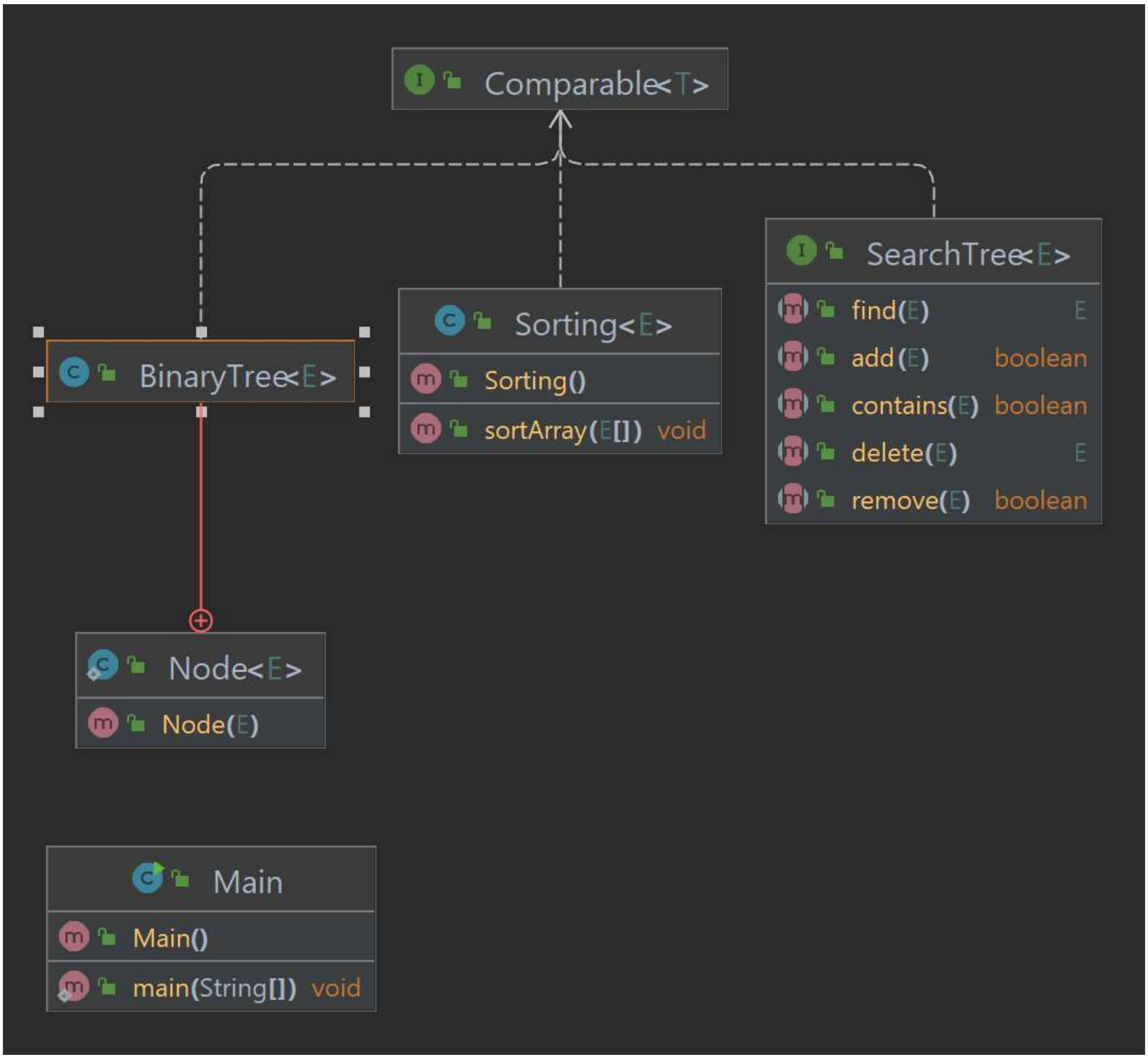
1. System Requirements

We need Operating system that have jdk 8 and jre 8 for use the program. We need some space to store objects of Stringbuilder, Binary Tree and strings/arrays. There is many integer array and string to perform test case.

Other thing we should consider is our resources like CPU, RAM, Stack size etc. Recursion method uses much more resources than loop method. So better CPU gives faster result.

2. Class Diagram



3. Problem Solution Approach

Firstly, I studied Binary Tree / Binary Search Tree to be able to do this homework. Then I implement Binary Tree class. After that, I started to think how should I convert Binary Tree to Binary Search Tree. I decided to do this operation in Binary Tree class (as a method).

