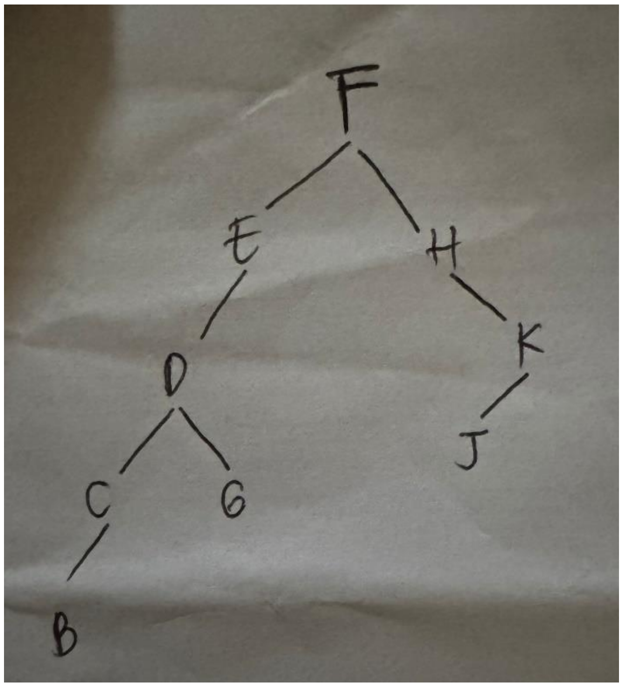
Nama : Dini Auliya Fauziah

Kelas : 47-02

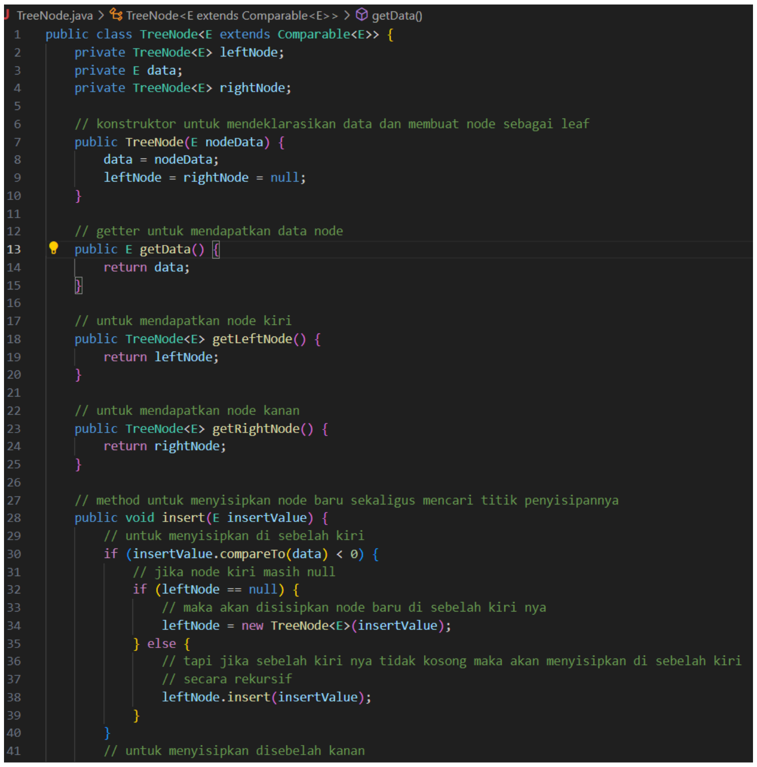
Nim : 607062300015

Link Github : https://github.com/diniauliyaf/Jurnal02_ISD_Dini-Auliya-Fauziah.git

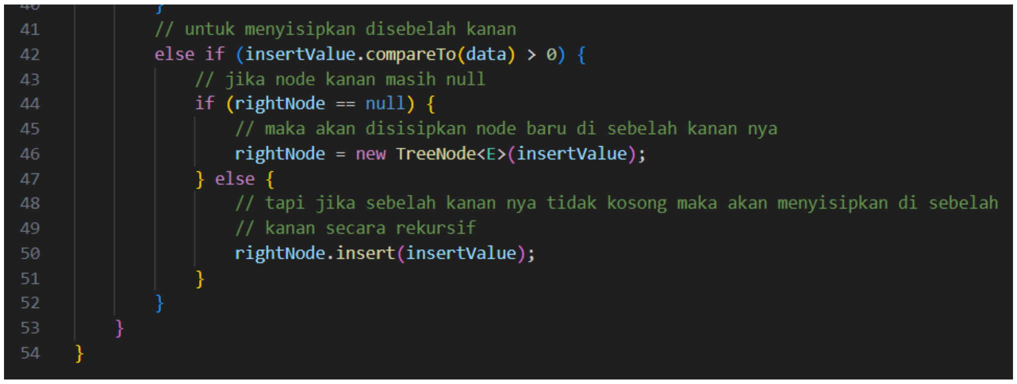
JURNAL 13



Cara Pengerjaan :

