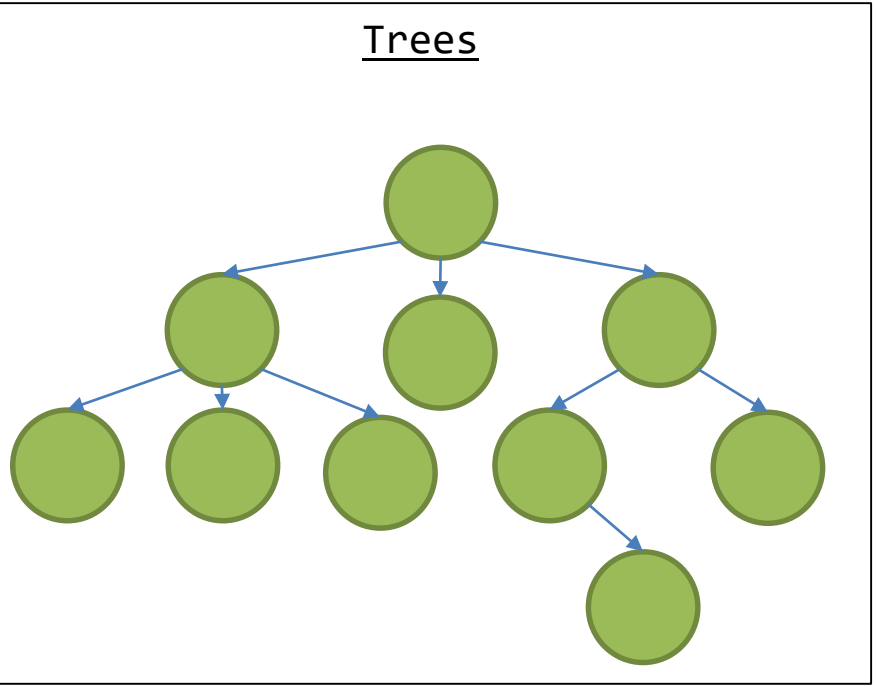
File Edit View Help

File Edit View Help

Binary Search Trees

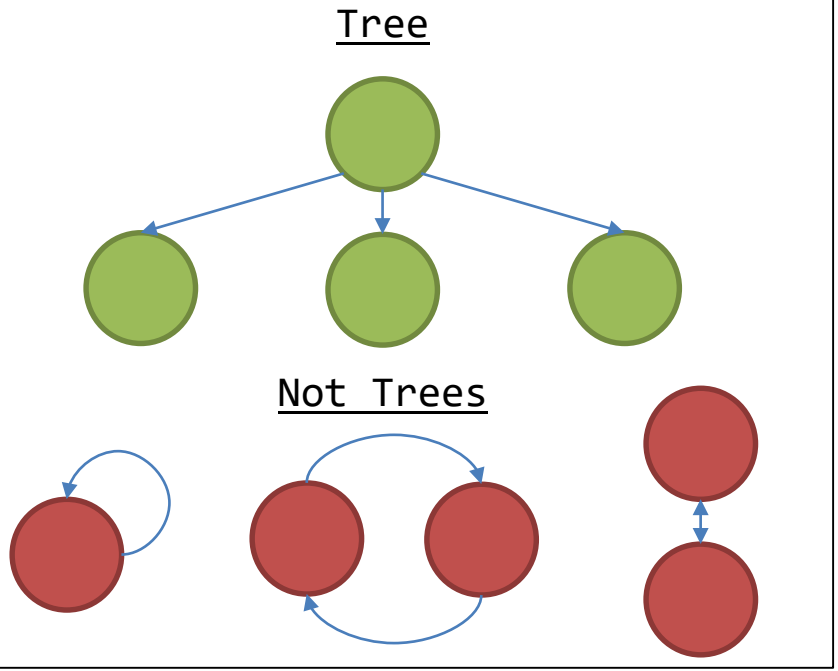
Trees

- Definition: A data structure that can be defined recursively as a collection of nodes, where each node is a data structure consisting of a value, together with a list of references (edges) to nodes, with the constraints that no reference is duplicated, and none points to the root.



Trees

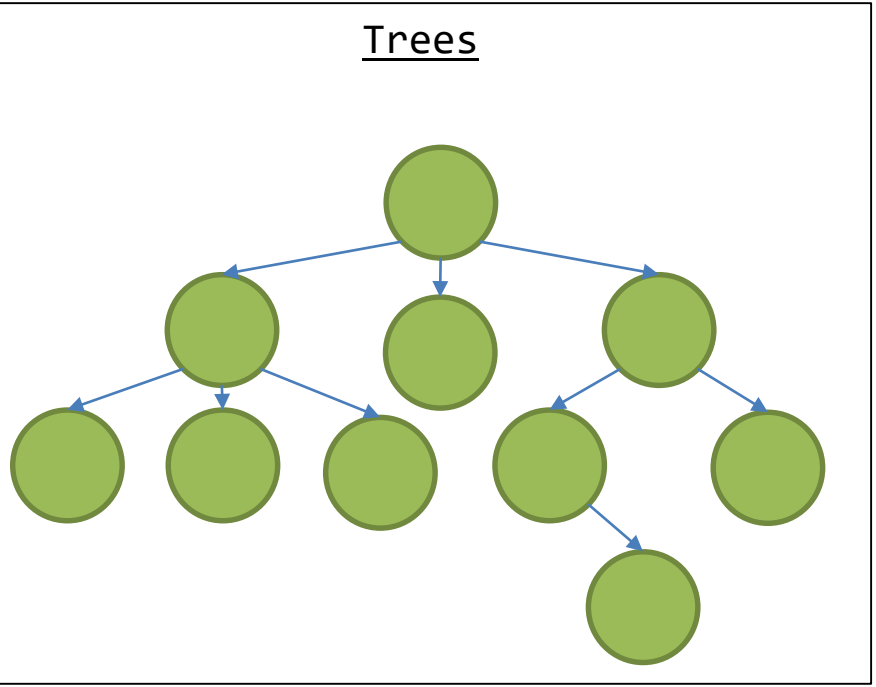
- Trees Have
  - Nodes
  - Edges
- Trees CANNOT
  - Contain Self-Referencing Edges
  - Have Cycles
  - Be Disjointed



Trees

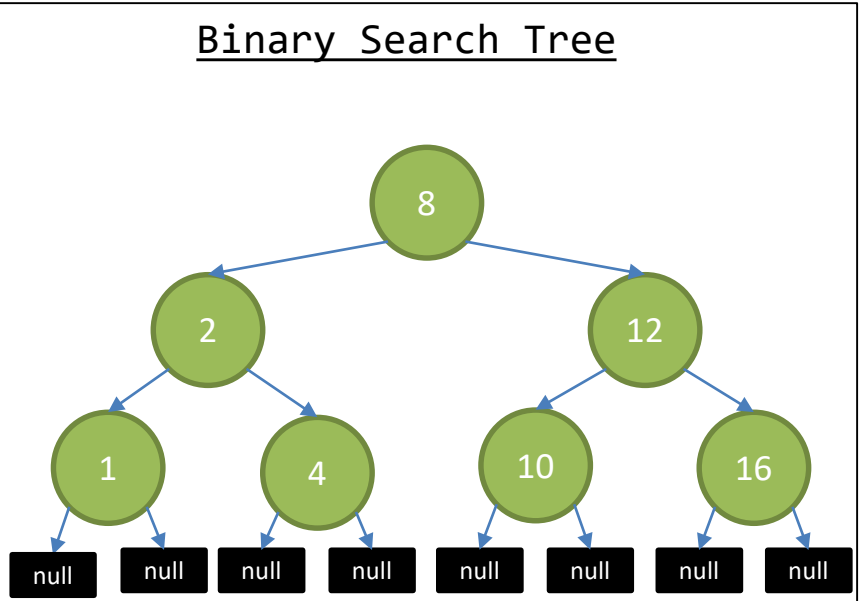
Common Terms

- **Root** – The top node in a tree.
- **Child** – A node's reference which is at a lower level
- **Parent** – The converse notion of *child*.
- **Siblings** – Nodes with the same parent.
- **Leaf** – a node with no children.
- **Degree** – number of sub trees of a node.
- **Edge** – connection between one node to another.
- **Path** – a sequence of nodes and edges connecting a node with a descendant.
- **Level** – The level of a node is defined by 1 + the number of connections between the node and the root.
- **Height of tree** –The height of a tree is the number of edges on the longest downward path between the root and a leaf.
- **Height of node** –The height of a node is the number of edges on the longest downward path between that node and a leaf.
- **Depth** –The depth of a node is the number of edges from the node to the tree's root node.



