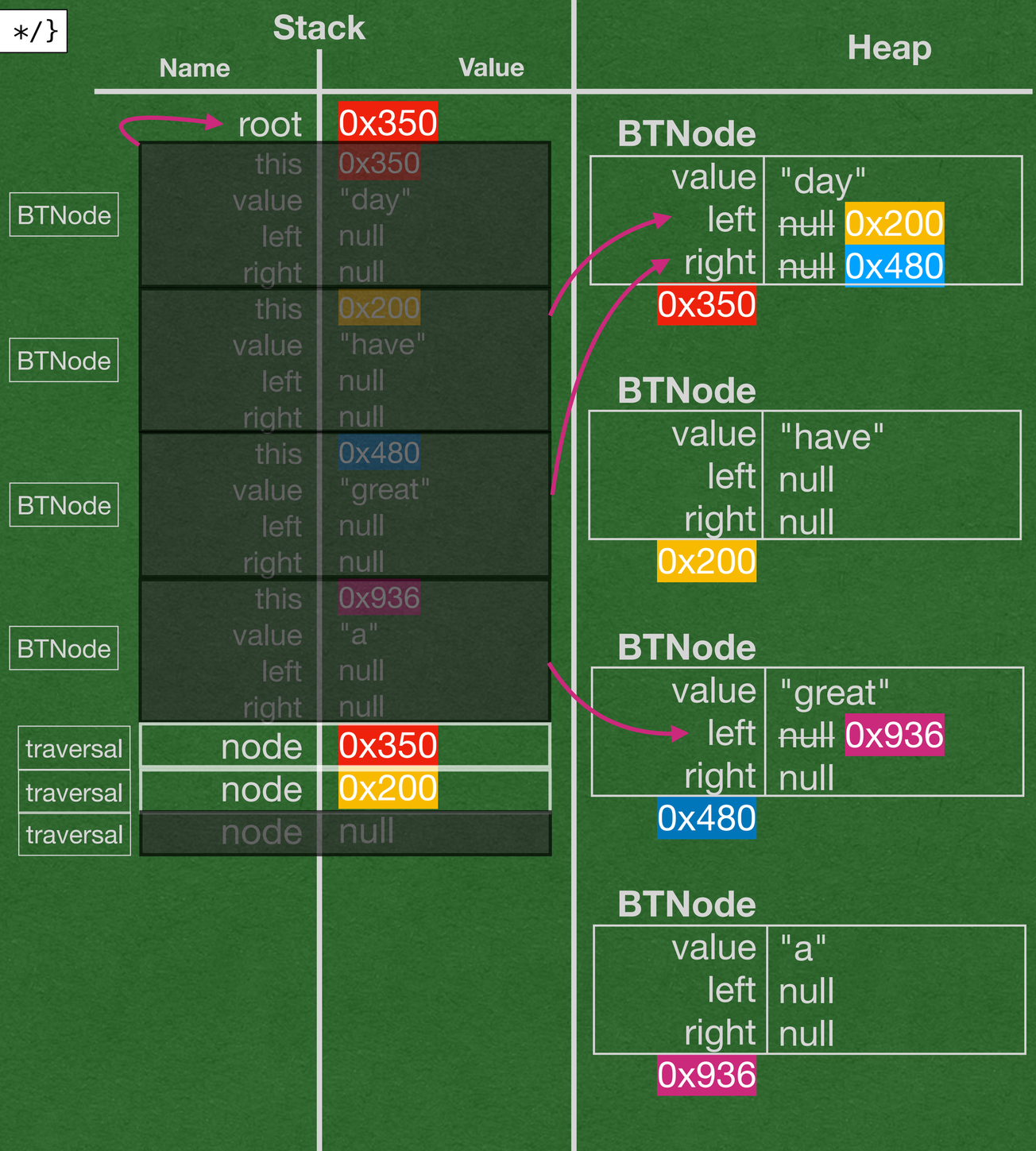
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```



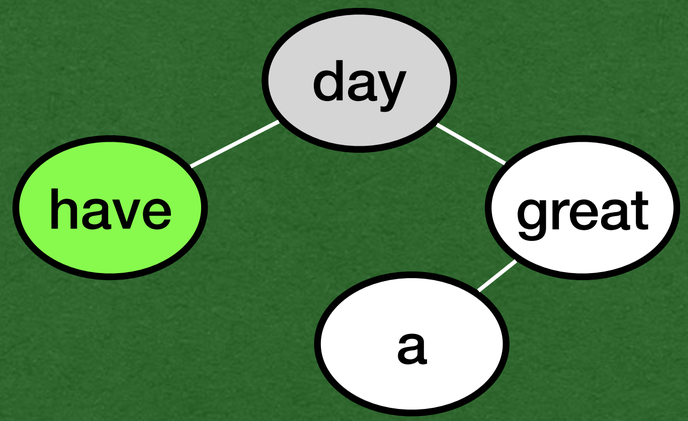

**in/out**

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



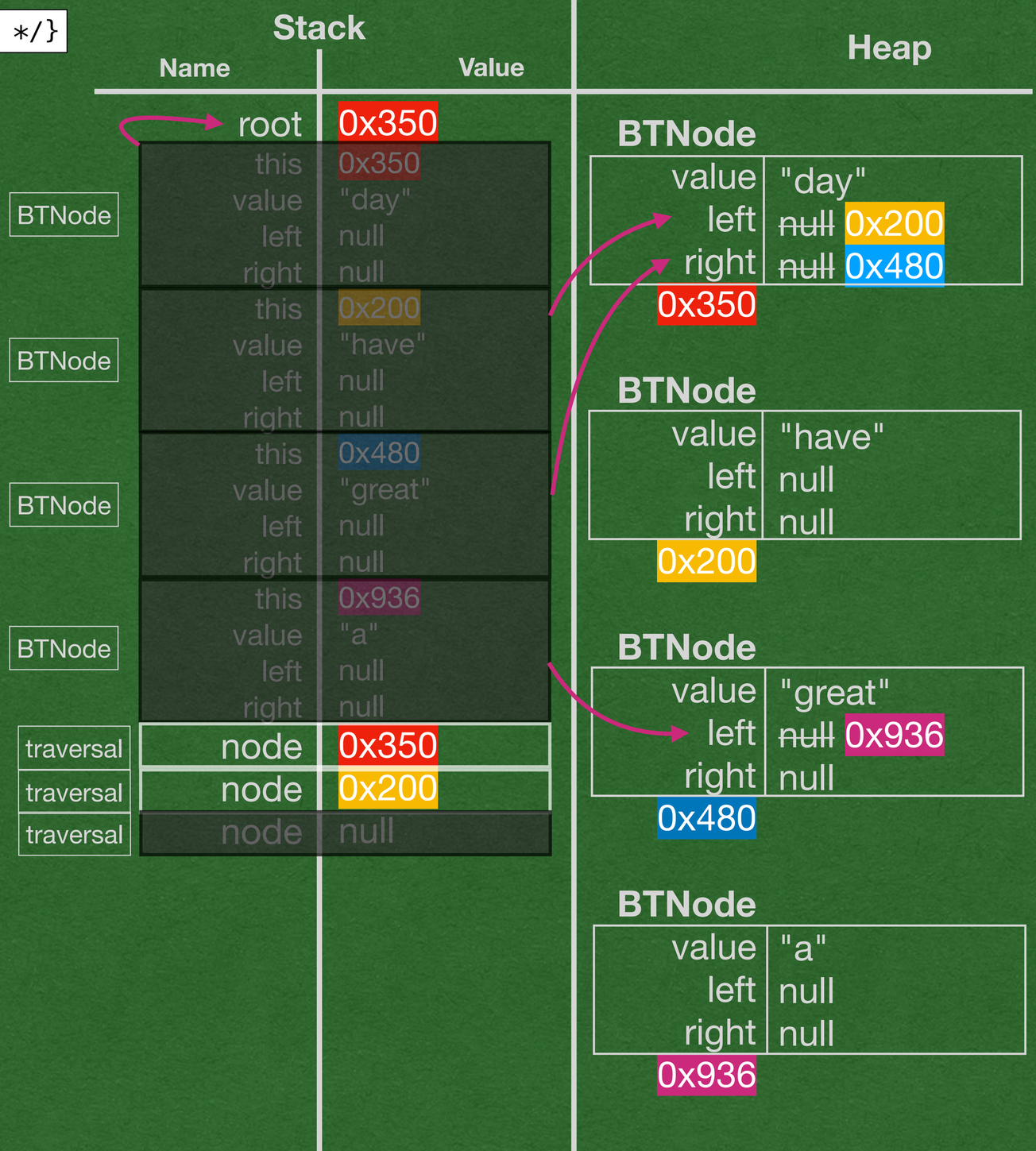
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```