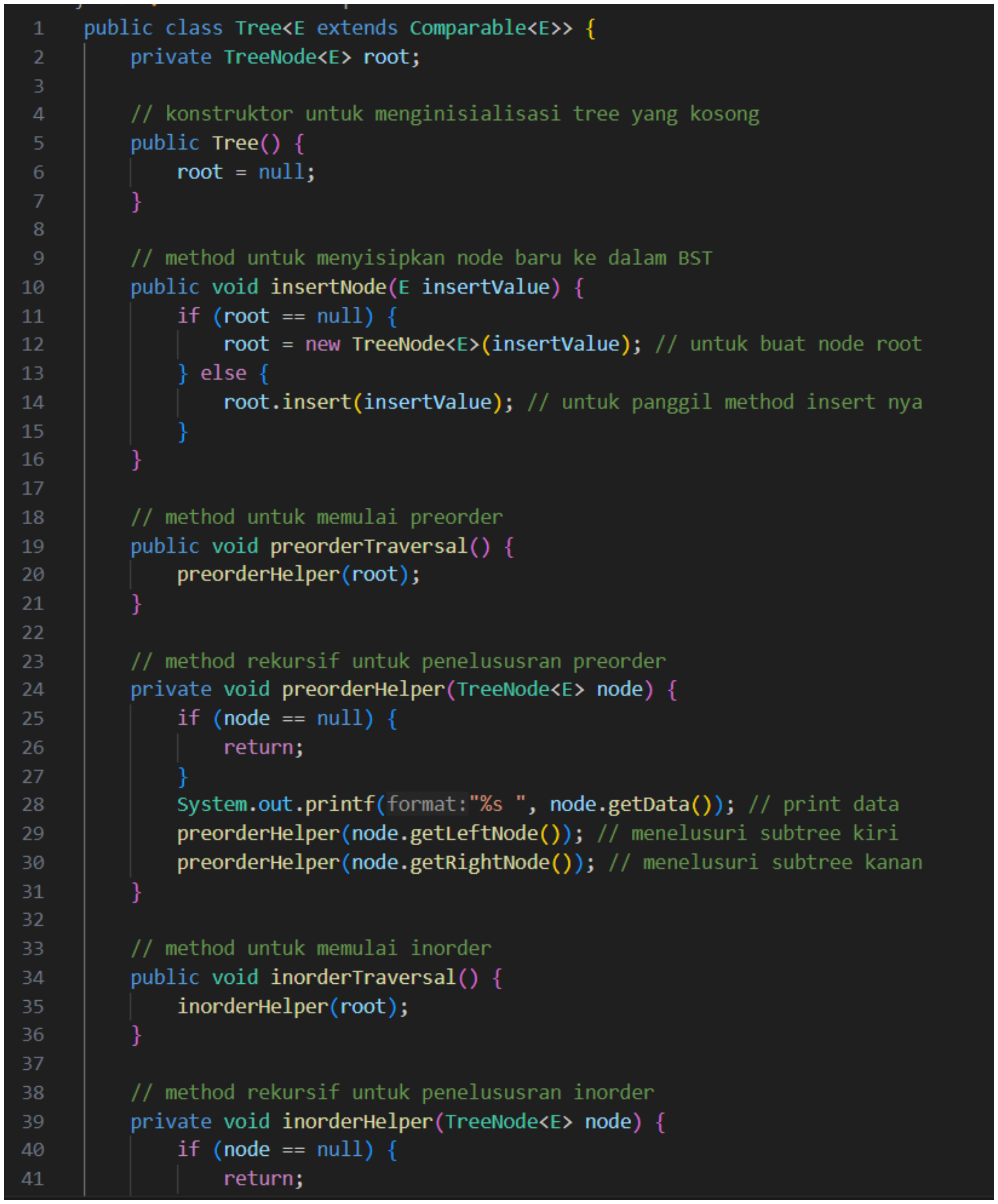

TreeNode.java > TreeNode<E extends Comparable<E>> > getData()

```java
public class TreeNode<E extends Comparable<E>> {
    private TreeNode<E> leftNode;
    private E data;
    private TreeNode<E> rightNode;

    // konstruktor untuk mendeklarasikan data dan membuat node sebagai leaf
    public TreeNode(E nodeData) {
        data = nodeData;
        leftNode = rightNode = null;
    }

    // getter untuk mendapatkan data node
    public E getData() {
        return data;
    }

    // untuk mendapatkan node kiri
    public TreeNode<E> getLeftNode() {
        return leftNode;
    }

    // untuk mendapatkan node kanan
    public TreeNode<E> getRightNode() {
        return rightNode;
    }

    // method untuk menyisipkan node baru sekaligus mencari titik penyisipannya
    public void insert(E insertValue) {
        // untuk menyisipkan di sebelah kiri
        if (insertValue.compareTo(data) < 0) {
            // jika node kiri masih null
            if (leftNode == null) {