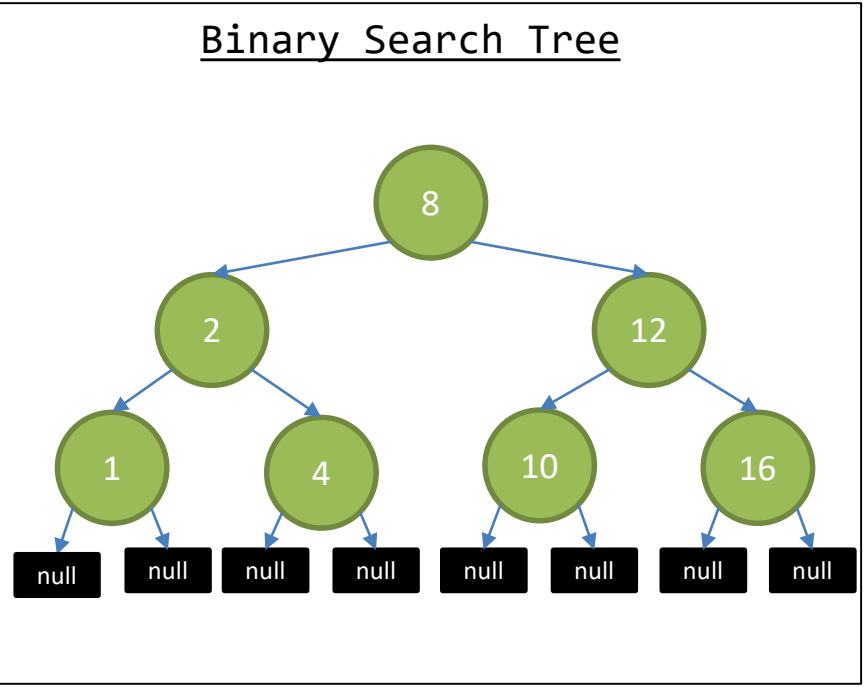
Binary Search Trees

- Tree Structure
- Node's data must be comparable
- Node's have at most two children
  - Left Child
  - Right Child
- Left child's value must be LESS THAN the parent's value
- Right child's value must be GREATER THAN the parent's value
- No Duplicate Values
- Assume Leaves are NULL references



Binary Search Trees

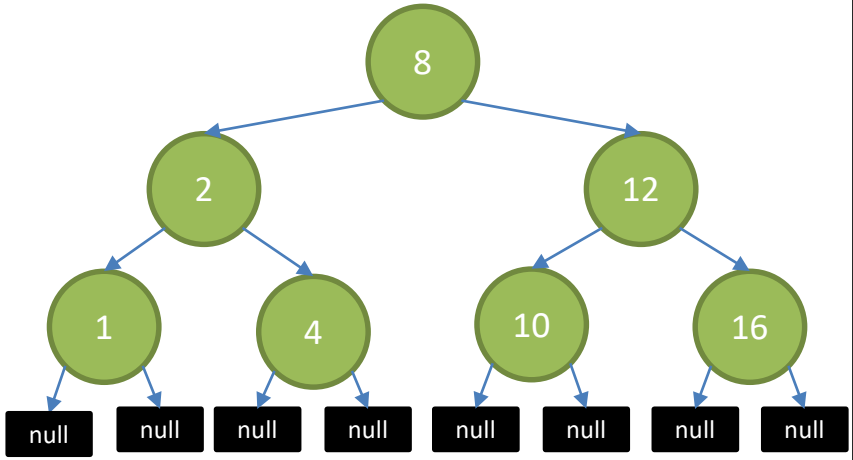
- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference



Binary Search Trees

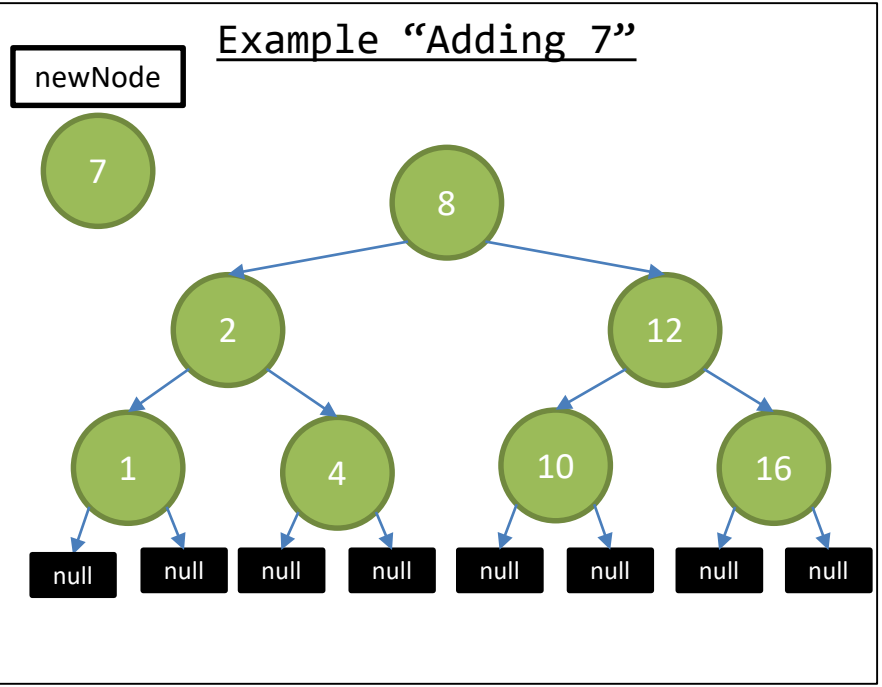
- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference

Example "Adding 7"