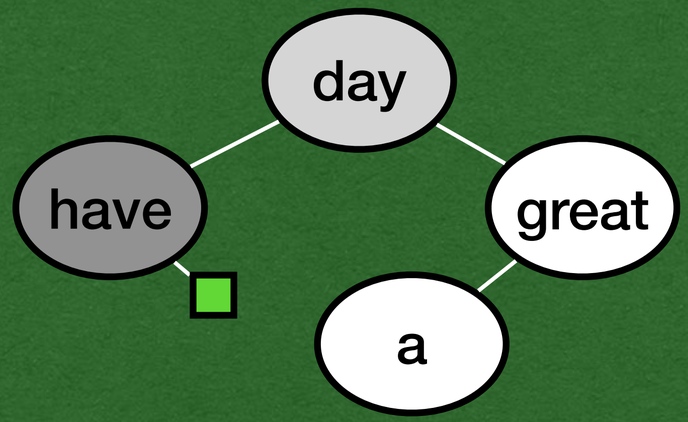

**in/out**

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



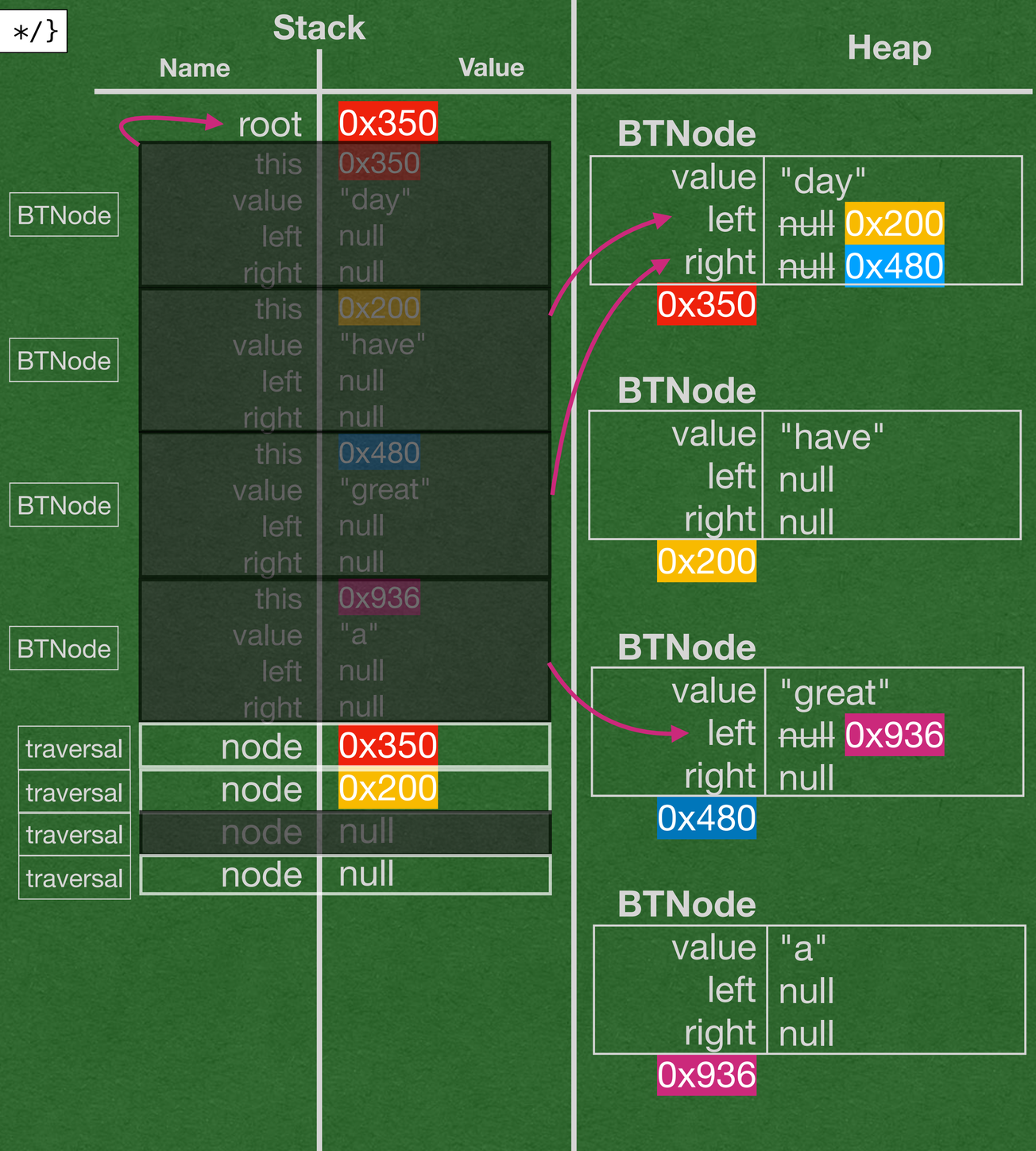
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```



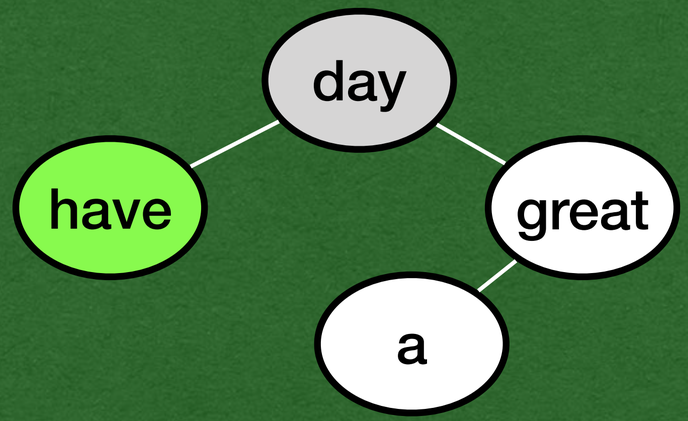

**in/out**

- Right recursive call is also for a null node



```
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);
    traversal(root);
}
```




**in/out**

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

**Stack**

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

**Heap**

**BTNode** (0x350)

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** (0x200)

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** (0x480)

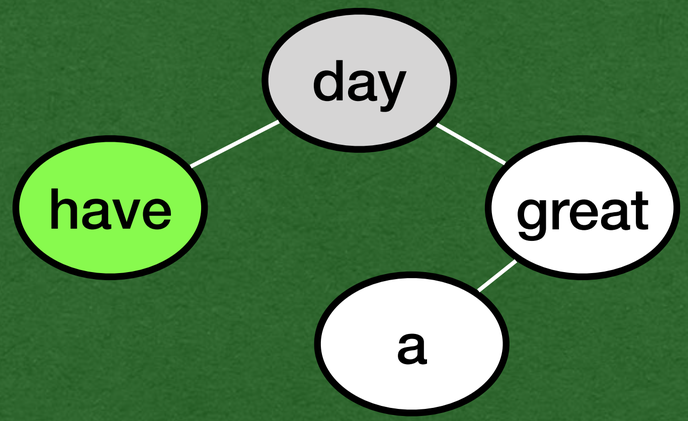
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** (0x936)

| | |
|---|---|
| value | "a" |
| left | null |
| right | null |

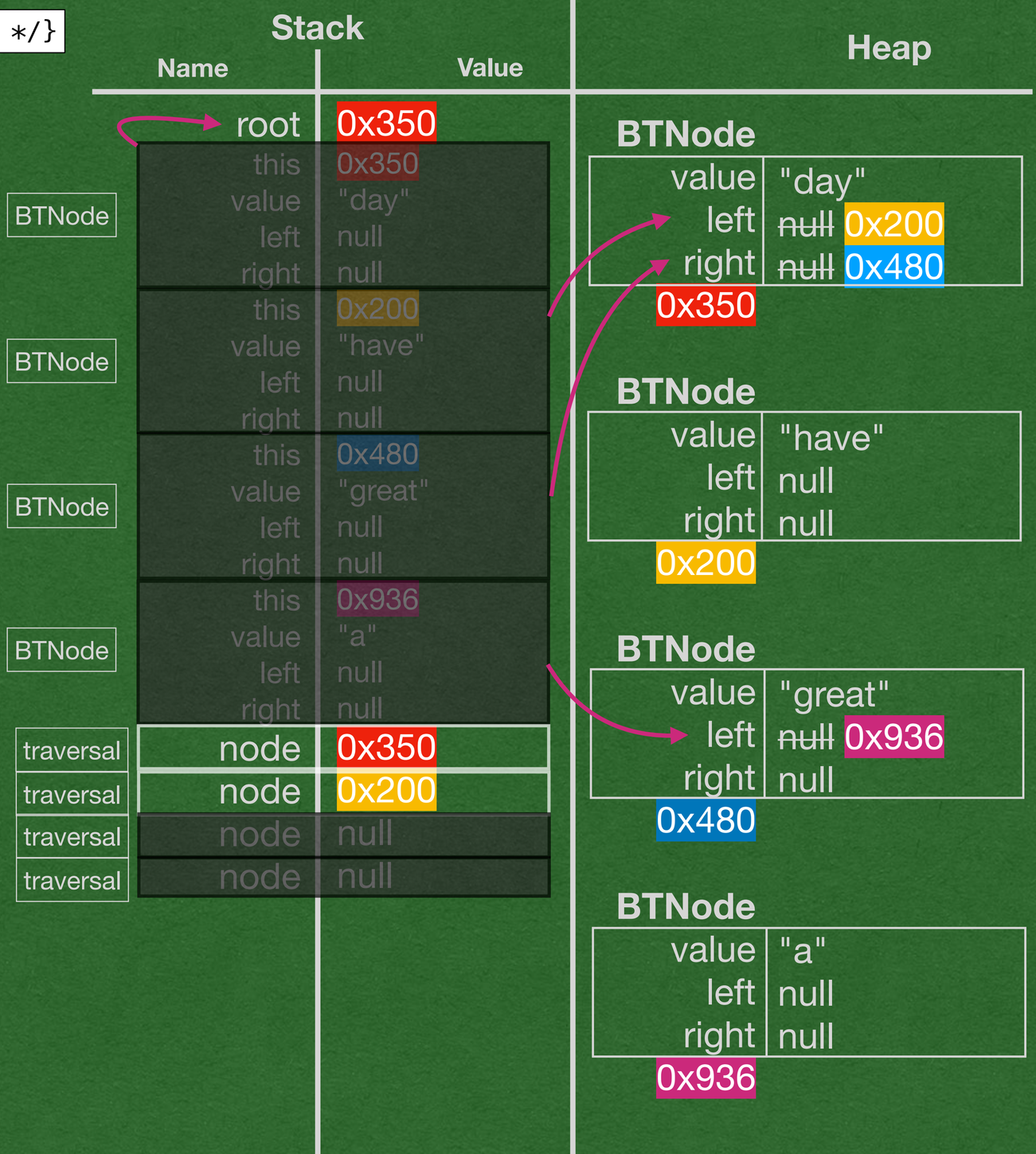
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```