                // maka akan disisipkan node baru di sebelah kiri nya
                leftNode = new TreeNode<E>(insertValue);
            } else {
                // tapi jika sebelah kiri nya tidak kosong maka akan menyisipkan di sebelah kiri
                // secara rekursif
                leftNode.insert(insertValue);
            }
        }
        // untuk menyisipkan disebelah kanan
```

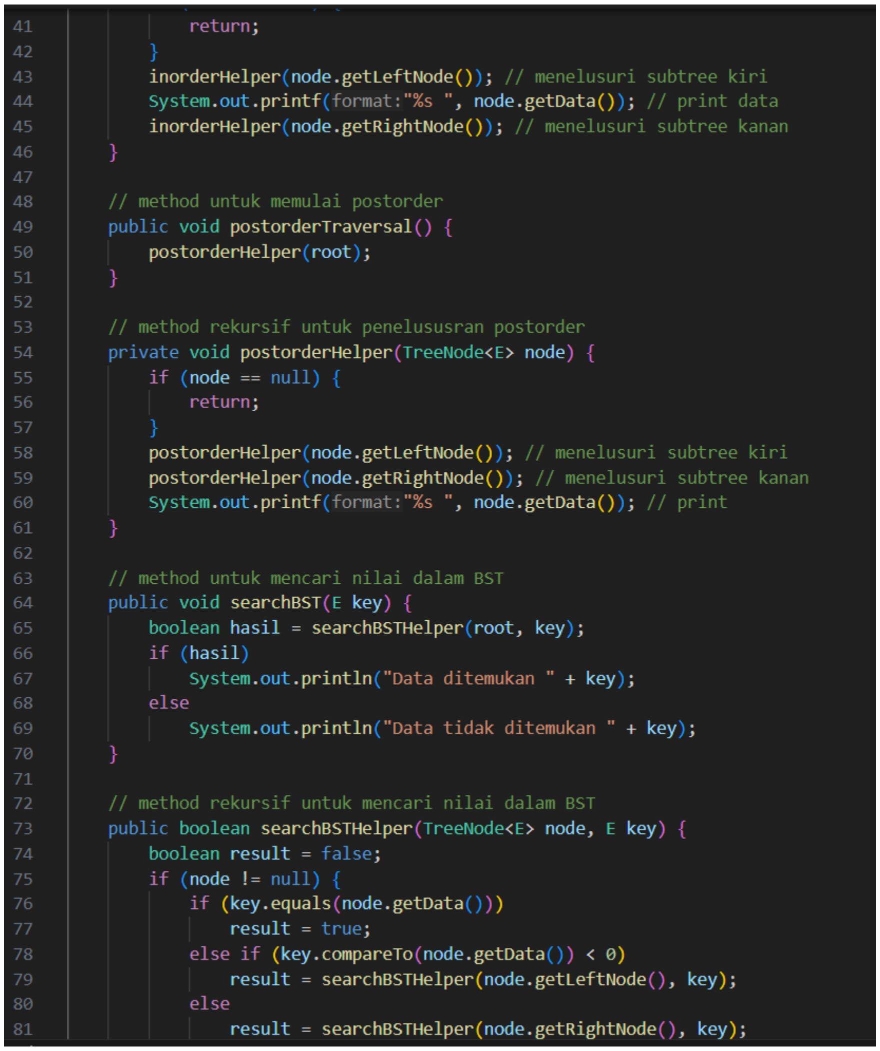


```java
        // untuk menyisipkan disebelah kanan
        else if (insertValue.compareTo(data) > 0) {
            // jika node kanan masih null
            if (rightNode == null) {
                // maka akan disisipkan node baru di sebelah kanan nya
                rightNode = new TreeNode<E>(insertValue);
            } else {
                // tapi jika sebelah kanan nya tidak kosong maka akan menyisipkan di sebelah
                // kanan secara rekursif
                rightNode.insert(insertValue);
            }
        }
    }
}
```