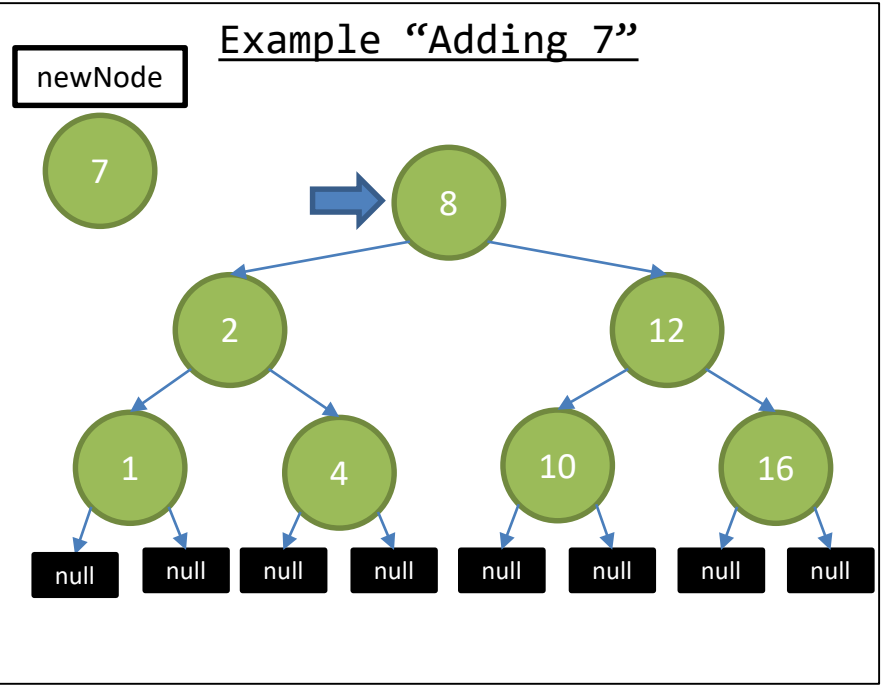
Binary Search Trees

- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference



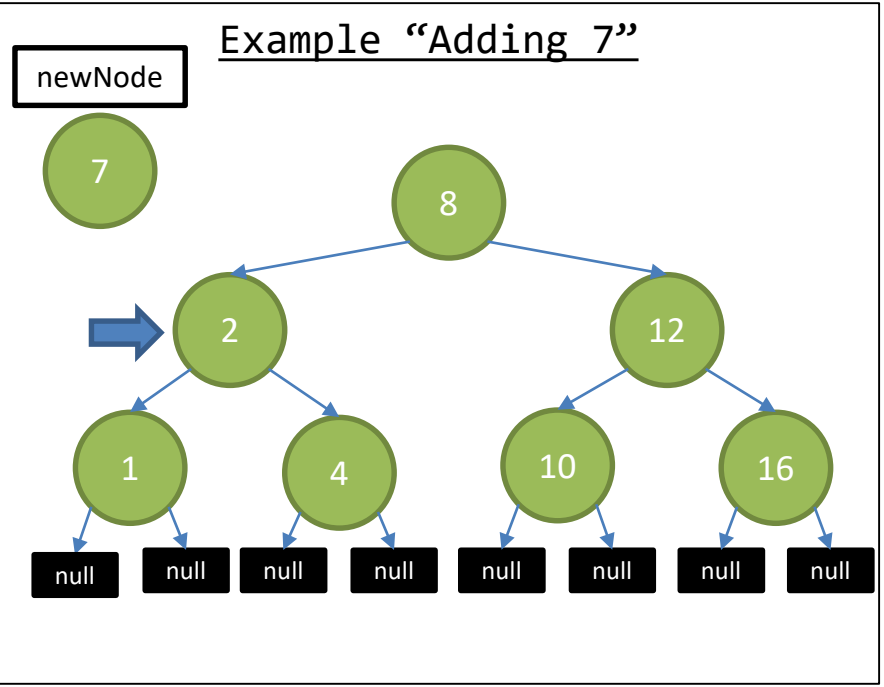
Binary Search Trees

- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference



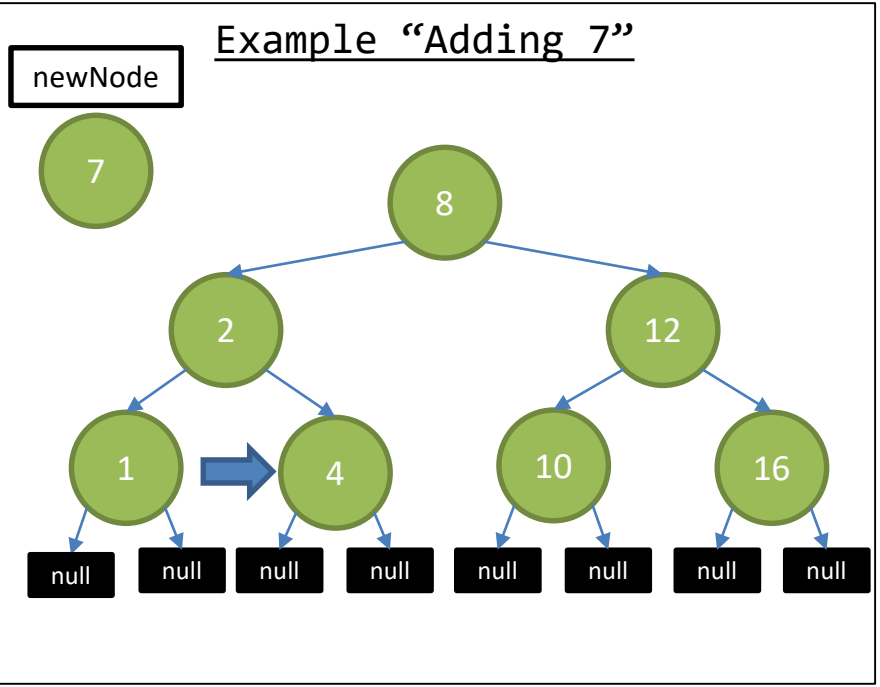
Binary Search Trees

- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference



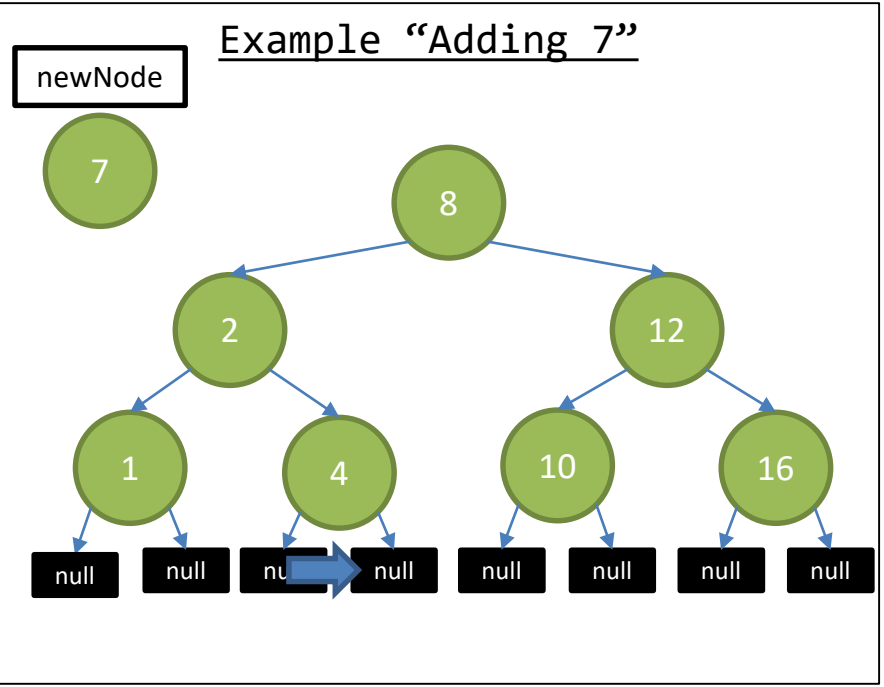
Binary Search Trees

- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference



Binary Search Trees

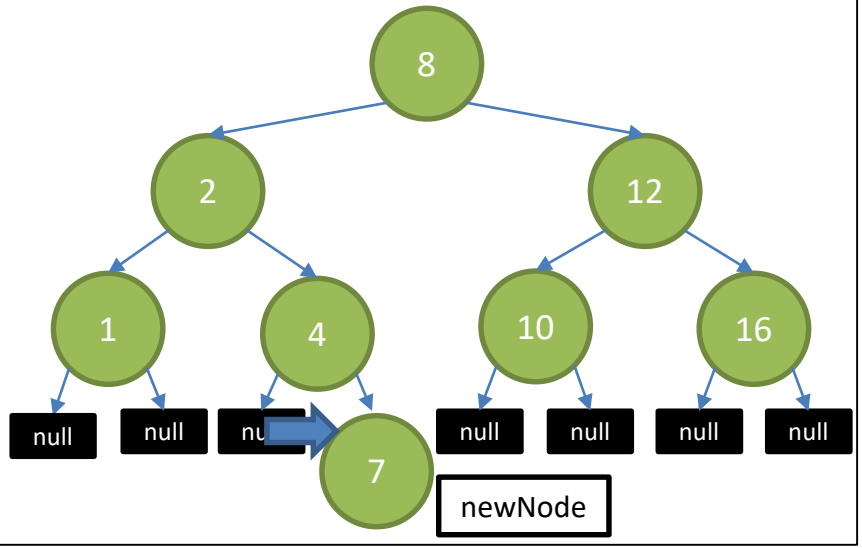
- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference



Binary Search Trees

- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference

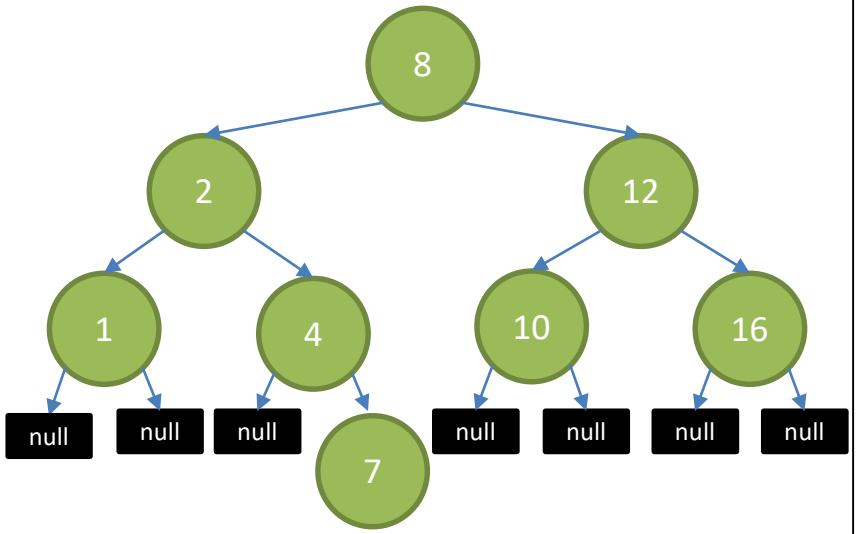
Example "Adding 7"



Binary Search Trees

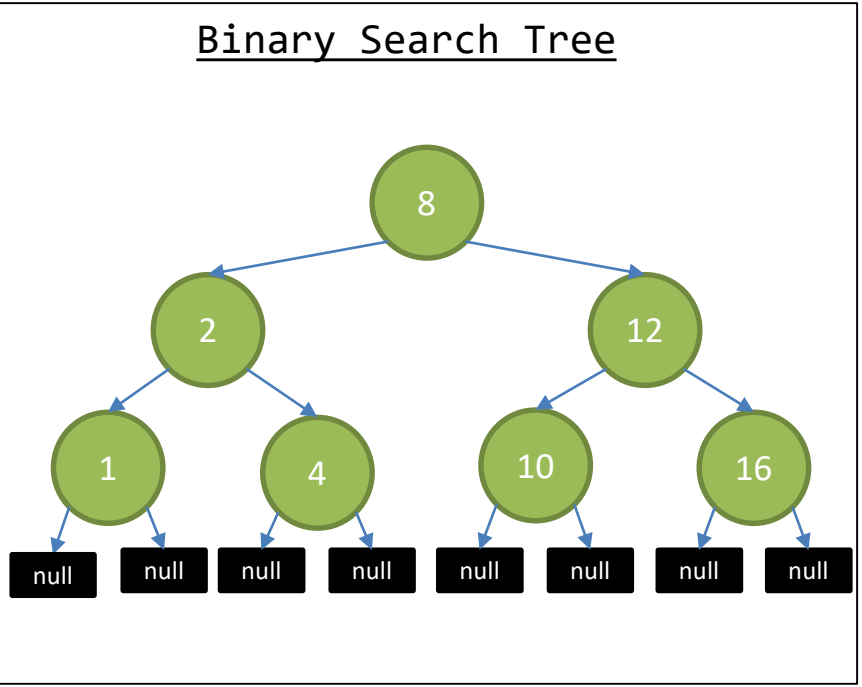
- Add
  - Construct a new Node
  - Start from the Root Node
  - Recursively go left and right until a leaf (NULL element) is found
  - Replace the leaf's reference with the newly created node's reference

Example "Adding 7"



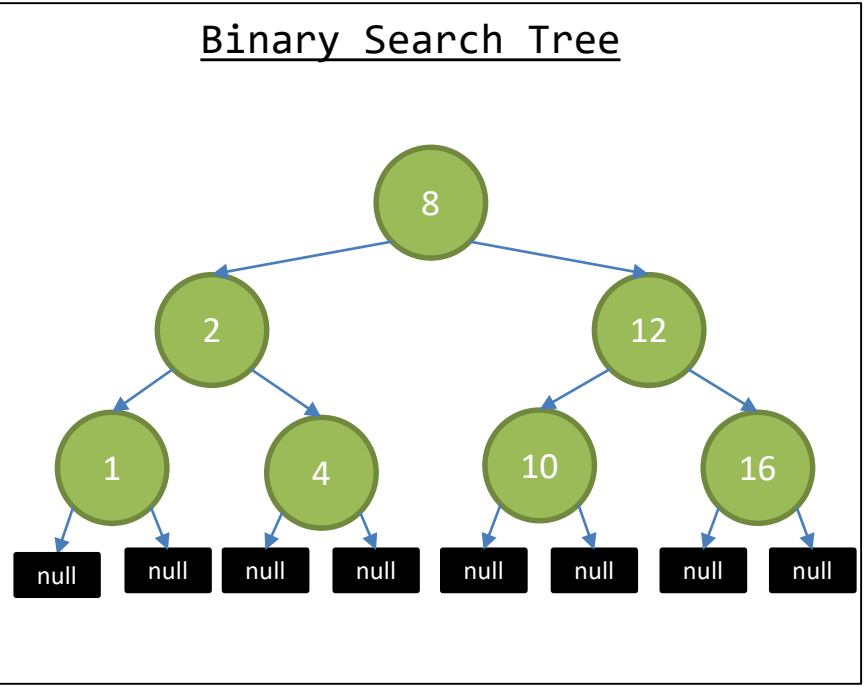
Binary Search Trees

- Traversals
  - Provide a way to traverse (“walk”) through the data structure
  - Not straight forward like linear structures
- 2 Versions
  - Depth-first (Stack Based)
  - Breadth-first (Queue Based)
- Depth-First
  - Travel deeper until a leaf is reached, then backtrack and do the same for a sibling node
- Breadth-First
  - Travel wider then move to the next level