**in/out**

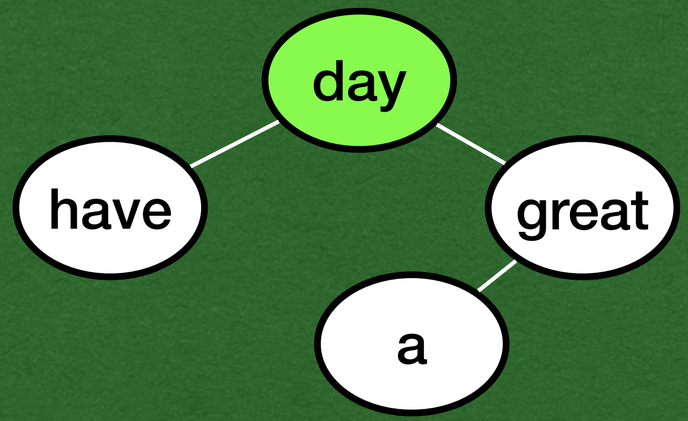
**have**




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

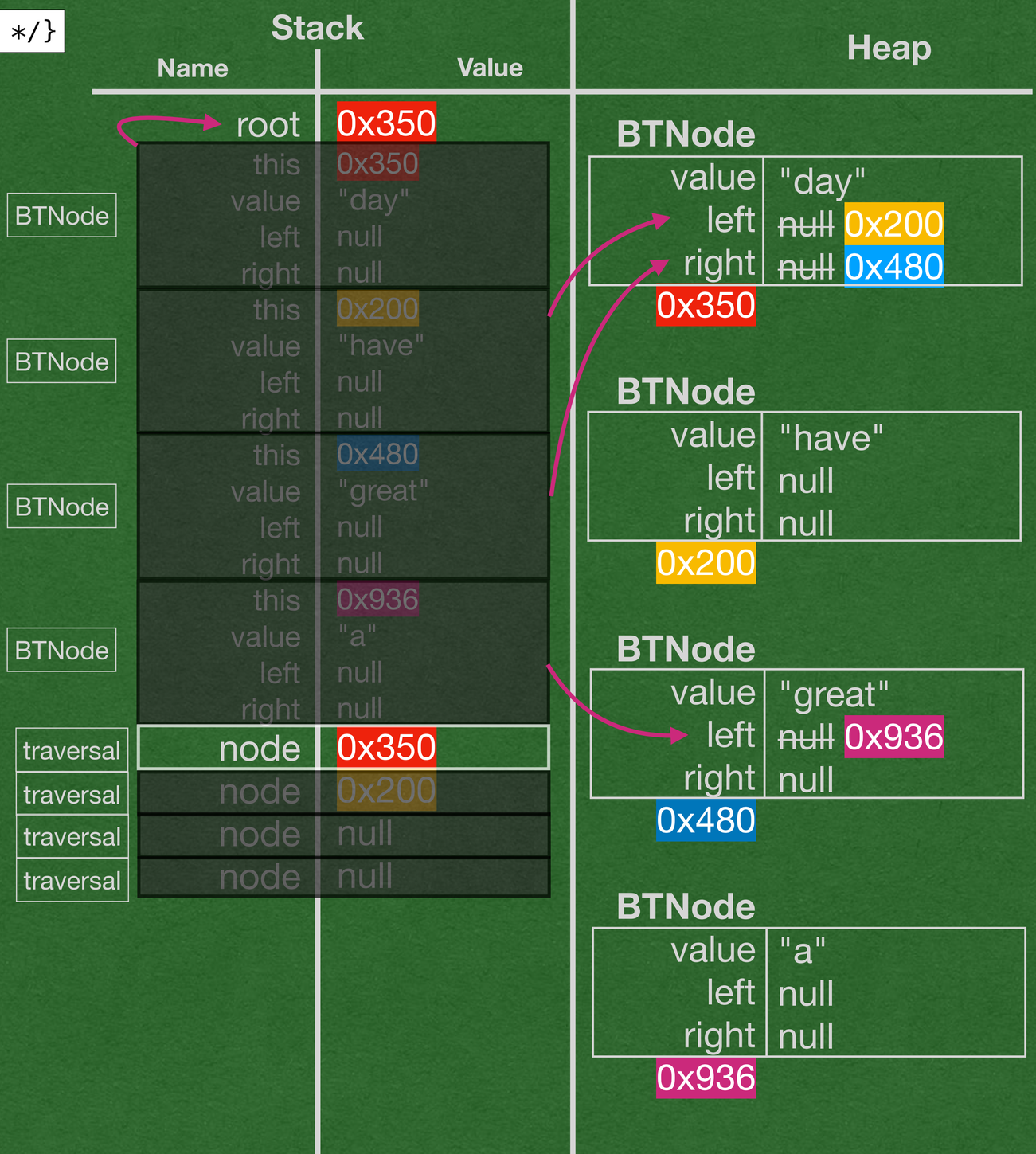
```
public BTNode(A value, BTNode<A> left, BTNode<A> right) {/* ... */}
```

```
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);
    traversal(root);
}
```




**in/out**

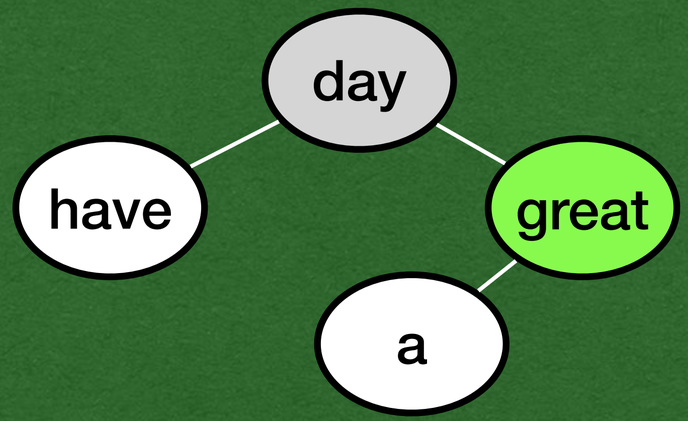
**have**




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

```
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);
    traversal(root);
}
```




**in/out**

**have**

- Make a recursive call to the right of the root

**Stack**

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

**Heap**

**BTNode** (0x350)

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** (0x200)

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** (0x480)

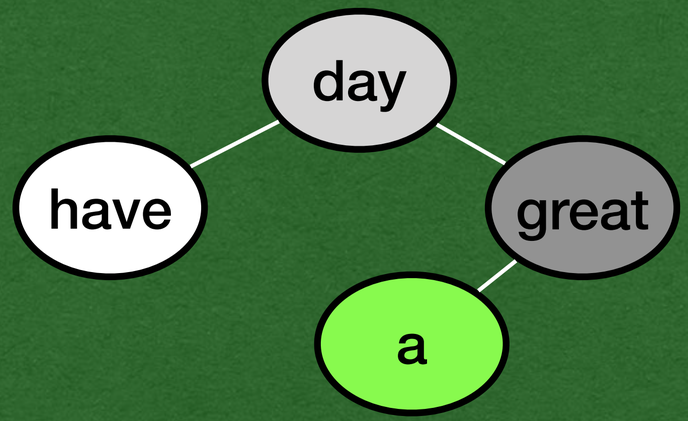
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** (0x936)

| | |
|---|---|
| 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);
    traversal(root);
}
```




**in/out**

**have**

- That node makes a recursive call to its left

**Stack**

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

**Heap**

**BTNode** 0x350

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** 0x200

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** 0x480

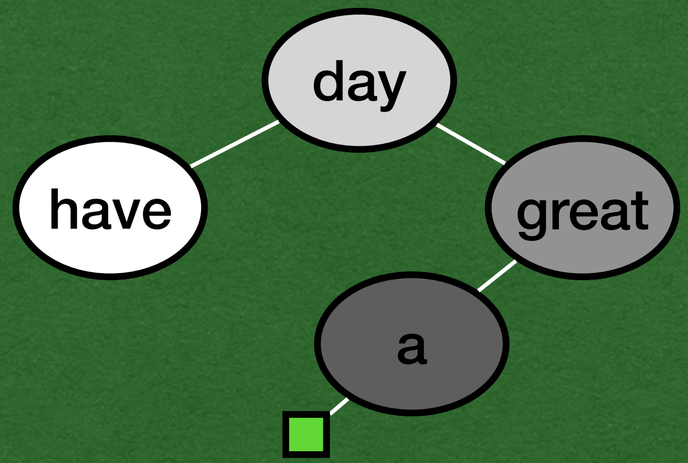
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** 0x936

| | |
|---|---|
| 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);
    traversal(root);
}
```