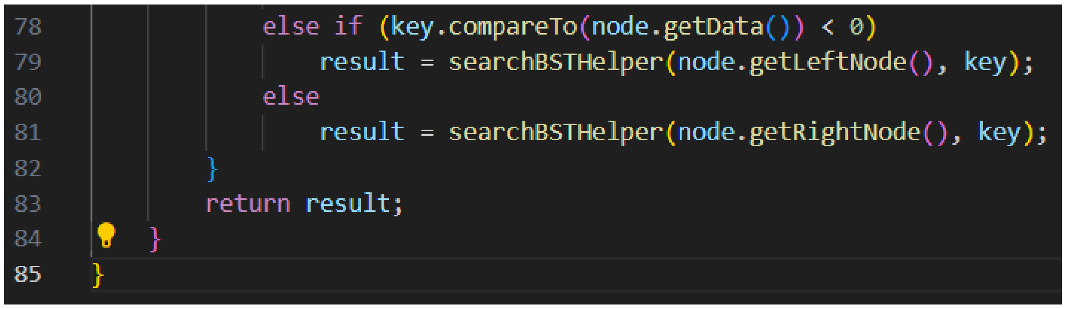
Tree :



```java
public class Tree<E extends Comparable<E>> {
    private TreeNode<E> root;

    // konstruktor untuk menginisialisasi tree yang kosong
    public Tree() {
        root = null;
    }

    // method untuk menyisipkan node baru ke dalam BST
    public void insertNode(E insertValue) {
        if (root == null) {
            root = new TreeNode<E>(insertValue); // untuk buat node root
        } else {
            root.insert(insertValue); // untuk panggil method insert nya
        }
    }

    // method untuk memulai preorder
    public void preorderTraversal() {
        preorderHelper(root);
    }

    // method rekursif untuk penelususran preorder
    private void preorderHelper(TreeNode<E> node) {
        if (node == null) {
            return;
        }
        System.out.printf(format:"%s ", node.getData()); // print data
        preorderHelper(node.getLeftNode()); // menelusuri subtree kiri
        preorderHelper(node.getRightNode()); // menelusuri subtree kanan
    }

    // method untuk memulai inorder
    public void inorderTraversal() {
        inorderHelper(root);
    }

    // method rekursif untuk penelususran inorder
    private void inorderHelper(TreeNode<E> node) {
        if (node == null) {
            return;
```