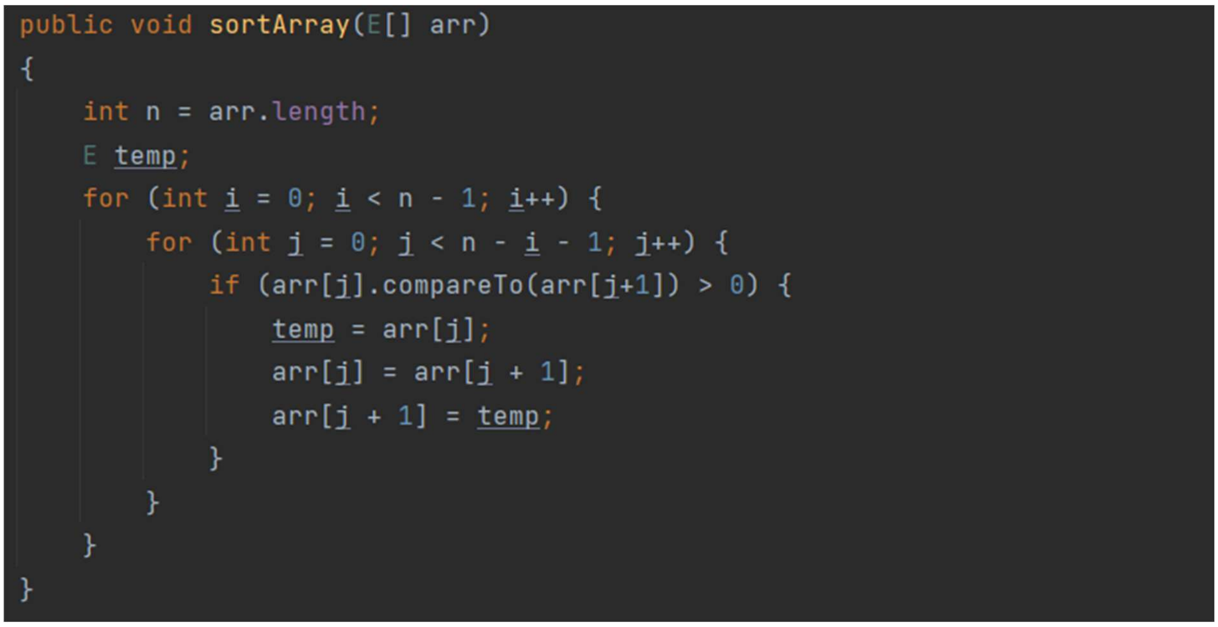
I create a n-size empty binary tree and n-size randomly generated array. Because of binary search tree should be sorted, I sort the array that I create and insert array items with inorder traverse method. This way works for converting binary tree to binary search tree.

4. Test Cases

- Check whether array’s size and binary tree’s size are equal.
- Check whether binary tree converted to binary search tree successfully.
- Check if print function works properly.
- Check if createNSizeBinaryTree works properly.
- Check if preOrderTraverse traverses tree properly.
- Check if sortArray method works properly.