**in/out**

**have**

- Recursive call of null to the left
- Return

**Stack**

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

**Heap**

**BTNode** 0x350

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** 0x200

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** 0x480

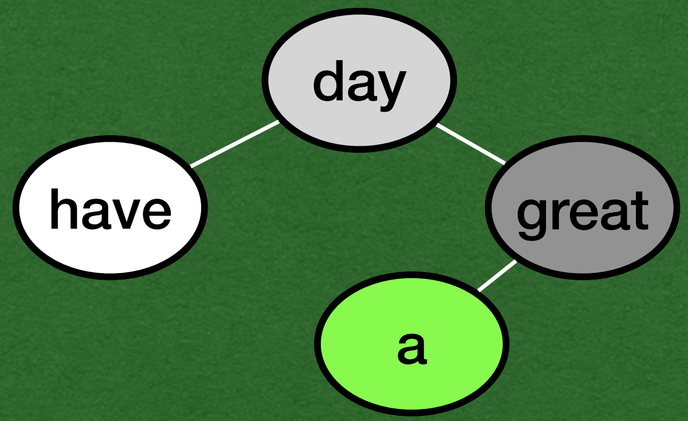
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** 0x936

| | |
|---|---|
| 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);
    traversal(root);
}
```




**in/out**

**have**

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

**Stack**

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

**Heap**

**BTNode** (0x350)

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** (0x200)

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** (0x480)

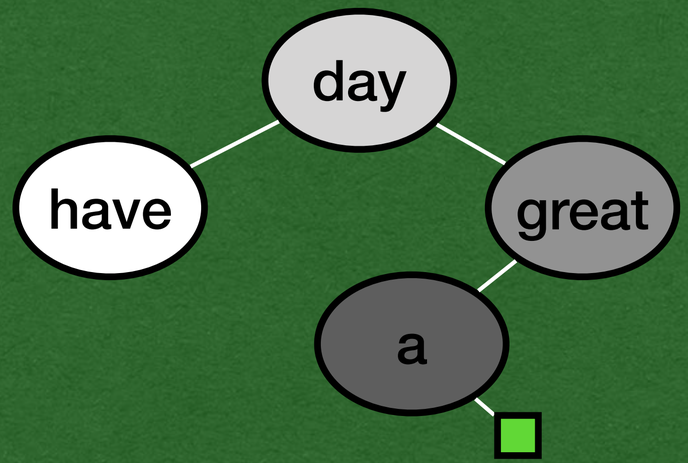
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** (0x936)

| | |
|---|---|
| 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);
    traversal(root);
}
```




**in/out**

**have**

- Make the right recursive call of null

**Stack**

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

**Heap**

**BTNode** (0x350)

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** (0x200)

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** (0x480)

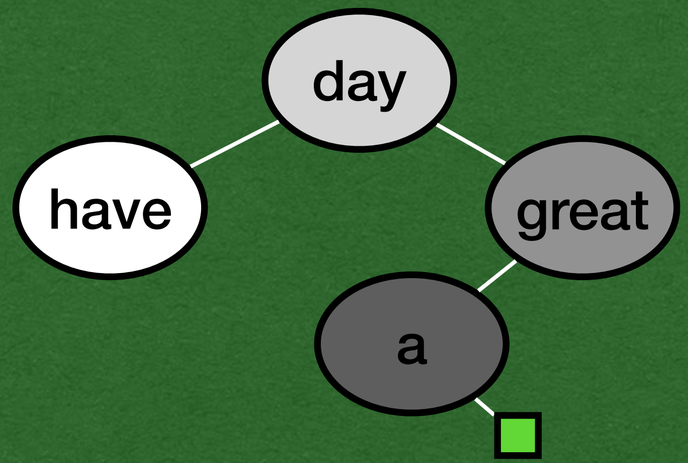
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** (0x936)

| | |
|---|---|
| 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);
    traversal(root);
}
```




**in/out**

**have**

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

**Stack**

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

**Heap**

**BTNode** 0x350

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** 0x200

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** 0x480

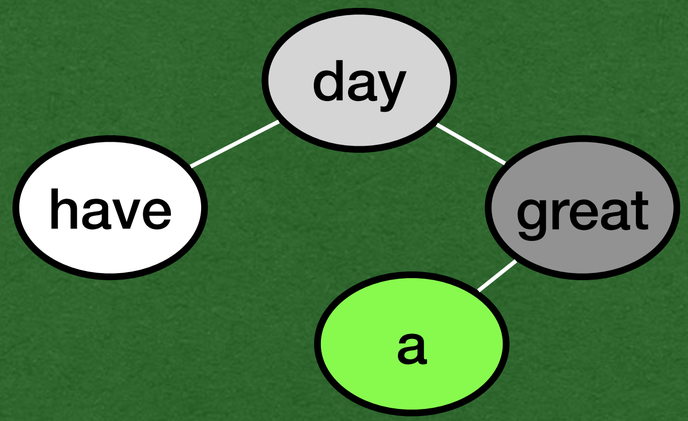
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** 0x936

| | |
|---|---|
| value | "a" |
| left | null |
| right | null |

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

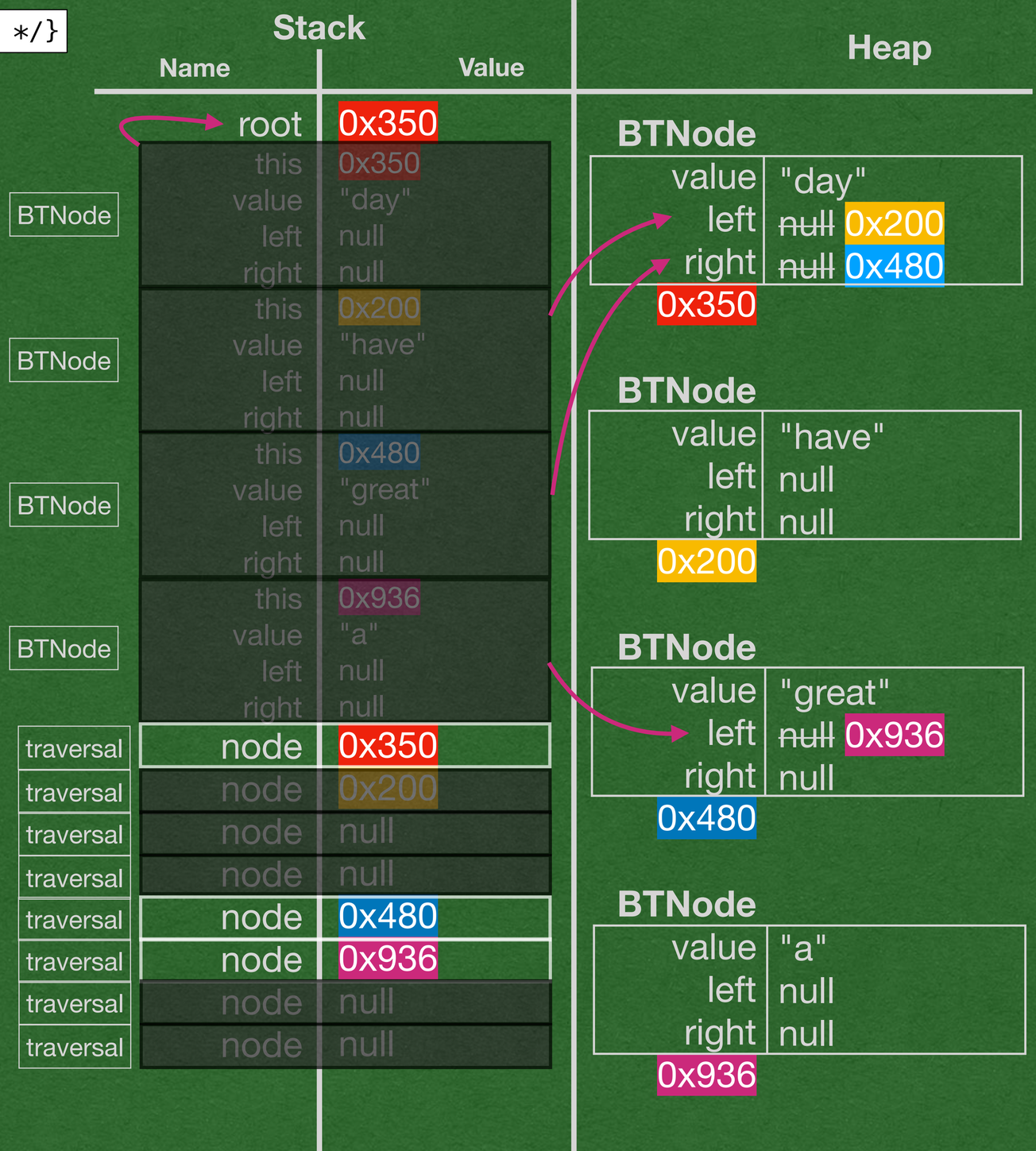
```
public static void traversal(BTNode<A> node) {
    if (node != null) {
        traversal(node.left);
        traversal(node.right);
        System.out.print(node.value + " ");
    }
}