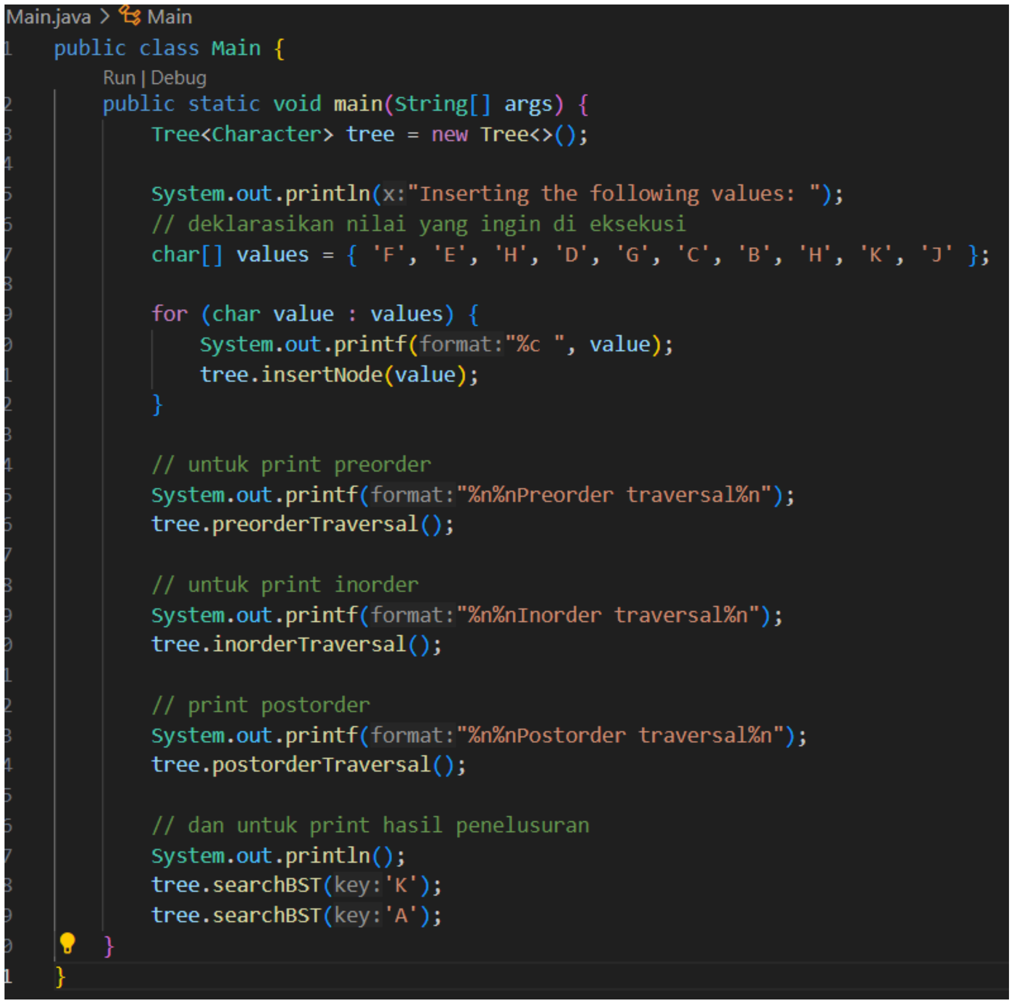
```java
            return;
        }
        inorderHelper(node.getLeftNode()); // menelusuri subtree kiri
        System.out.printf(format:"%s ", node.getData()); // print data
        inorderHelper(node.getRightNode()); // menelusuri subtree kanan
    }

    // method untuk memulai postorder
    public void postorderTraversal() {
        postorderHelper(root);
    }

    // method rekursif untuk penelususran postorder
    private void postorderHelper(TreeNode<E> node) {
        if (node == null) {
            return;
        }
        postorderHelper(node.getLeftNode()); // menelusuri subtree kiri
        postorderHelper(node.getRightNode()); // menelusuri subtree kanan
        System.out.printf(format:"%s ", node.getData()); // print
    }

    // method untuk mencari nilai dalam BST
    public void searchBST(E key) {
        boolean hasil = searchBSTHelper(root, key);
        if (hasil)
            System.out.println("Data ditemukan " + key);
        else
            System.out.println("Data tidak ditemukan " + key);
    }

    // method rekursif untuk mencari nilai dalam BST
    public boolean searchBSTHelper(TreeNode<E> node, E key) {
        boolean result = false;
        if (node != null) {
            if (key.equals(node.getData()))
                result = true;
            else if (key.compareTo(node.getData()) < 0)
                result = searchBSTHelper(node.getLeftNode(), key);
            else
                result = searchBSTHelper(node.getRightNode(), key);
```



```java
            else if (key.compareTo(node.getData()) < 0)
                result = searchBSTHelper(node.getLeftNode(), key);
            else
                result = searchBSTHelper(node.getRightNode(), key);
```

```java
        }
        return result;
    }
}
```

Main :



Main.java > Main

```
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Tree<Character> tree = new Tree<>();

        System.out.println(x:"Inserting the following values: ");
        // deklarasikan nilai yang ingin di eksekusi
        char[] values = { 'F', 'E', 'H', 'D', 'G', 'C', 'B', 'H', 'K', 'J' };

        for (char value : values) {
            System.out.printf(format:"%c ", value);
            tree.insertNode(value);
        }

        // untuk print preorder
        System.out.printf(format:"%n%nPreorder traversal%n");
        tree.preorderTraversal();

        // untuk print inorder
        System.out.printf(format:"%n%nInorder traversal%n");
        tree.inorderTraversal();

        // print postorder
        System.out.printf(format:"%n%nPostorder traversal%n");
        tree.postorderTraversal();

        // dan untuk print hasil penelusuran
        System.out.println();
        tree.searchBST(key:'K');
        tree.searchBST(key:'A');
    }
}
```