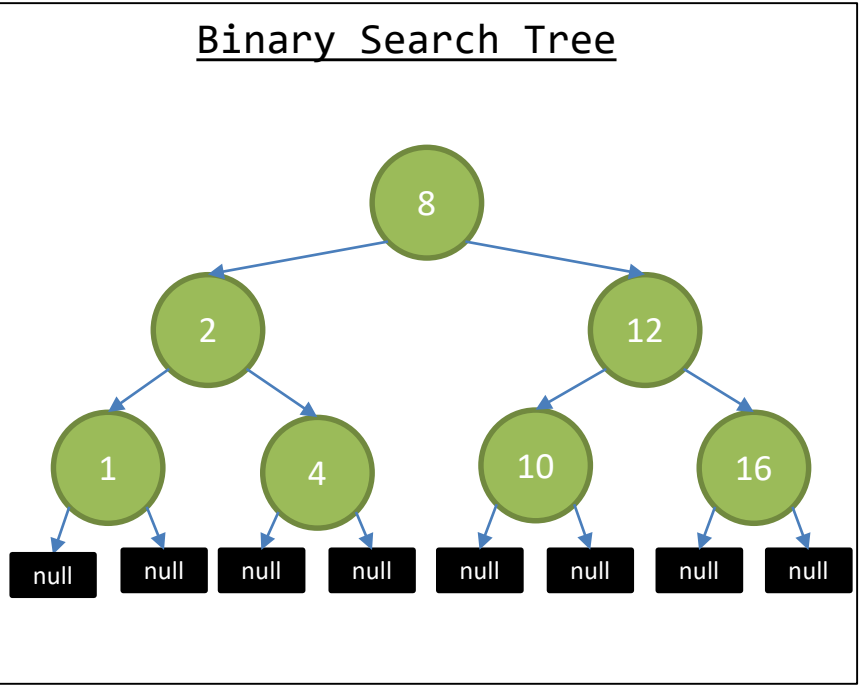
Binary Search Trees

- Depth-First Traversal Operators
  - LEFT: Traversal recursively down the left subtree
  - RIGHT: Traverse recursively down the right subree
  - PROCESS: Process the node such as printing a value or return true if they are equal
- Depth-First Traversals Types
  - Pre-Order
  - In-Order
  - Post-Order



Binary Search Trees

- Pre-Order
  - PROCESS, LEFT, RIGHT
  - “Pass by the Left”
- In-Order
  - LEFT, PROCESS, RIGHT
  - “Pass Underneath”
- Post-Order
  - LEFT, RIGHT, PROCESS
  - “Pass by the Right”



File Edit View Help

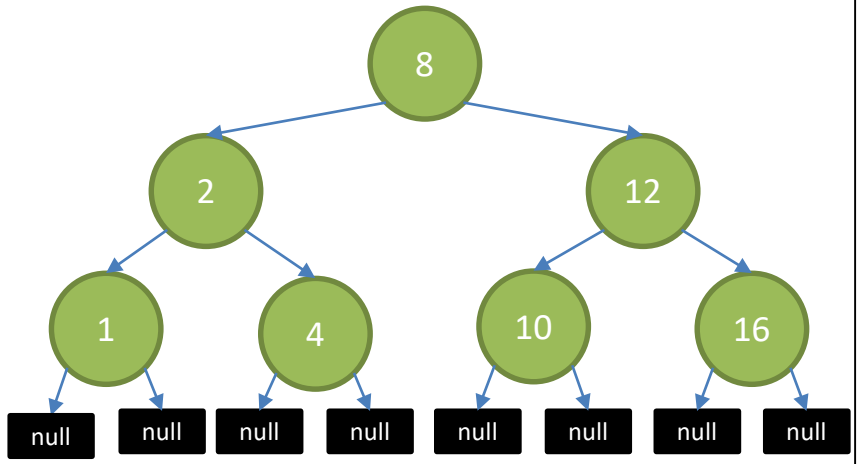
Binary Search Trees

Stack Example

Main Method

Call Stack

Binary Search Tree



Binary Search Trees

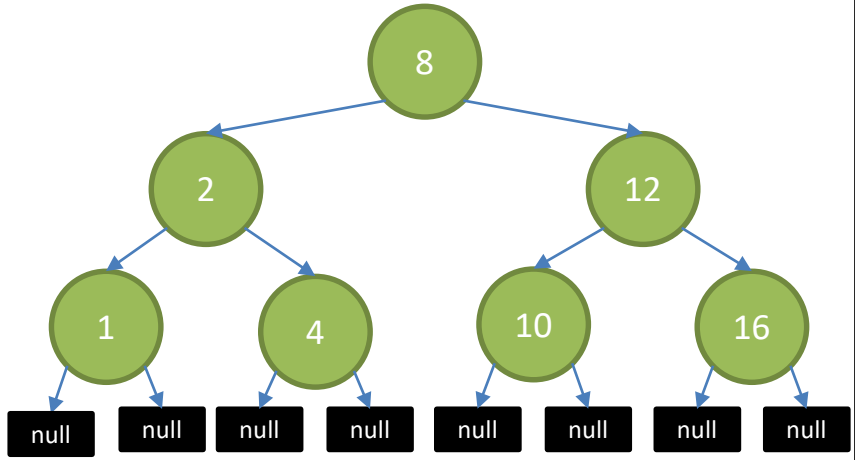
Stack Example

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

Main Method

Call Stack

Binary Search Tree

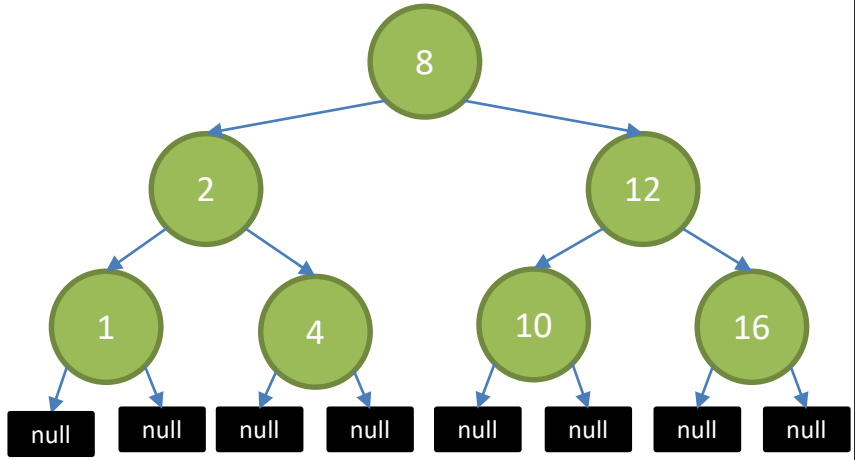


Binary Search Trees

Stack Example

printPreorder()
Main Method
Call Stack

Binary Search Tree



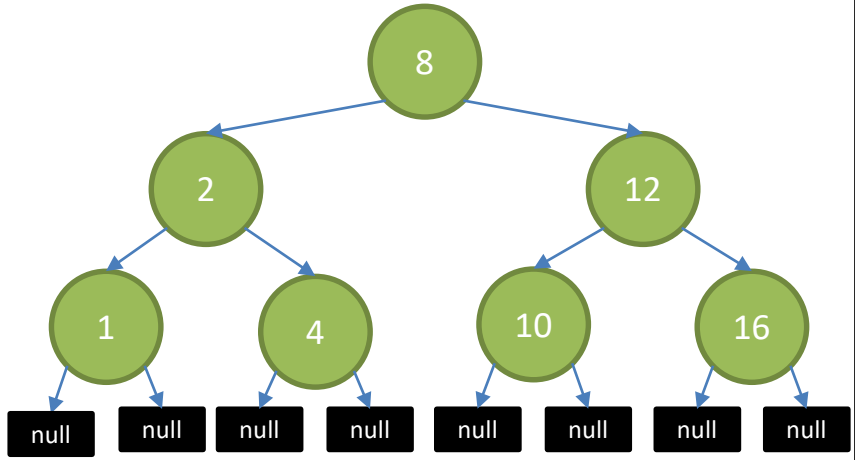
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



File Edit View Help

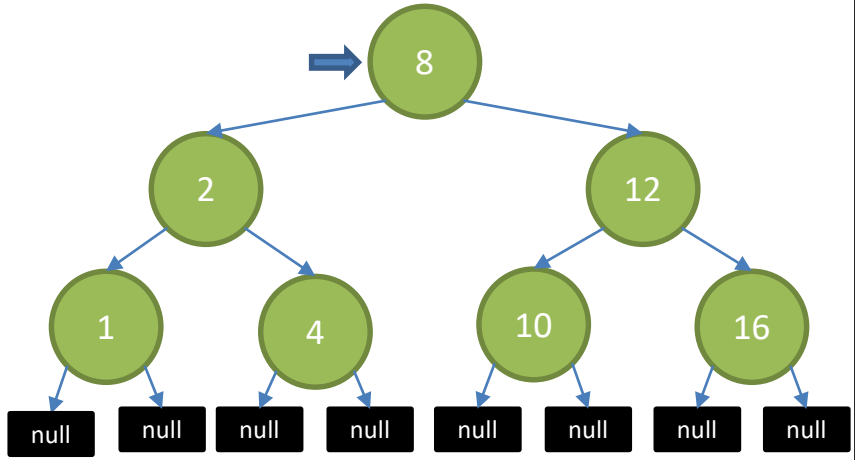
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