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);
    traversal(root);
}
```




**in/out**

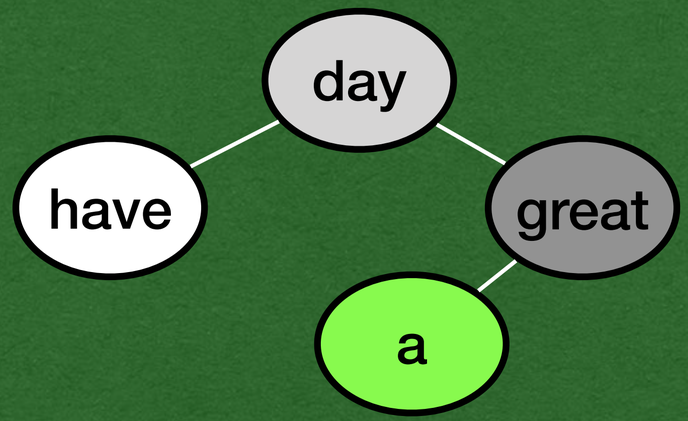
**have**

- This stack frame already made both recursive calls



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

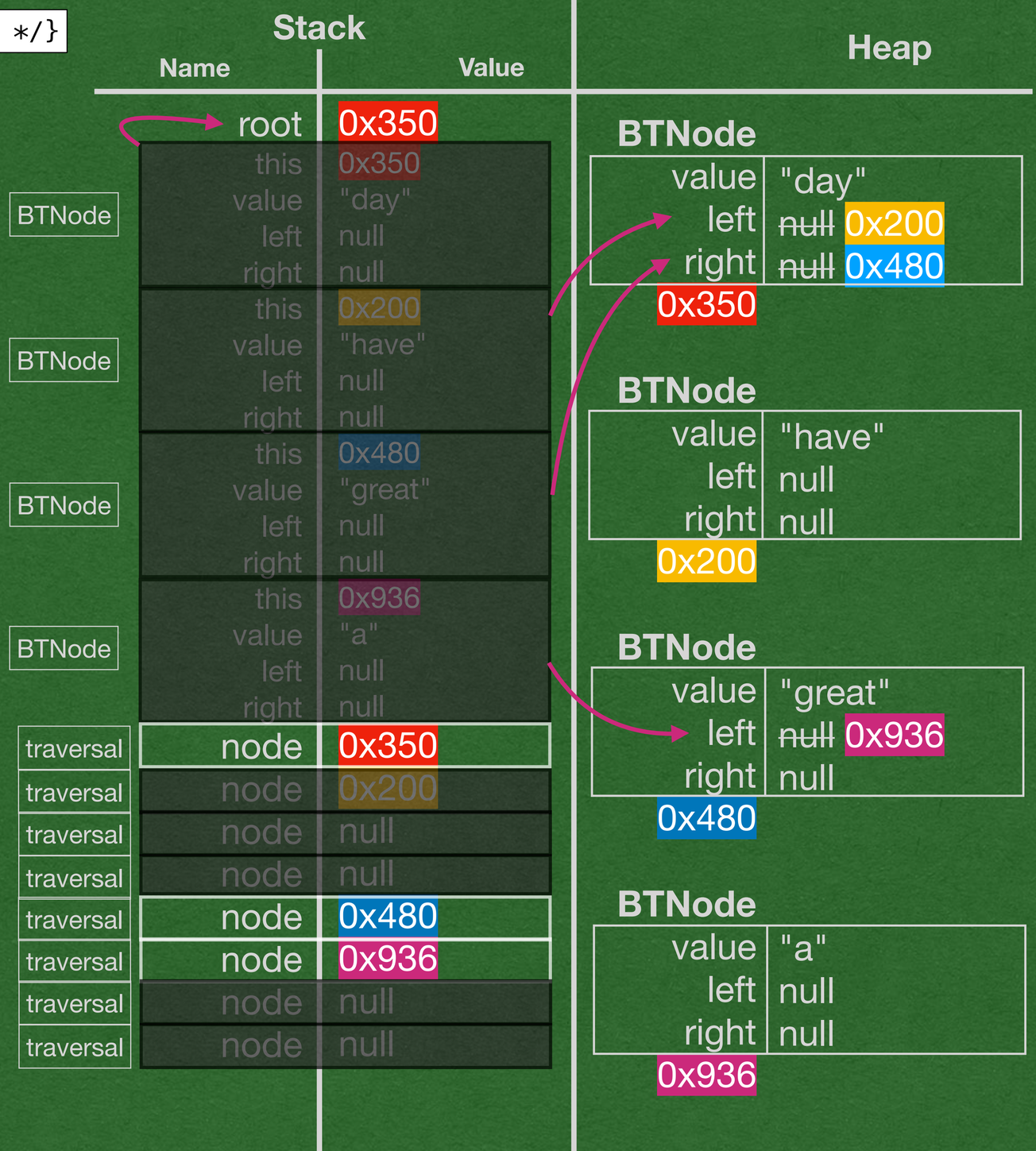
```
public static void traversal(BTNode<A> node) {
    if (node != null) {
        traversal(node.left);
        traversal(node.right);
        System.out.print(node.value + " ");
    }
}

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);
    traversal(root);
}
```




**in/out**

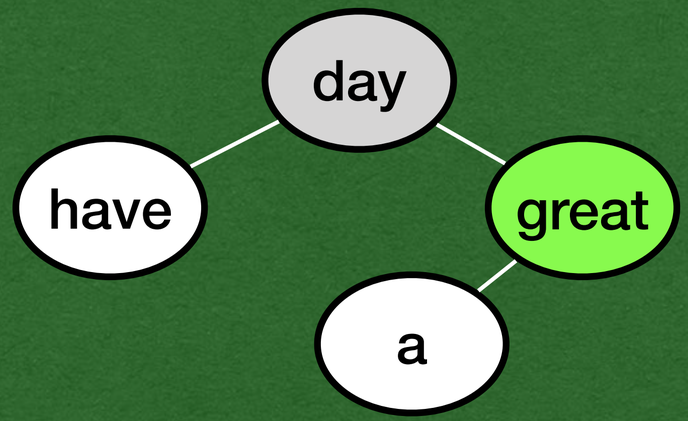
**have a**

- Print "a" to the screen



```
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);
    traversal(root);
}
```




**in/out**

**have a**

- Return back to the previous stack frame

**Stack**

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

**Heap**

**BTNode** 0x350

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** 0x200

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** 0x480

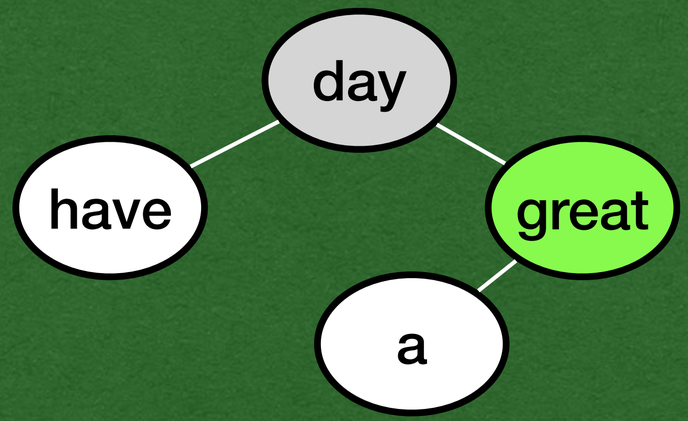
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** 0x936

| | |
|---|---|
| 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);
    traversal(root);
}
```




**in/out**

**have a**

- Make the right recursive call

**Stack**

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

**Heap**

**BTNode** (0x350)

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** (0x200)

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** (0x480)

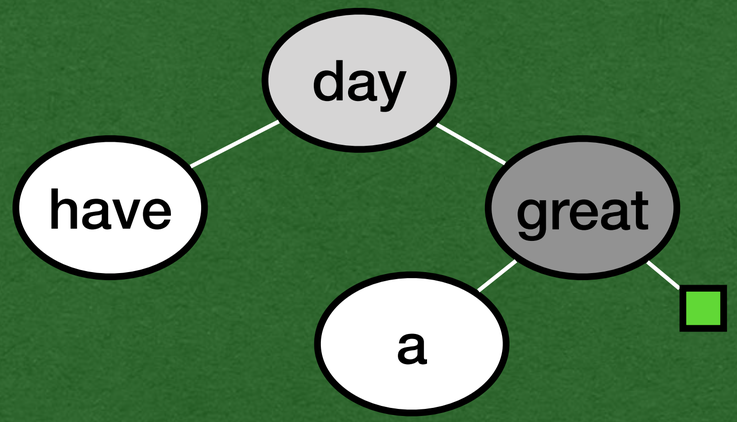
| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** (0x936)

| | |
|---|---|
| value | "a" |
| left | null |
| right | null |

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

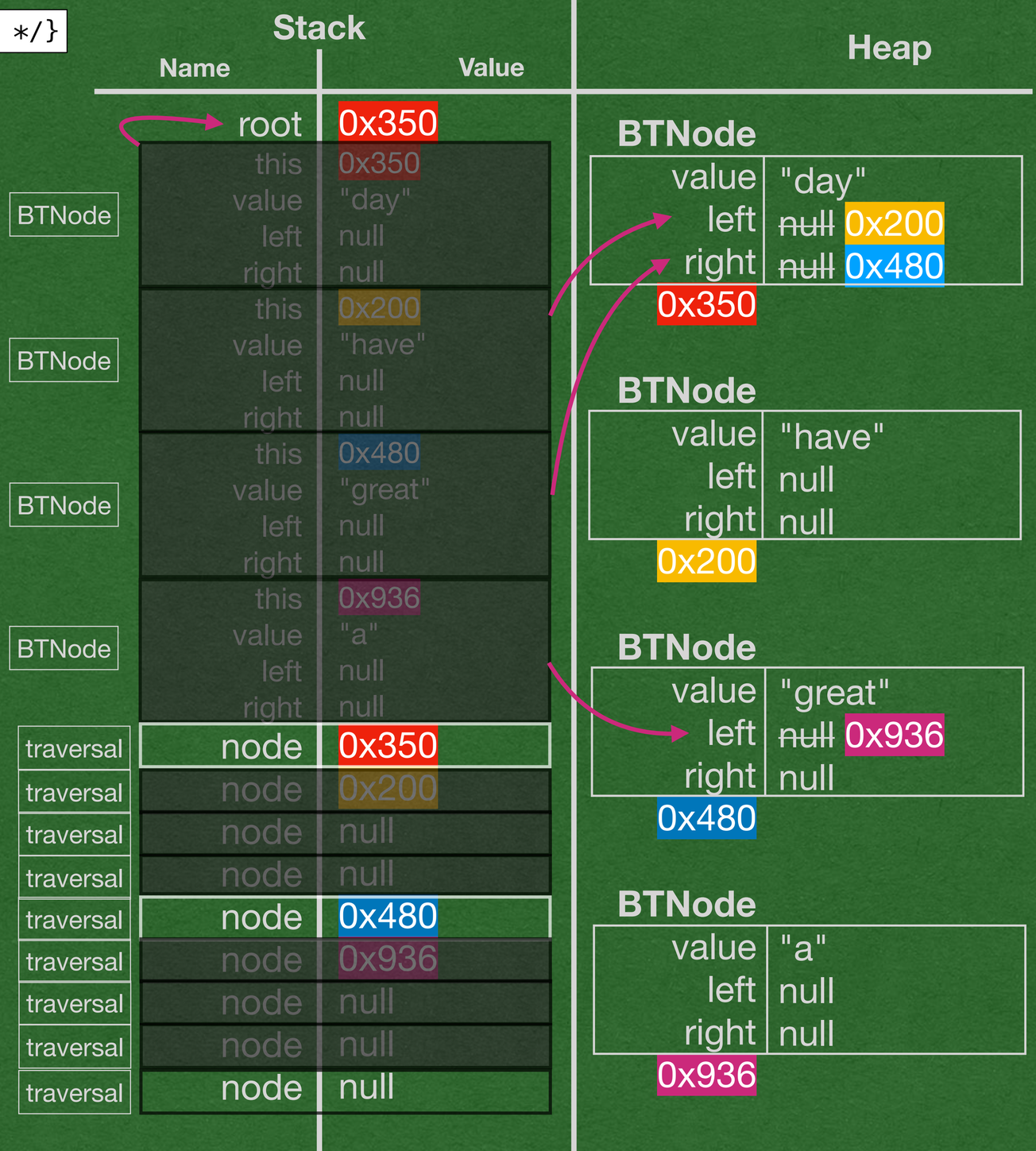
```
public static void traversal(BTNode<A> node) {
    if (node != null) {
        traversal(node.left);
        traversal(node.right);
        System.out.print(node.value + " ");
    }
}

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);
    traversal(root);
}
```