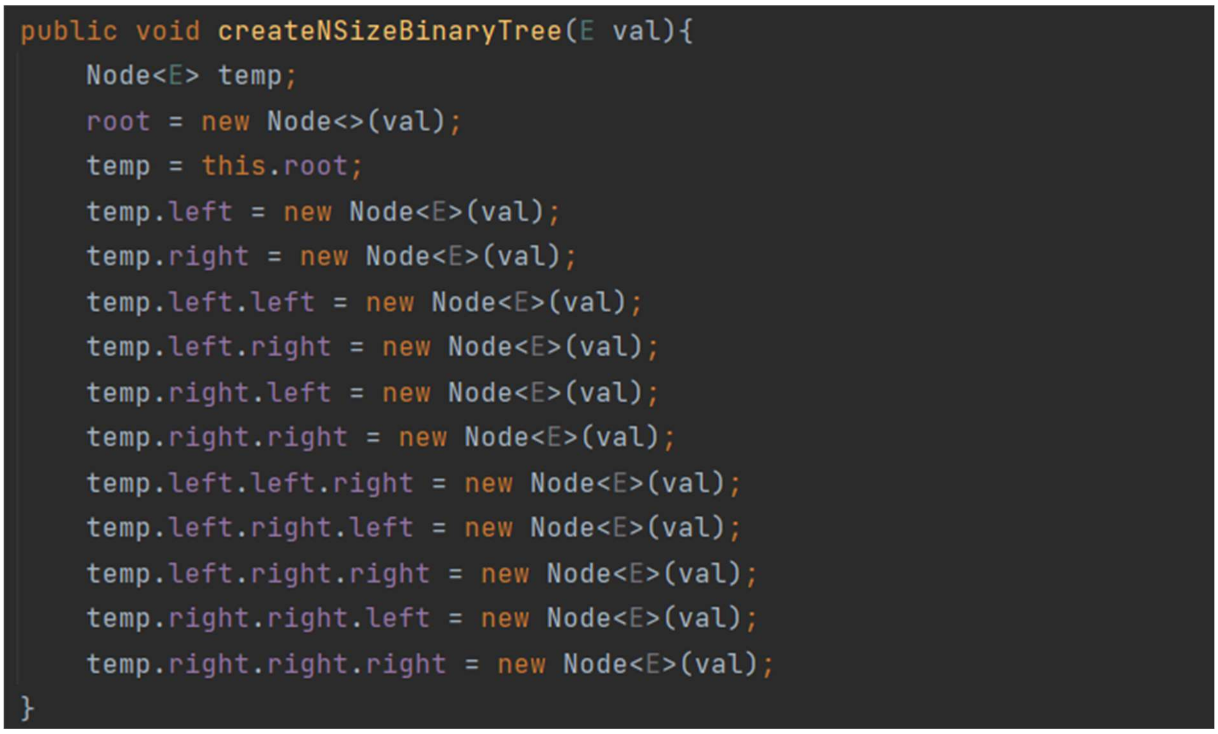
5. Time Complexities

Question-1:



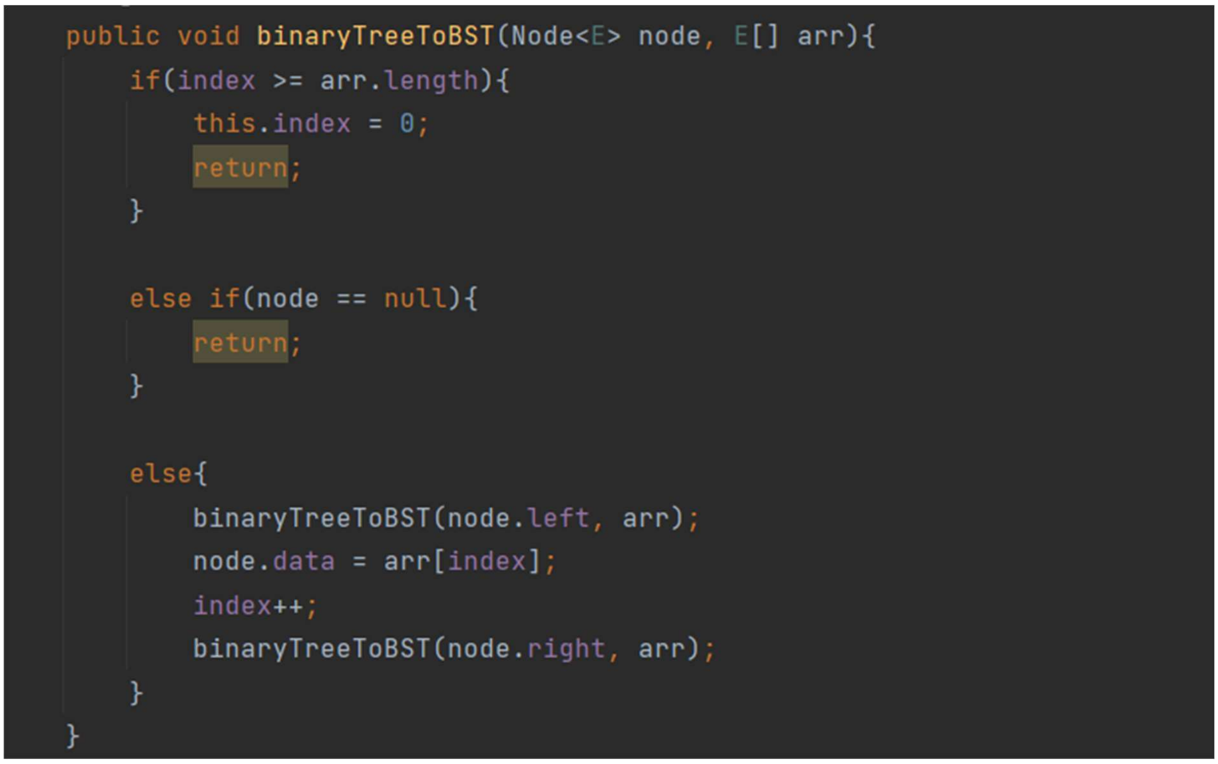
```java
public void sortArray(E[] arr)
{
    int n = arr.length;
    E temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j].compareTo(arr[j+1]) > 0) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

sortArray(E[] arr) ➔ T(n) = θ(n^2)



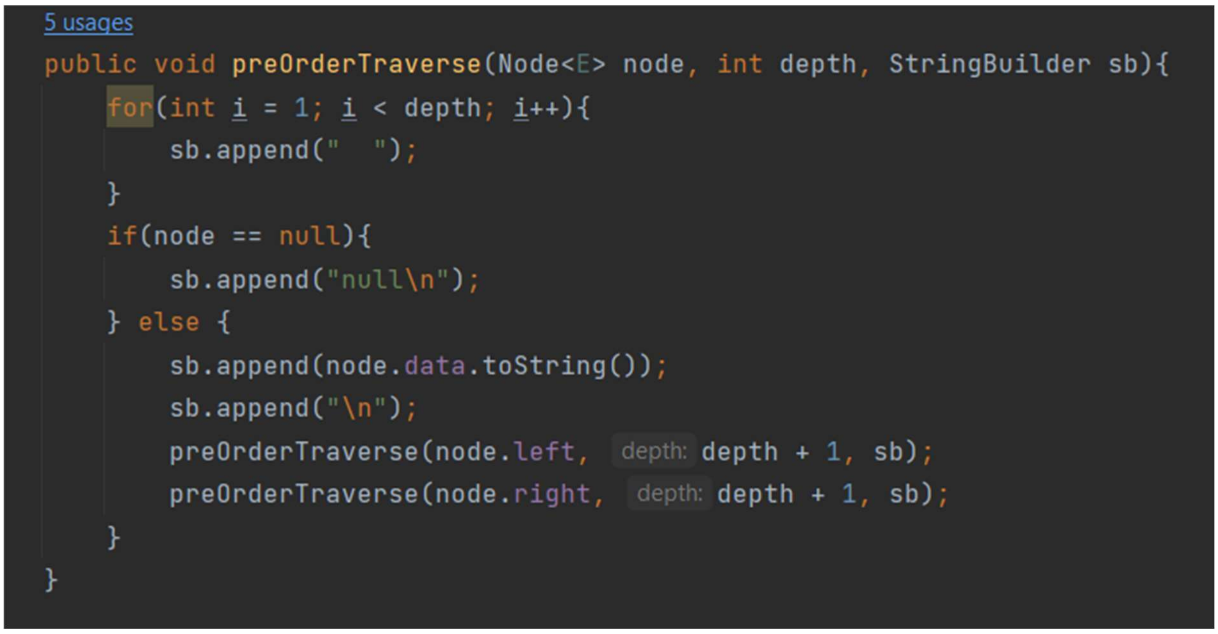
```java
public void createNSizeBinaryTree(E val){
    Node<E> temp;
    root = new Node<>(val);
    temp = this.root;
    temp.left = new Node<E>(val);
    temp.right = new Node<E>(val);
    temp.left.left = new Node<E>(val);
    temp.left.right = new Node<E>(val);
    temp.right.left = new Node<E>(val);
    temp.right.right = new Node<E>(val);
    temp.left.left.right = new Node<E>(val);
    temp.left.right.left = new Node<E>(val);
    temp.left.right.right = new Node<E>(val);
    temp.right.right.left = new Node<E>(val);
    temp.right.right.right = new Node<E>(val);
}
```

createNSizeBinaryTree(E val) ➔ T(n) = θ(1)



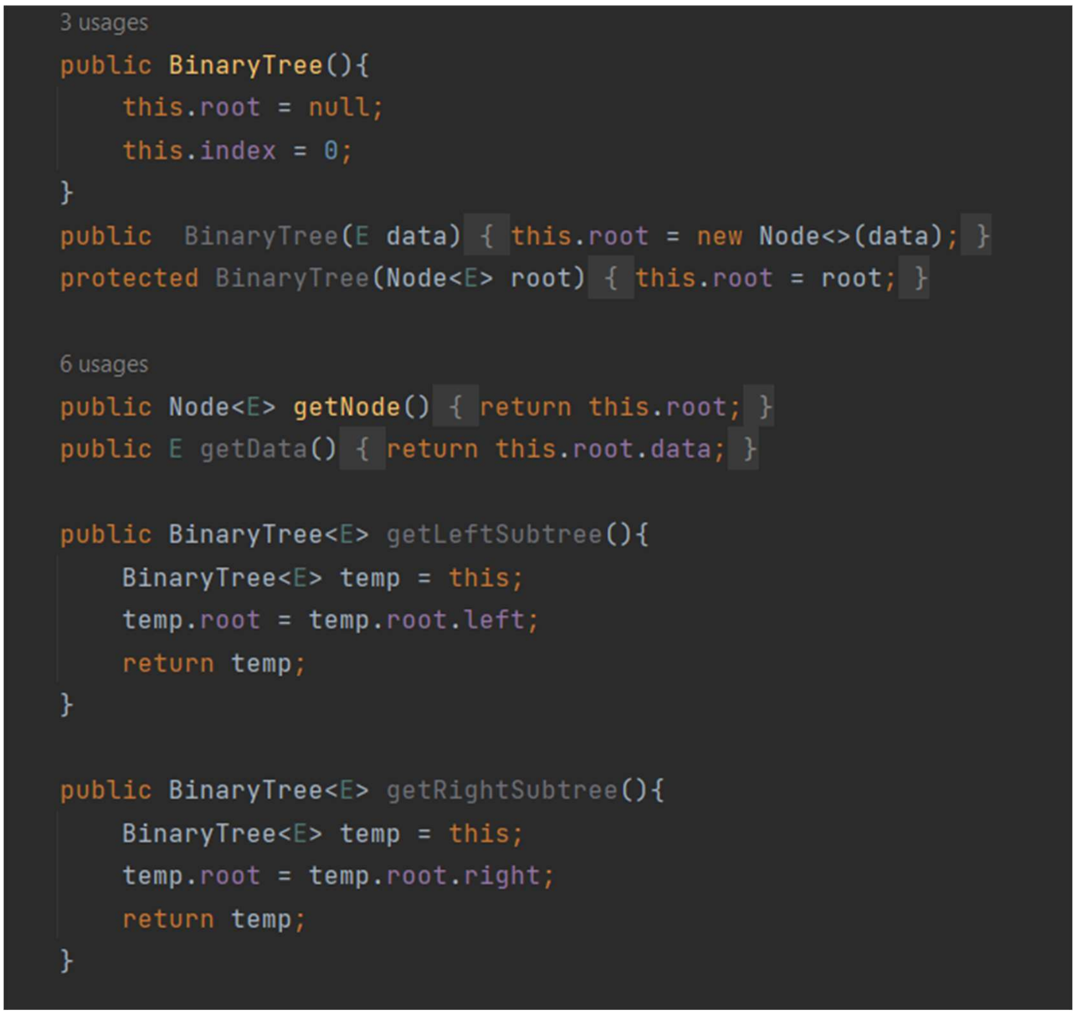
```java
    public void binaryTreeToBST(Node<E> node, E[] arr){
        if(index >= arr.length){
            this.index = 0;
            return;
        }

