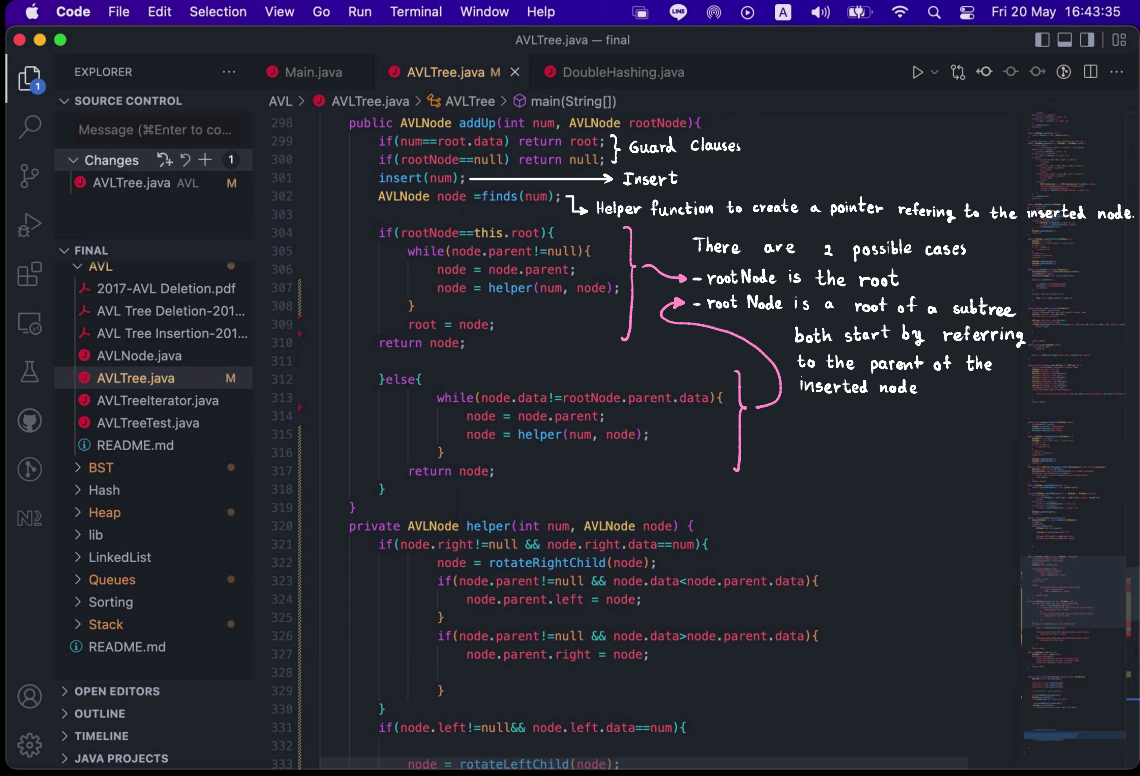
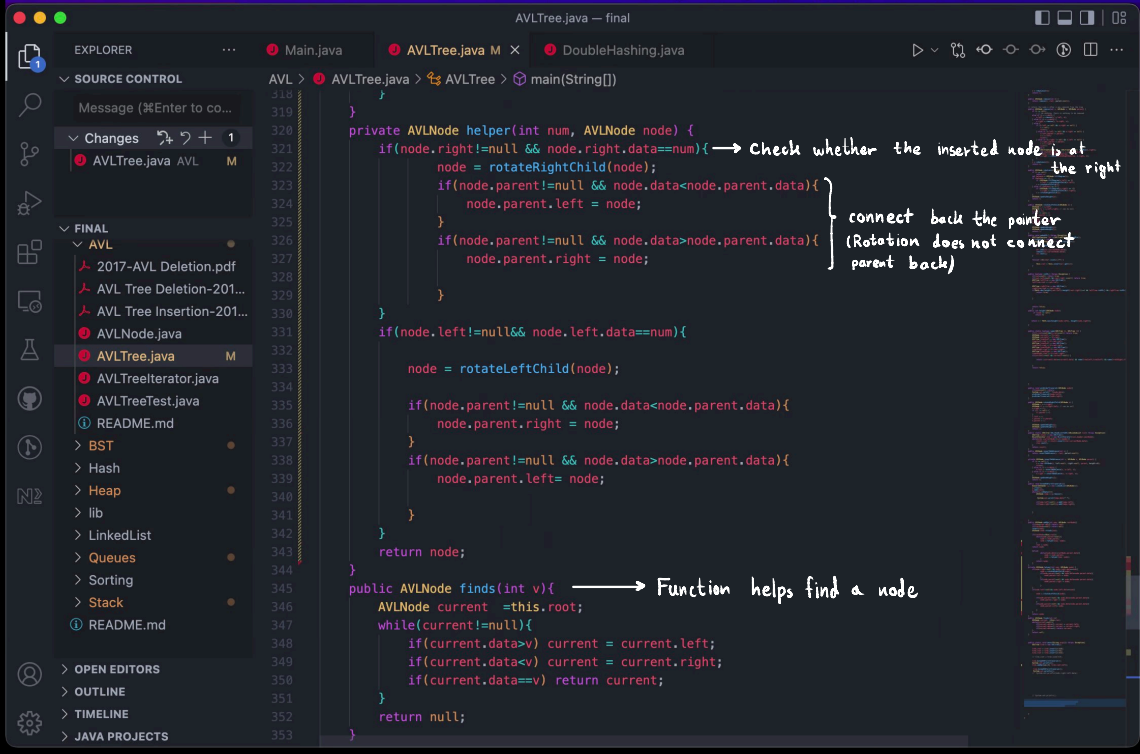
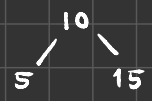