**in/out**

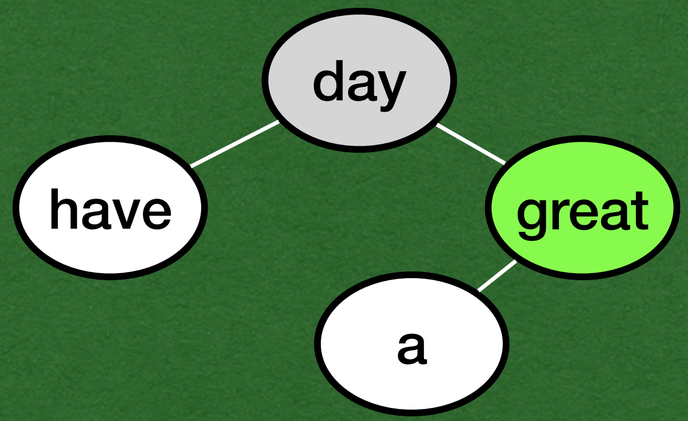
**have a**

- Base case of null
- Just return



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

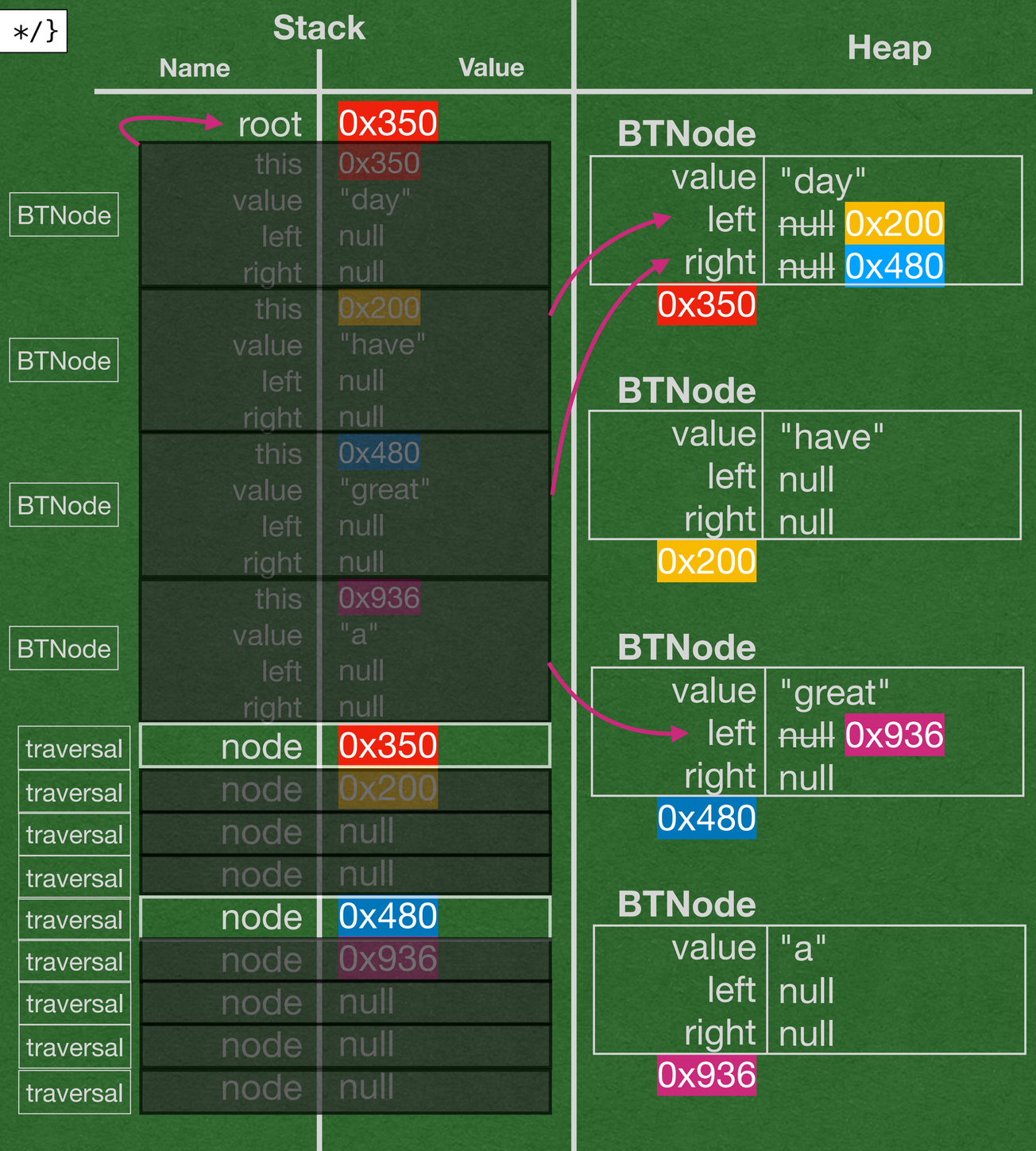
```
public static void traversal(BTNode<A> node) {
    if (node != null) {
        traversal(node.left);
        traversal(node.right);
        System.out.print(node.value + " ");
    }
}

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);
    traversal(root);
}
```