        else if(node == null){
            return;
        }

        else{
            binaryTreeToBST(node.left, arr);
            node.data = arr[index];
            index++;
            binaryTreeToBST(node.right, arr);
        }
    }
```

binaryTreeToBST ➔ T(n) = O(nlogn) on average



```
5 usages
public void preOrderTraverse(Node<E> node, int depth, StringBuilder sb){
    for(int i = 1; i < depth; i++){
        sb.append("  ");
    }
    if(node == null){
        sb.append("null\n");
    } else {
        sb.append(node.data.toString());
        sb.append("\n");
        preOrderTraverse(node.left, depth: depth + 1, sb);
        preOrderTraverse(node.right, depth: depth + 1, sb);
    }
}
```

preOrderTraverse(Node<E> node, int depth, StringBuilder sb)
➔ T(n) = O(n)



```
3 usages
public BinaryTree(){
    this.root = null;
    this.index = 0;
}
public  BinaryTree(E data) { this.root = new Node<>(data); }
protected BinaryTree(Node<E> root) { this.root = root; }

6 usages
public Node<E> getNode() { return this.root; }
public E getData() { return this.root.data; }

public BinaryTree<E> getLeftSubtree(){
    BinaryTree<E> temp = this;
    temp.root = temp.root.left;
    return temp;
}

public BinaryTree<E> getRightSubtree(){
    BinaryTree<E> temp = this;
    temp.root = temp.root.right;
    return temp;
}
```

All getters/constructors ➔ θ(1)

6. Running Command and Results

The file should be unzipped and opened as a Project in IntellijIdea. All the methods are accessible on driver code.