File Edit View Help

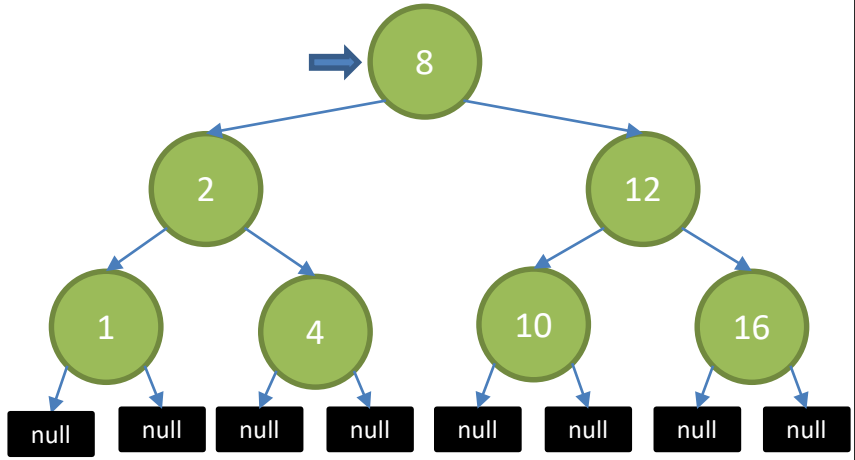
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



File Edit View Help

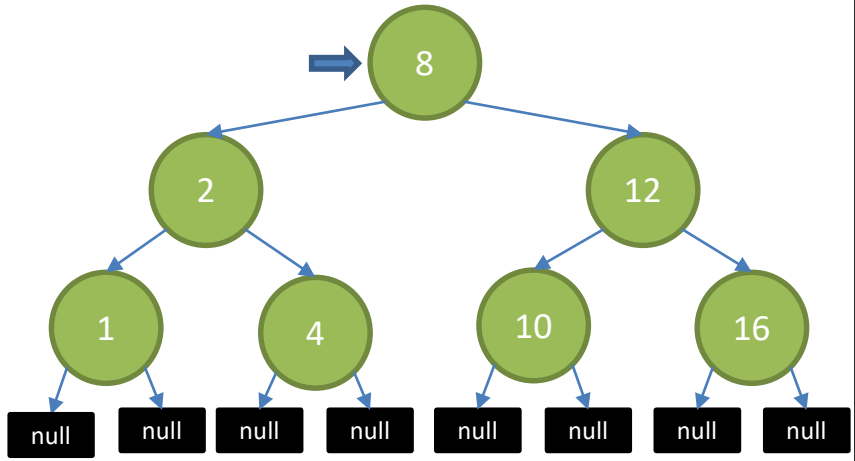
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8

File Edit View Help

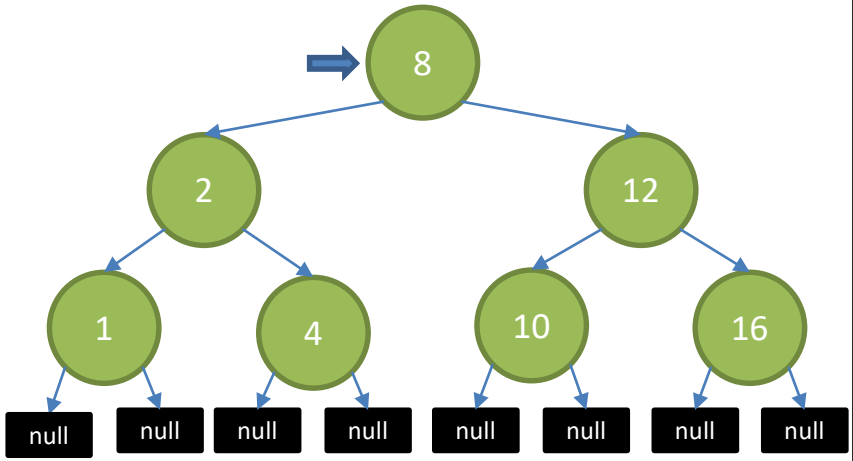
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8

File Edit View Help

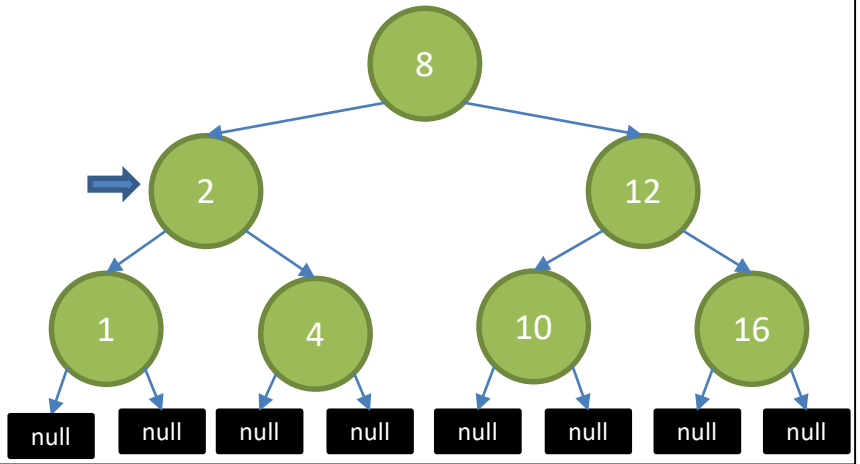
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8

File Edit View Help

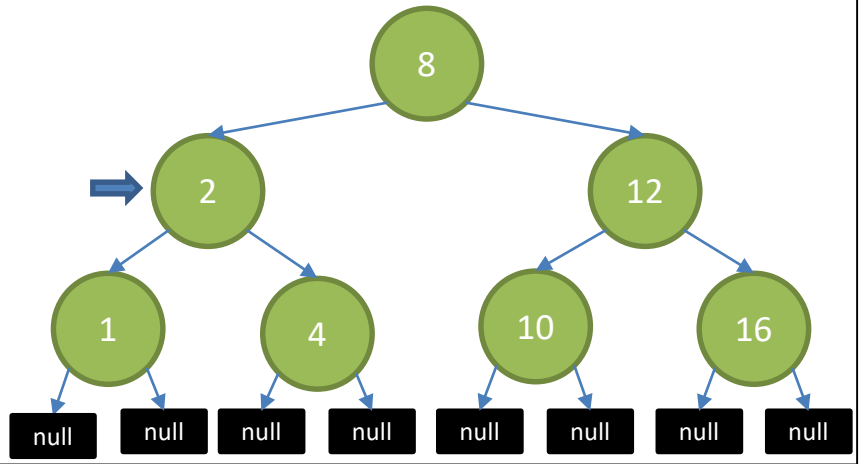
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8

File Edit View Help

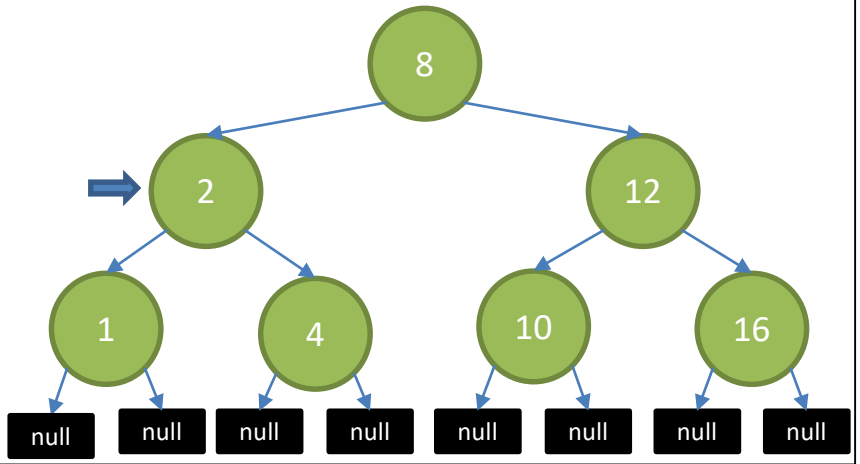
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2

File Edit View Help

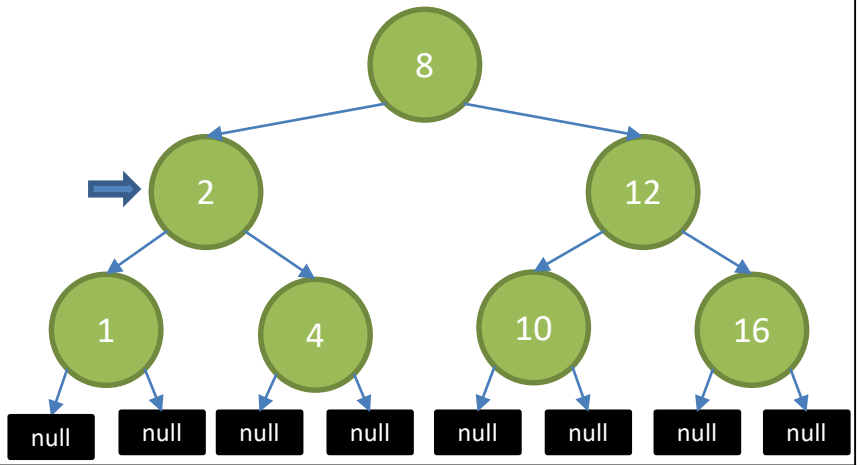
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2