**in/out**

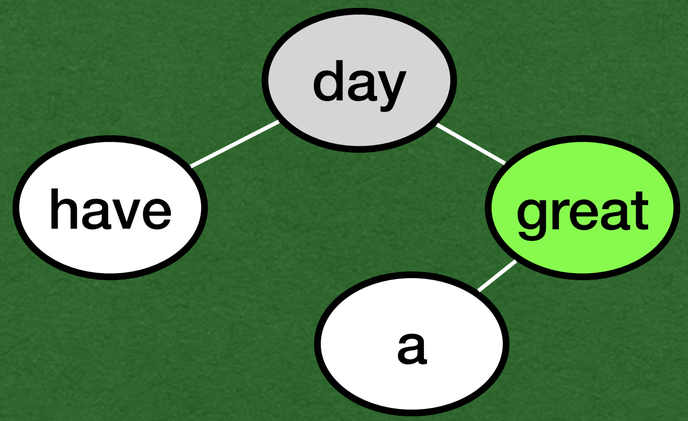
**have a**

- Stack frame done with both recursive calls



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

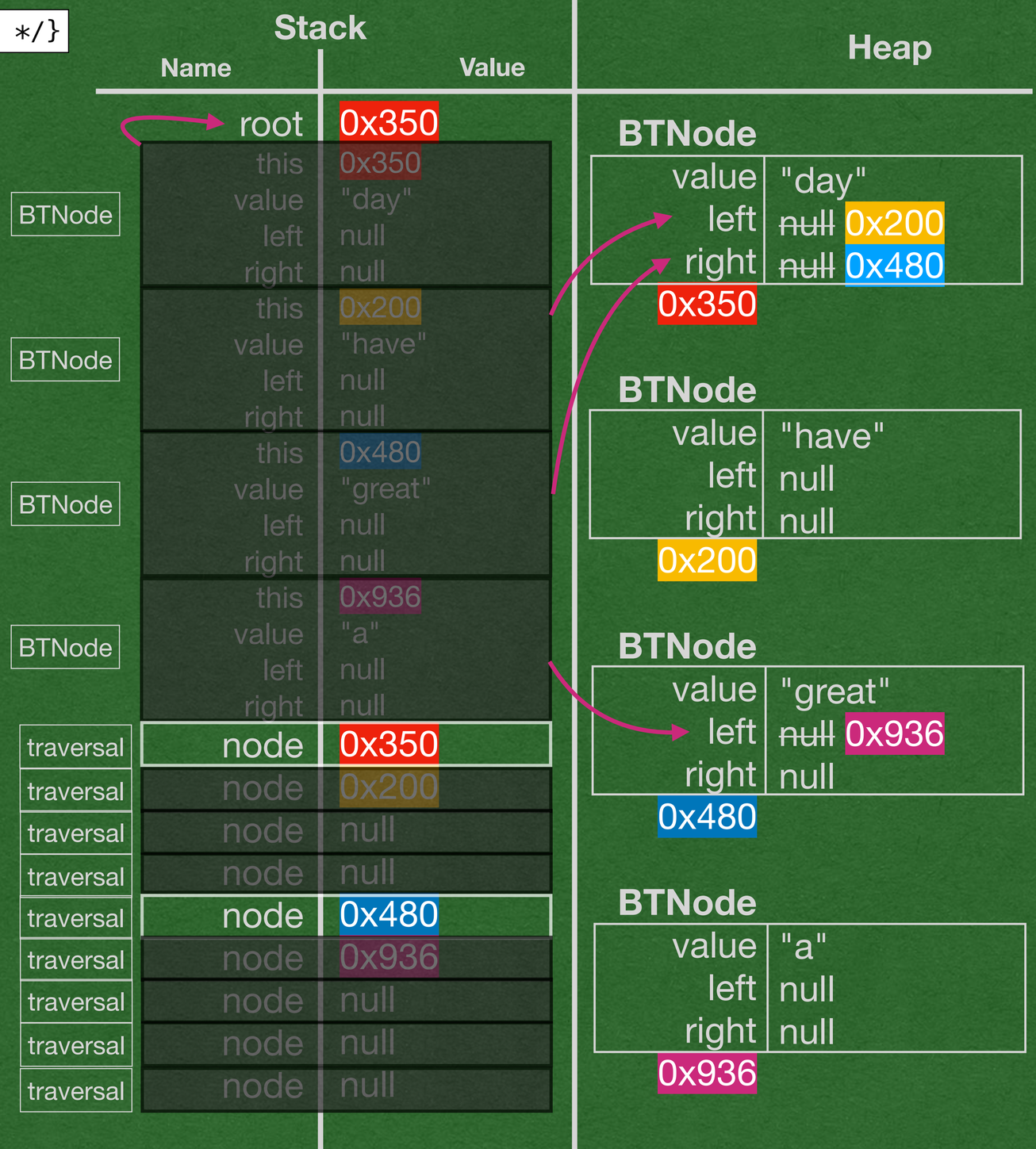
```
public static void traversal(BTNode<A> node) {
    if (node != null) {
        traversal(node.left);
        traversal(node.right);
        System.out.print(node.value + " ");
    }
}

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);
    traversal(root);
}
```




**in/out**

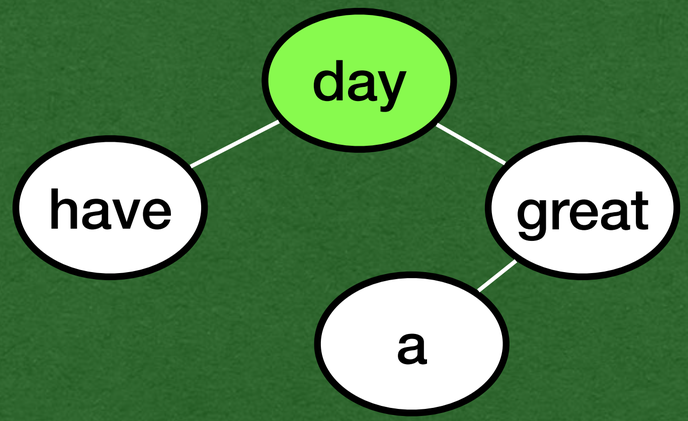
**have a great**

- Print "great" to the screen



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

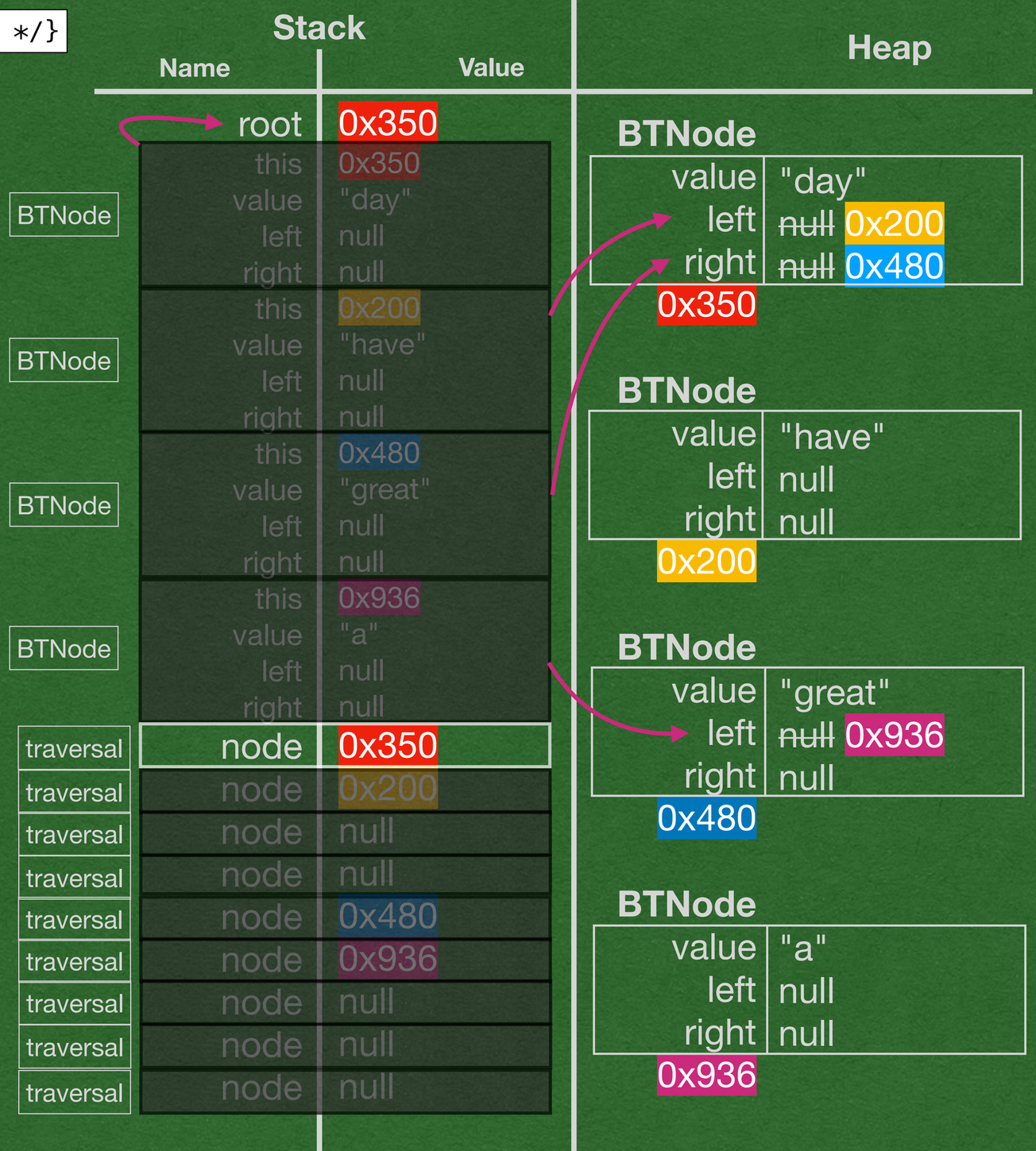
```
public static void traversal(BTNode<A> node) {
    if (node != null) {
        traversal(node.left);
        traversal(node.right);
        System.out.print(node.value + " ");
    }
}

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);
    traversal(root);
}
```




