Ahmad Siddiq Priaji (22/496854/PA/21370)

BinaryTreeNode.java

```
public class BinaryTreeNode {
    String label;
    BinaryTreeNode left;
    BinaryTreeNode right;

    BinaryTreeNode (String label, BinaryTreeNode left, BinaryTreeNode right) {
        this.label = label;
        this.left = left;
        this.right = right;
    }

    public String toString() {
        return "[" + label + ", " + left + ", " + right + "]";
    }
}
```

TraverseBinaryTree

```
public class TraverseBinaryTree {
    public static void traversePreorder(BinaryTreeNode t) {
        if (t == null) return;
        System.out.println("visited node " + t.label);
        traversePreorder(t.left);
        traversePreorder(t.right);
    }

    public static void traverseInorder(BinaryTreeNode t) {
        if (t == null) return;
        traverseInorder(t.left);
        System.out.println("visited node " + t.label);
        traverseInorder(t.right);
    }

    public static void traversePostorder(BinaryTreeNode t) {
        if (t == null) return;
        traversePostorder(t.left);
        traversePostorder(t.right);
        System.out.println("visited node " + t.label);
    }
```

```
    public static void main(String[] args) {
        BinaryTreeNode tree, p, q;
        p = new BinaryTreeNode("c", null, null);
        p = new BinaryTreeNode("b", p, null);
        q = new BinaryTreeNode("d", null, null);
        tree = new BinaryTreeNode("a", p, q);

        System.out.println(tree);

        System.out.println("Preorder traversal");
        traversePreorder(tree);

        System.out.println("Inorder traversal");
        traverseInorder(tree);

        System.out.println("Postorder traversal");
        traversePostorder(tree);
    }
}
```

Hasil:

```
[a, [b, [c, null, null], null], [d, null, null]]
Preorder traversal
visited node a
visited node b
visited node c
visited node d
Inorder traversal
visited node c
visited node b
visited node a
visited node d
Postorder traversal
visited node c
visited node b
visited node d
visited node a
```

ShowBinaryTree.java

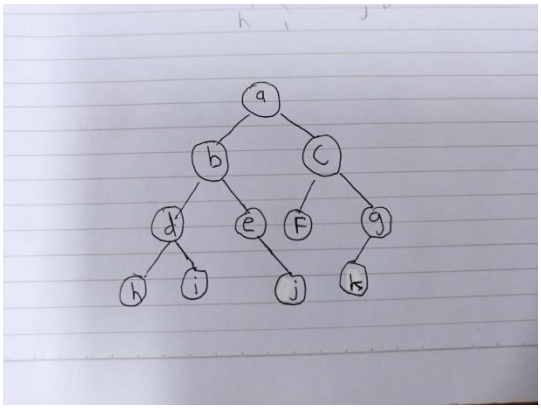
```
public class ShowBinaryTree {
    private static String getSpaces(int n) {
        String s = "";
        for (int i=0; i<n; i++) s += " ";
```

```
        return s;
    }

    public static String prettyPrint(BinaryTreeNode node) {
        return prettyPrint(0, node);
    }

    public static String prettyPrint(int n, BinaryTreeNode node) {
        if (node == null) return getSpaces(n) + "null\n";
        String s = "";
        s += prettyPrint(n+2, node.right);
        s += getSpaces(n) + node.label + "\n";
        s += prettyPrint(n+2, node.left);
        return s;
    }

    public static void main(String[] args) {
        BinaryTreeNode tree, p, q, r;
        p = new BinaryTreeNode("h", null, null);
        q = new BinaryTreeNode("i", null, null);
        p = new BinaryTreeNode("d", p, q);
        q = new BinaryTreeNode("j", null, null);
        q = new BinaryTreeNode("e", null, q);
        r = new BinaryTreeNode("b", p, q);

        p = new BinaryTreeNode("f", null, null);
        q = new BinaryTreeNode("k", null, null);
        q = new BinaryTreeNode("g", q, null);
        p = new BinaryTreeNode("c", p, q);

        tree = new BinaryTreeNode("a", r, p);
        System.out.println(prettyPrint(tree));
    }

}
```

Hasil:

```
        null
      g
          null
        k
          null
    c
        null
      f
        null
  a
          null
        j
          null
      e
        null
    b
          null
        i
          null
      d
          null
        h
          null
```

Gambar manual:



Hasil tree pada program sama seperti gambar manual