File Edit View Help

Binary Search Trees

Stack Example

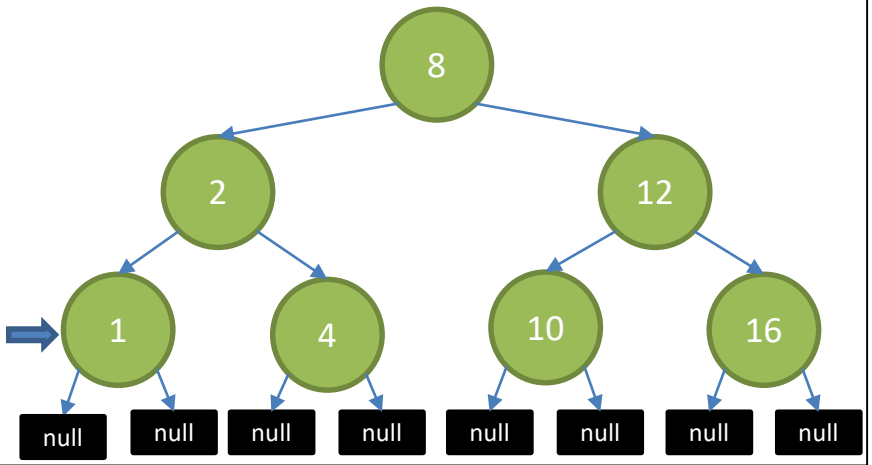
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2

File Edit View Help

Binary Search Trees

Stack Example

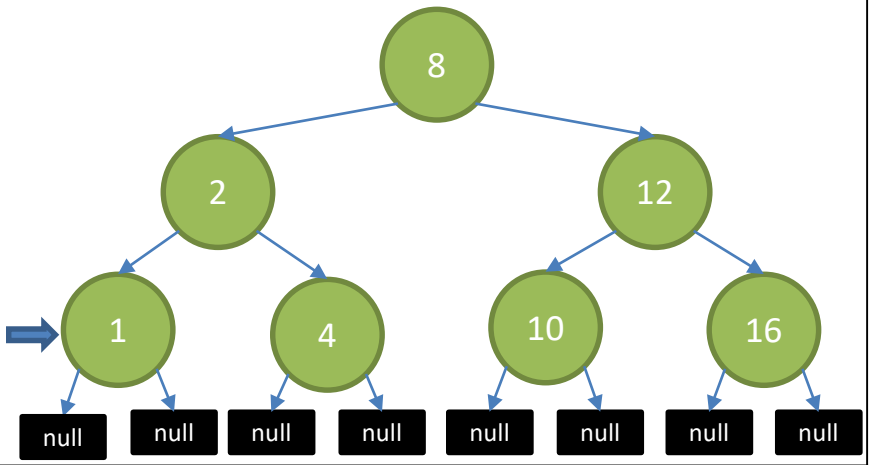
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2

File Edit View Help

Binary Search Trees

Stack Example

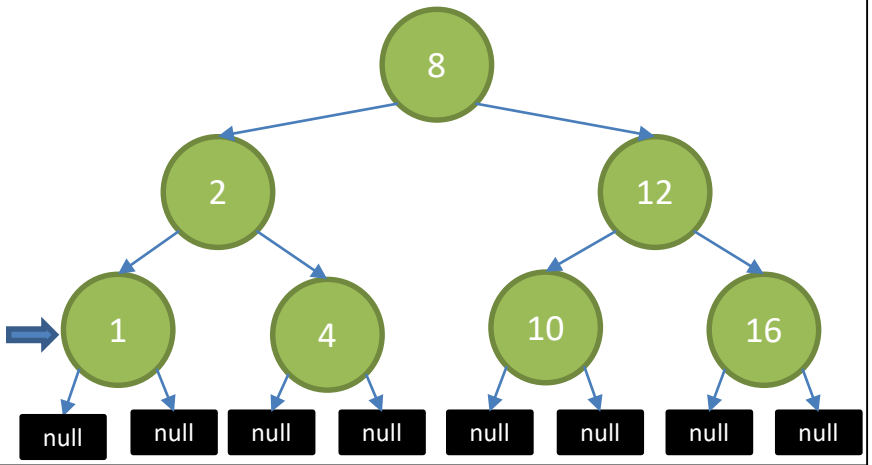
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

Stack Example

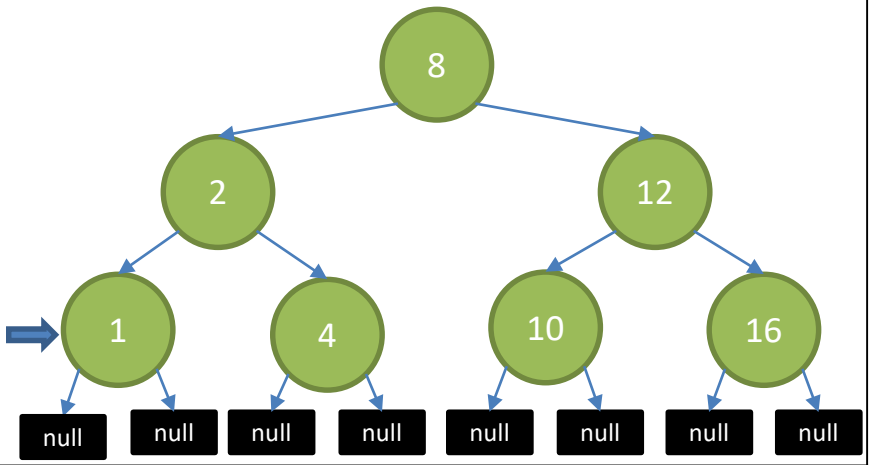
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

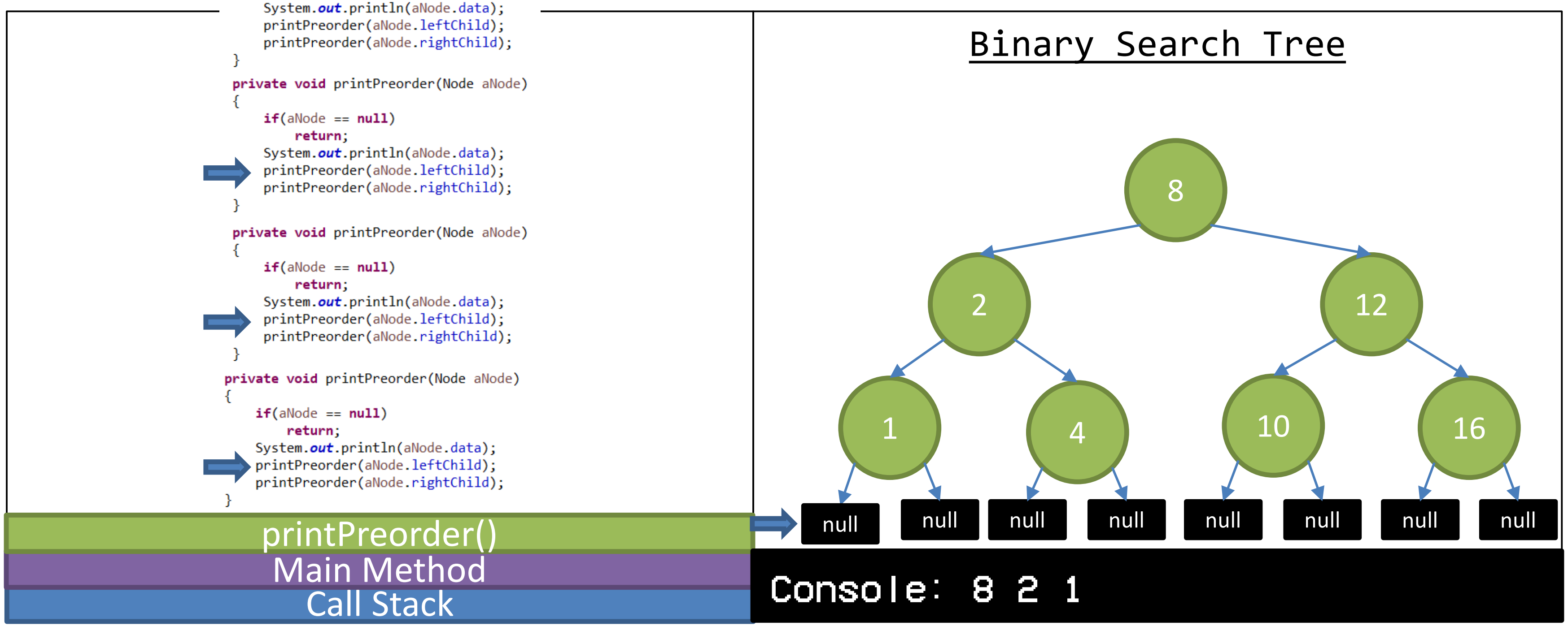
Binary Search Tree



Console: 8 2 1

Binary Search Trees

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```



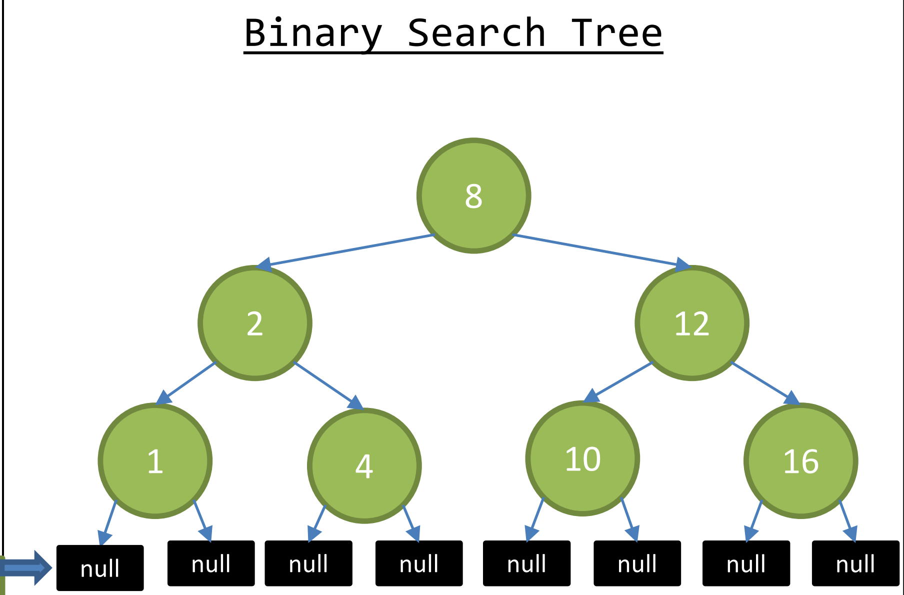
Binary Search Trees

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()