Code File Edit Selection View Go Run Terminal Window Help
Fri 20 May 16:43:35
AVLTree.java — final
EXPLORER
Main.java
AVLTree.java M
DoubleHashing.java
SOURCE CONTROL
Message (⌘Enter to co...
Changes 1
AVLTree.java AVL M
FINAL
AVL
2017-AVL Deletion.pdf
AVL Tree Deletion-201...
AVL Tree Insertion-201...
AVLNode.java
AVLTree.java M
AVLTreeIterator.java
AVLTreeTest.java
README.md
BST
Hash
Heap
lib
LinkedList
Queues
Sorting
Stack
README.md
OPEN EDITORS
OUTLINE
TIMELINE
JAVA PROJECTS
AVL > AVLTree.java > AVLTree > main(String[])
public AVLNode addUp(int num, AVLNode rootNode){
if(num==root.data) return root;
if(rootNode==null) return null;
Guard Clauses
insert(num);
Insert
AVLNode node =finds(num);
Helper function to create a pointer refering to the inserted node.
if(rootNode==this.root){
while(node.parent!=null){
node = node.parent;
node = helper(num, node);
}
root = node;
return node;
There are 2 possible cases
- rootNode is the root
- root Node is a root of a subtree
both start by referring to the parent of the inserted node
}else{
while(node.data!=rootNode.parent.data){
node = node.parent;
node = helper(num, node);
}
return node;
}
}
private AVLNode helper(int num, AVLNode node) {
if(node.right!=null && node.right.data==num){
node = rotateRightChild(node);
if(node.parent!=null && node.data<node.parent.data){
node.parent.left = node;
}
if(node.parent!=null && node.data>node.parent.data){
node.parent.right = node;
}
}
if(node.left!=null&& node.left.data==num){
node = rotateLeftChild(node);


Code File Edit Selection View Go Run Terminal Window Help
Fri 20 May 16:43:42
AVLTree.java — final
EXPLORER
Main.java
AVLTree.java M
DoubleHashing.java
SOURCE CONTROL
Message (⌘Enter to co...
Changes 1
AVLTree.java AVL M
FINAL
AVL
2017-AVL Deletion.pdf
AVL Tree Deletion-201...
AVL Tree Insertion-201...
AVLNode.java
AVLTree.java M
AVLTreeIterator.java
AVLTreeTest.java
README.md
BST
Hash
Heap
lib
LinkedList
Queues
Sorting
Stack
README.md
OPEN EDITORS
OUTLINE
TIMELINE
JAVA PROJECTS
AVL > AVLTree.java > AVLTree > main(String[])
}
}
private AVLNode helper(int num, AVLNode node) {
if(node.right!=null && node.right.data==num){
node = rotateRightChild(node);
if(node.parent!=null && node.data<node.parent.data){
node.parent.left = node;
}
if(node.parent!=null && node.data>node.parent.data){
node.parent.right = node;
}
}
if(node.left!=null&& node.left.data==num){
node = rotateLeftChild(node);
if(node.parent!=null && node.data<node.parent.data){
node.parent.right = node;
}
if(node.parent!=null && node.data>node.parent.data){
node.parent.left= node;
}
}
return node;
}
public AVLNode finds(int v){
AVLNode current =this.root;
while(current!=null){
if(current.data>v) current = current.left;
if(current.data<v) current = current.right;
if(current.data==v) return current;
}
return null;
}
Check whether the inserted node is at the right
connect back the pointer (Rotation does not connect parent back)
Function helps find a node

Example

Tree

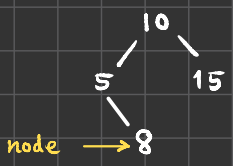


call addUp (8 , root)

```
public AVLNode addUp(int num, AVLNode rootNode){
    if(num==root.data) return root;
    if(rootNode==null) return null;
    insert(num);
    AVLNode node =finds(num);
```

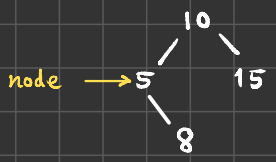
} Passes all Guard clause

A new AVLNode is inserted & node points to it.



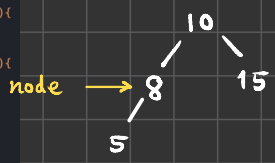

```
    if(rootNode==this.root){
        while(node.parent!=null){
            node = node.parent;
            node = helper(num, node);
        }
        root = node;
    return node;

    }else{
            while(node.data!=rootNode.parent.data){
                node = node.parent;
                node = helper(num, node);
            }
        return node;
    }
}
```

} root node passed into method
node now points to 5
& helper (8, node) is called




```
private AVLNode helper(int num, AVLNode node) {
    if(node.right!=null && node.right.data==num){
            node = rotateRightChild(node);
            if(node.parent!=null && node.data<node.parent.data){
                node.parent.left = node;
            }
            if(node.parent!=null && node.data>node.parent.data){
                node.parent.right = node;

            }
    }
    if(node.left!=null&& node.left.data==num){

        node = rotateLeftChild(node);

        if(node.parent!=null && node.data<node.parent.data){
            node.parent.right = node;
        }
        if(node.parent!=null && node.data>node.parent.data){
            node.parent.left= node;

        }
    }
    return node;
}
```

→ condition met & rotation is performed

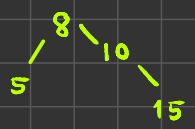


connection of node 10 & node 8 is made

but node.parent = 10 ≠ null so loop continues

helper (8, node) is called

↑ Refers to node 10



done.