22. Trees :: Binary Search Trees :: *findNode*()

It is convenient when implementing *find*() and *add*() to have a helper method that will find a node in the BST containing a specific key. Our helper method will be called *findNode*() and it will search either the entire tree (if *root* is the root of the entire BST) or a subtree (when *root* is the root of a subtree) for the node containing *key*.

```
findNode(root: Node<K, E>, key: K): Node<K, E>
```

findNode() returns:

1. Null if *pRoot* is an empty tree, or
2. A reference to the node containing *key* if *key* was found, or
3. A reference to what would be the parent node of the node containing *key* if *key* were in the BST.

22. Trees :: Binary Search Trees :: *findNode*() Pseudocode

```
Method BST.findNode(In: root: BST.Node<K, E>; In: key: K) Returns BST.Node<K, E>
  -- If root is null, the tree is empty, so return null to indicate key is not found
  If root is null Then
    Return null
  -- If key equals the key in root then we have found the node containing key so
  -- return root.
  ElseIf key == root.key Then
    Return root
  -- If key is less than the key in root then if key is to be found, it will be found
  -- in the left subtree of root. However, if root does not have a left subtree, then
  -- we return root which would be the parent node of where key would be if it was in
  -- the tree.
  ElseIf key < root.key Then
    If root.leftChild is not null Then Return findNode(root.leftChild, key)
    Else Return root
  -- Otherwise key must be greater than the key in root so search the right subtree
  -- of root. However, if root does not have a right subtree, then we return root
  -- which would be the parent node of where key would be if it was in the tree.
  Else
    If root.rightChild is not null Then Return findNode(root.rightChild, key)
    Else Return root
  End If
End Method BST.findnode
```