Main Method

Call Stack



Console: 8 2 1

File Edit View Help

Binary Search Trees

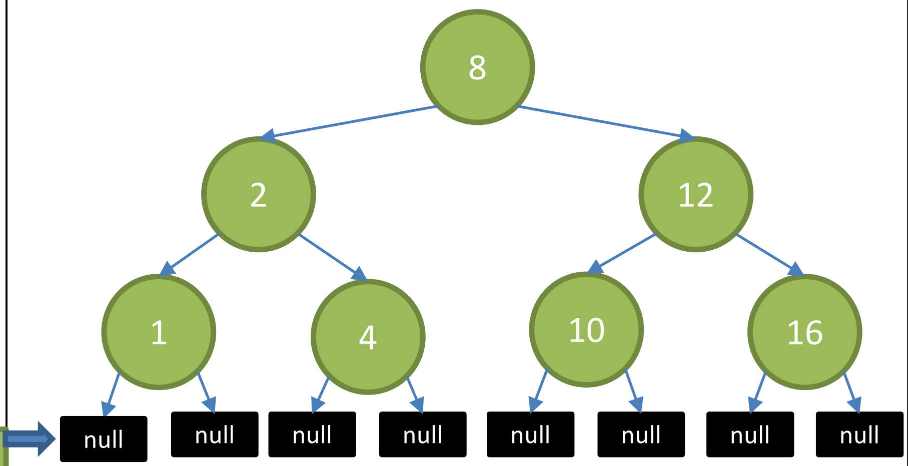
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()

Main Method

Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

Stack Example

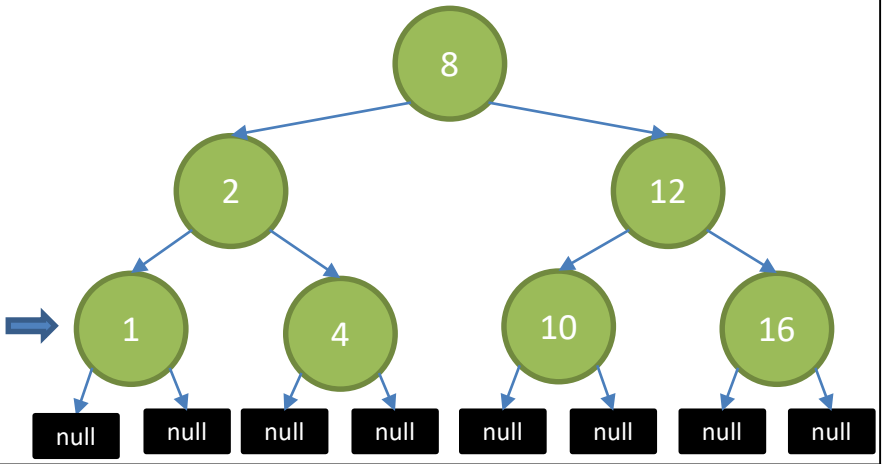
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

Stack Example

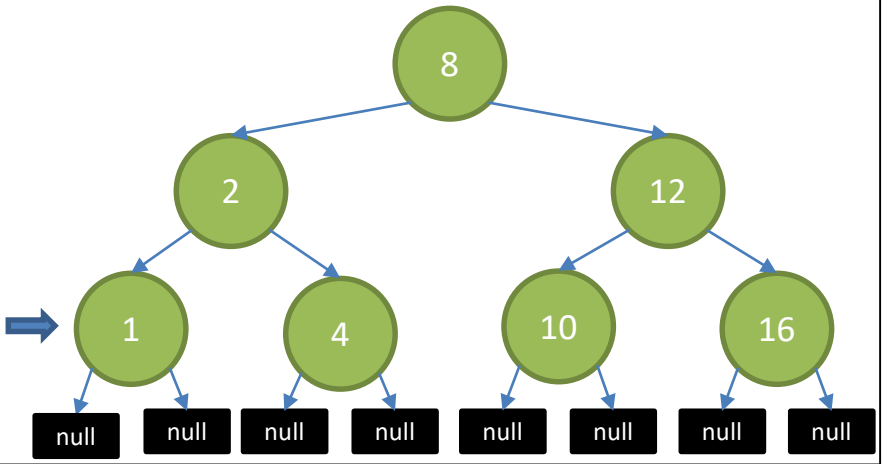
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

Stack Example

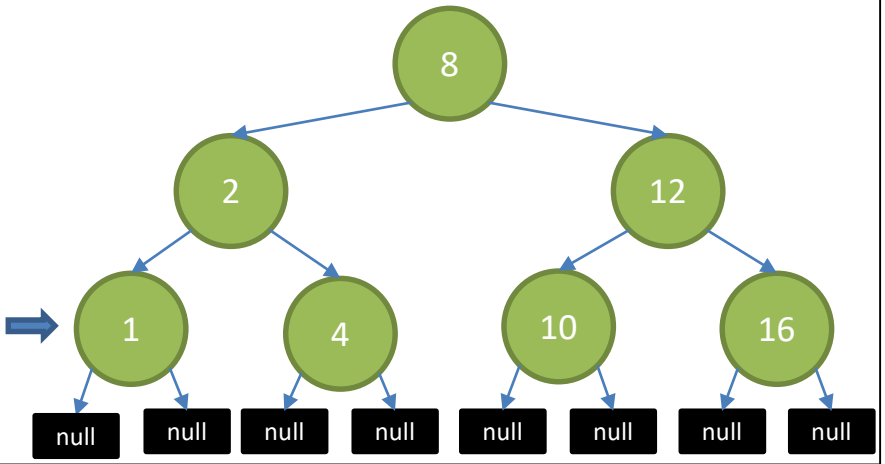
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

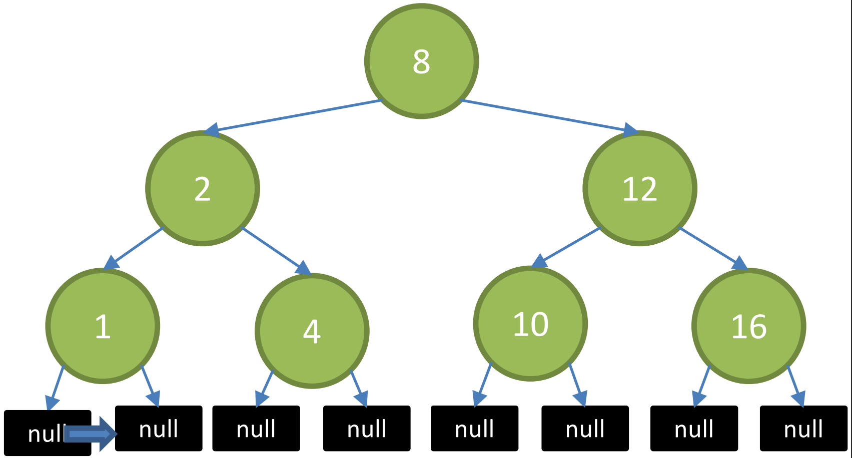
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()

Main Method

Call Stack

Binary Search Tree



Console: 8 2 1

Binary Search Trees

```java
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

```java
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

```java
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

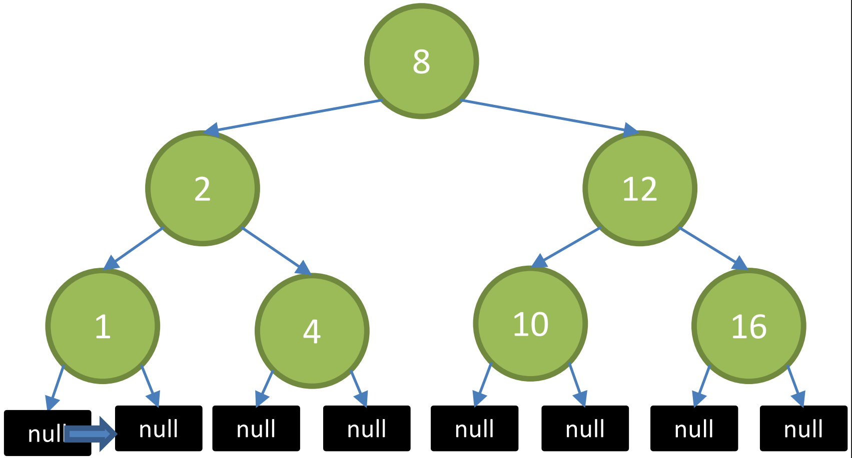
```java
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()