**in/out**

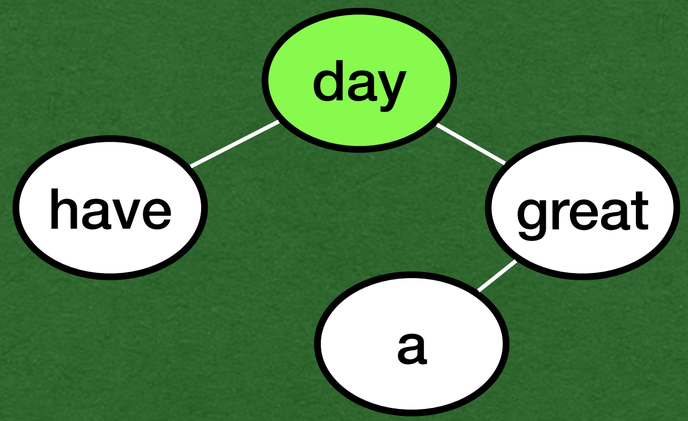
**have a great**

- Root node is now done with both recursive calls



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

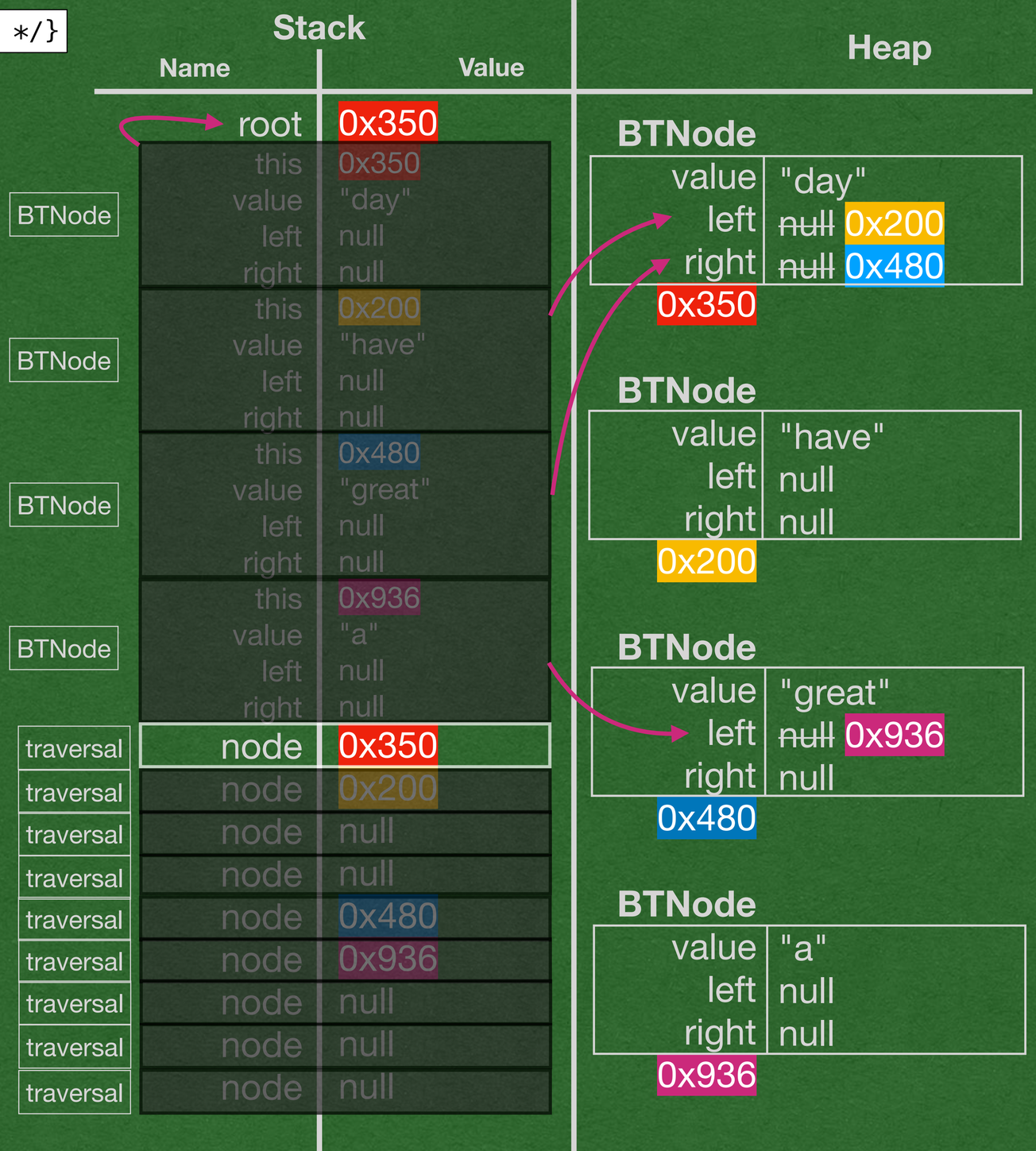
```
public static void traversal(BTNode<A> node) {
    if (node != null) {
        traversal(node.left);
        traversal(node.right);
        System.out.print(node.value + " ");
    }
}

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);
    traversal(root);
}
```




**in/out**

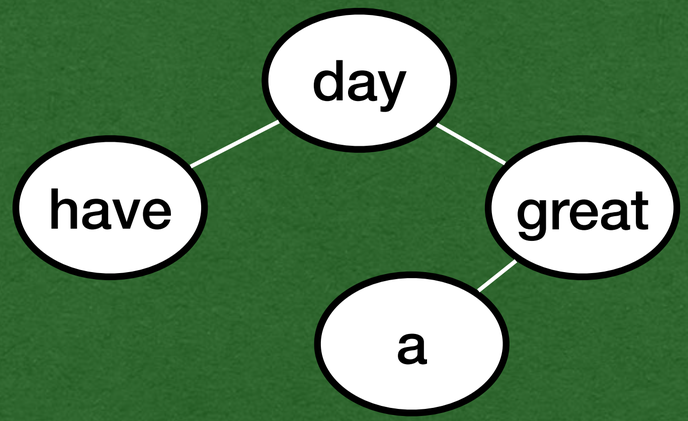
**have a great day**

- Print "day" to the screen



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

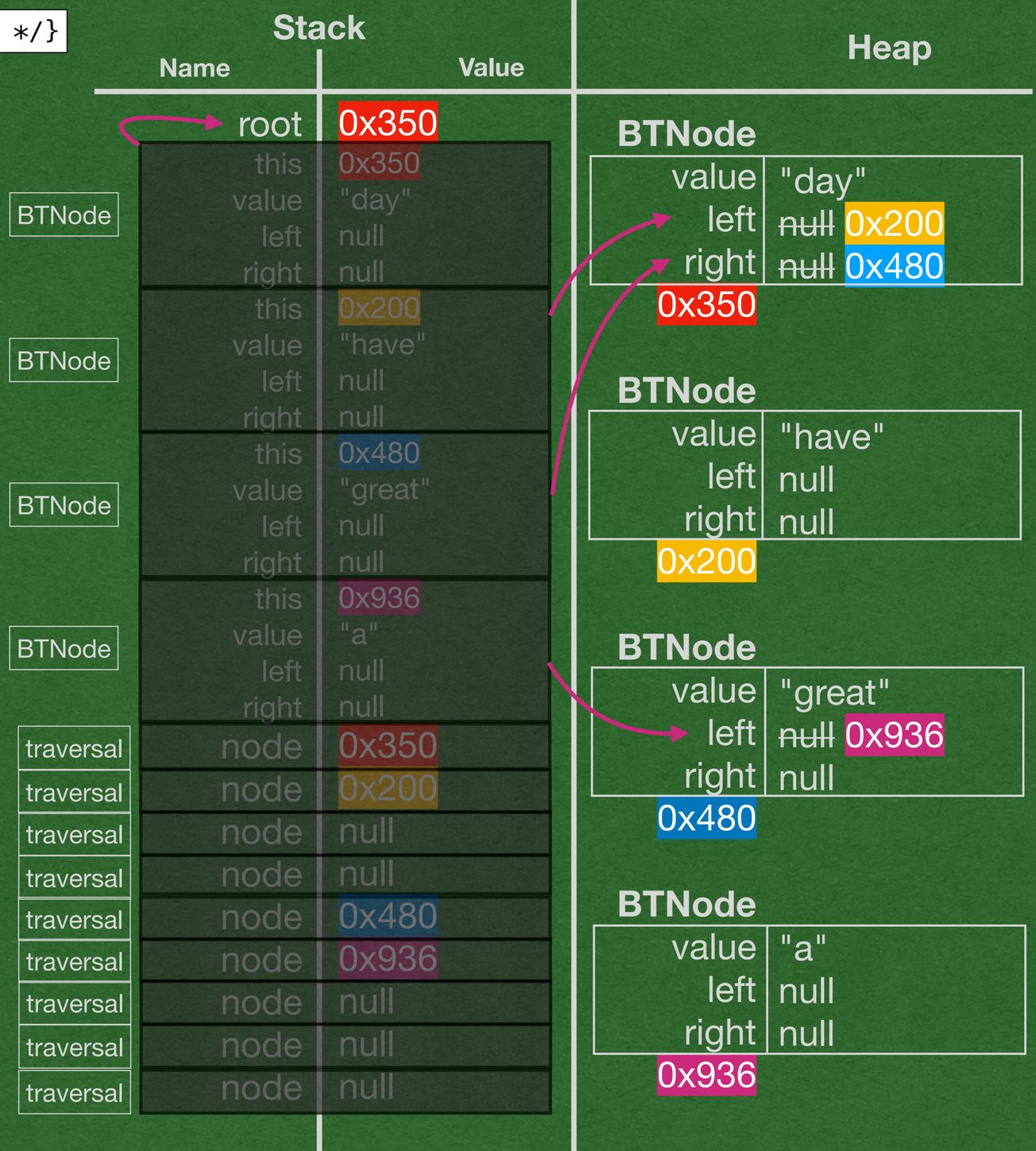
```
public static void traversal(BTNode<A> node) {
    if (node != null) {
        traversal(node.left);
        traversal(node.right);
        System.out.print(node.value + " ");
    }
}

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);
    traversal(root);
}
```




**in/out**

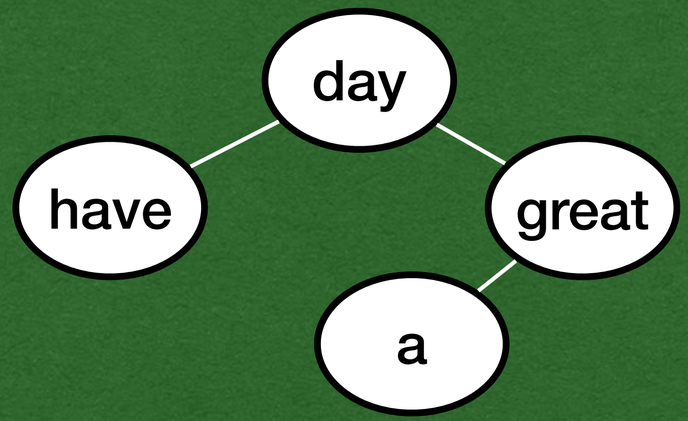
**have a great day**

- The traversal is over
- All stack frames have returned



```
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);
    traversal(root);
}
```




**in/out**

**have a great day**

- Program ends

**Stack**

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

**Heap**

**BTNode** (0x350)

| | |
|---|---|
| value | "day" |
| left | ~~null~~ 0x200 |
| right | ~~null~~ 0x480 |

**BTNode** (0x200)

| | |
|---|---|
| value | "have" |
| left | null |
| right | null |

**BTNode** (0x480)

| | |
|---|---|
| value | "great" |
| left | ~~null~~ 0x936 |
| right | null |

**BTNode** (0x936)

| | |
|---|---|
| value | "a" |
| left | null |
| right | null |