Main Method

Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

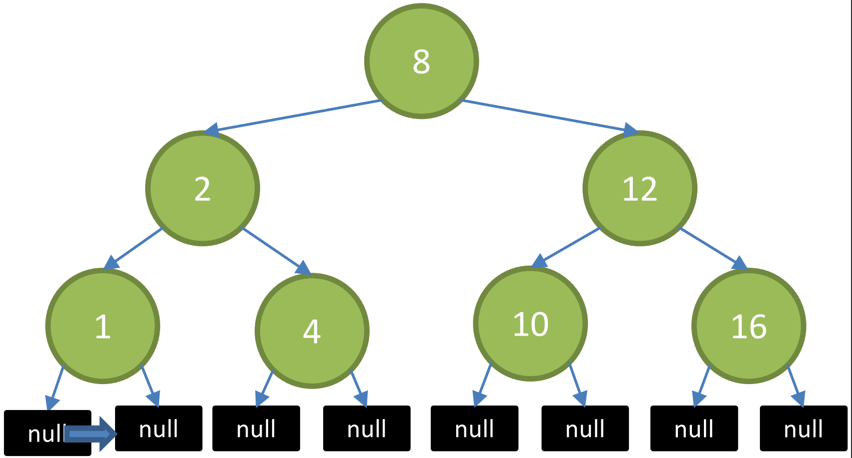
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()

Main Method

Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

Stack Example

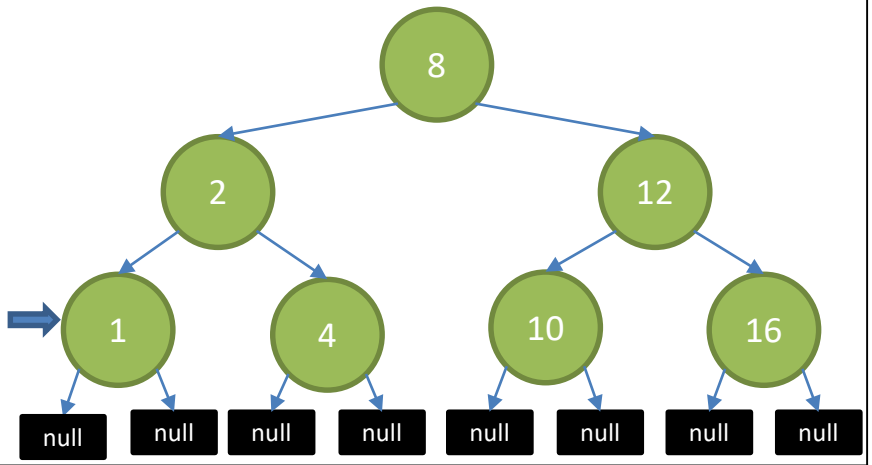
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

Stack Example

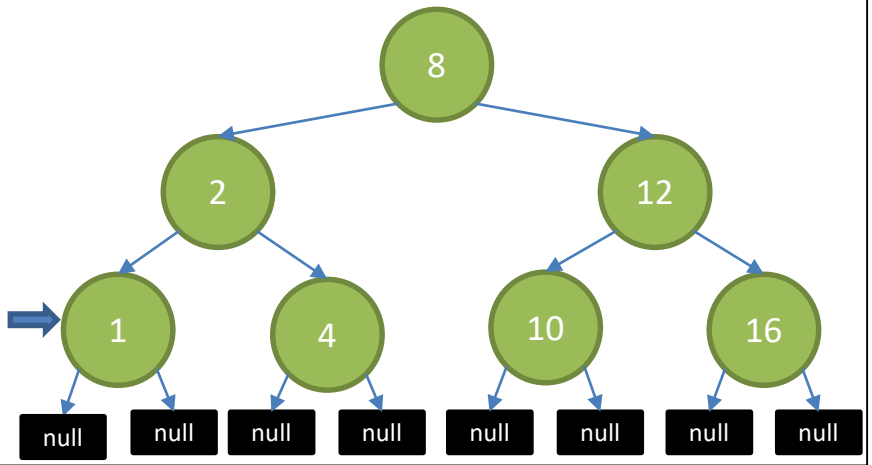
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

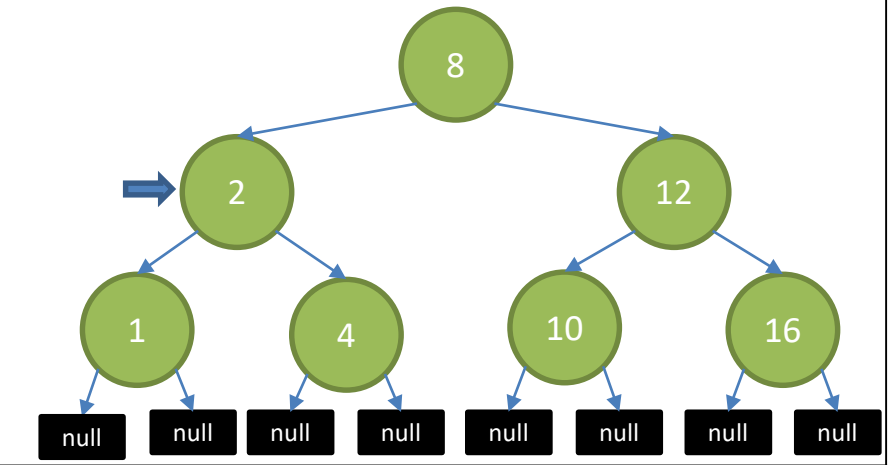
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

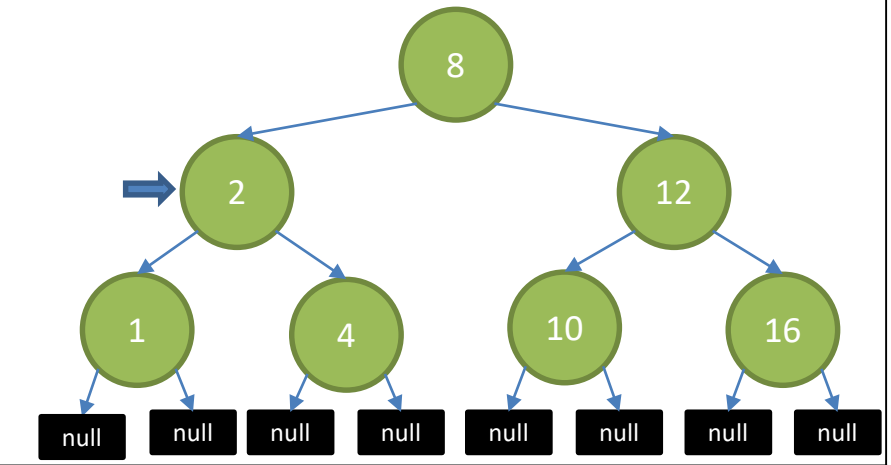
Binary Search Trees

Stack Example

```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

Binary Search Trees

Stack Example

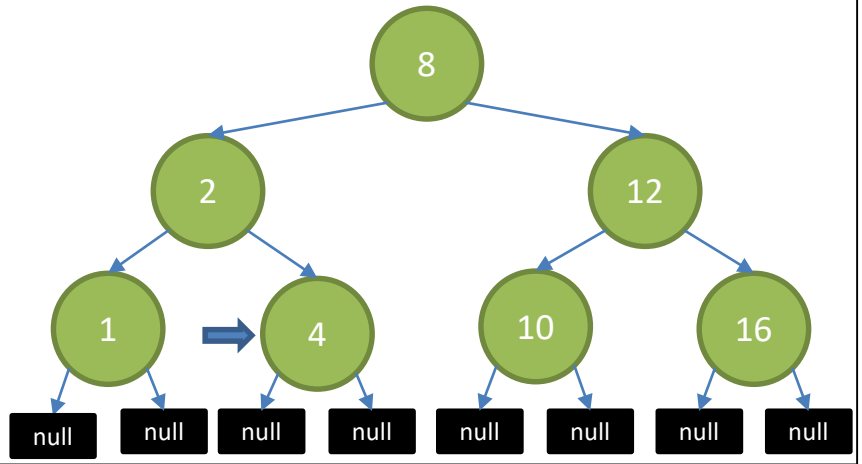
```
private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}

private void printPreorder(Node aNode)
{
    if(aNode == null)
        return;
    System.out.println(aNode.data);
    printPreorder(aNode.leftChild);
    printPreorder(aNode.rightChild);
}
```

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1

File Edit View Help

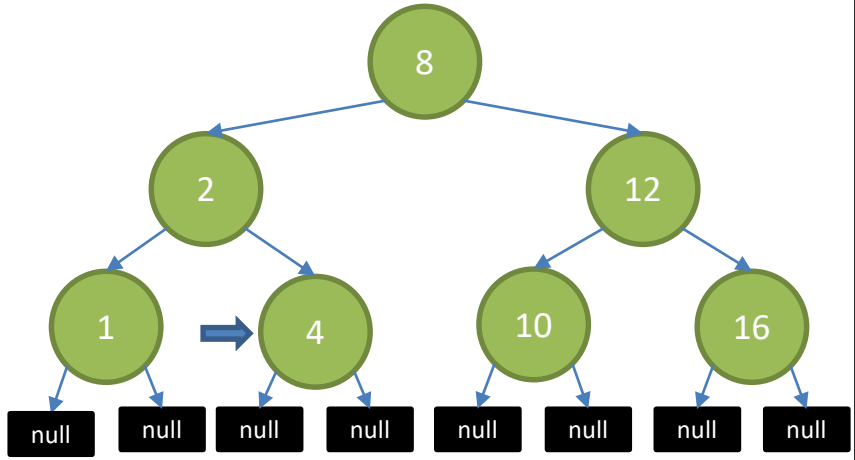
A Few Traversals Later

Binary Search Trees

Stack Example

printPreorder()
Main Method
Call Stack

Binary Search Tree



Console: 8 2 1 4 12 10 16

Binary Search Trees

- Breadth-First Traversal
  1. Add the Root node to a queue
  2. Dequeue a Node
  3. Process the Node's Value
  4. Enqueue the Node's Children
  5. Repeat Step 2 until the queue is empty