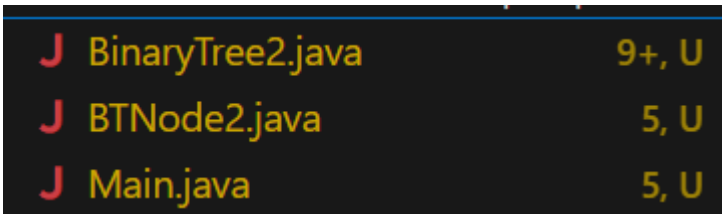
1. File

- 

    - BinaryTree.java berfungsi sebagai tempat untuk menyimpan perintah perintah pembuatan Binary Tree
    - BTNode.java berfungsi sebagai tempat untuk menyimpan perintah perintah untuk mengecek setiap node pada binary tree yang sudah kita buat dengan perintah dari BinaryTree.java
    - Main berfungsi sebagai tempat untuk eksekusi perintah perintah yang sudah kita buat sebelumnya
2. BinaryTree.java

- ```
  /* Function to insert data recursively */
  private BTNode2 insert(BTNode2 node, E data) {
      if (node == null)
          node = new BTNode2(data);
      else {
          if (node.getLeft() == null)
              node.left = insert(node.left, data);
          else
              node.right = insert(node.right, data);
      }
      return node;
  }
  ```
    - Fungsi diatas adalah untuk melakukan insert suatu data ke dalam binarytree yang telah kita buat dengan memasukkan data ke bagian kiri dahulu baru ke sebelah kanan jika bagian kiri sudah terisi
  ```

```java
/* Function to count number of nodes */
public int countNodes() {
    return countNodes(root);
}

/* Function to count number of nodes recursively */
private int countNodes(BTNode2 r) {
    if (r == null)
        return 0;
    else {
        int l = 1;
        l += countNodes(r.getLeft());
        l += countNodes(r.getRight());
        return l;
    }
}
```

- Fungsi untuk menghitung banyak node atau data yang tersimpan pada suatu tree

```java
/* Function to search for an element recursively */
private boolean search(BTNode2 r, E val) {
    if (r.getData() == val)
        return true;
    if (r.getLeft() != null)
        if (search(r.getLeft(), val))
            return true;
    if (r.getRight() != null)
        if (search(r.getRight(), val))
            return true;
    return false;
}
```

- Fungsi untuk mencari suatu data menggunakan perintah **search**

```java
/* Function for inorder traversal */
public void inorder() {
    inorder(root);
}

private void inorder(BTNode2 r) {
    if (r != null) {
        inorder(r.getLeft());
        System.out.print(r.getData() + " ");
        inorder(r.getRight());
    }
}
```

- Fungsi untuk mengurutkan data dengan metode **inorder**

```java
/* Function for preorder traversal */
public void preorder() {
    preorder(root);
}

private void preorder(BTNode2 r) {
    if (r != null) {
        System.out.print(r.getData() + " ");
        preorder(r.getLeft());
        preorder(r.getRight());
    }
}
```

- Fungsi untuk mengurutkan data dengan metode **preorder**

```java
/* Function for postorder traversal */
public void postorder() {
    postorder(root);
}

private void postorder(BTNode2 r) {
    if (r != null) {
        postorder(r.getLeft());
        postorder(r.getRight());
        System.out.print(r.getData() + " ");
    }
}
```

- Fungsi untuk mengurutkan data dengan metode **postorder**

3. BTnode.java

```java
/* Constructor */
public BTNode2() {
    left = null;
    right = null;
    data = null;
}

/* Constructor */
public BTNode2(E item) {
    left = null;
    right = null;
    data = item;
}
```

- Konstruktor untuk perintah

```java
/* Function to set left node */
public void setLeft(BTNode2 n) {
    left = n;
}

/* Function to set right node */
public void setRight(BTNode2 n) {
    right = n;
}

/* Function to get left node */
public BTNode2 getLeft() {
    return left;
}

/* Function to get right node */
public BTNode2 getRight() {
    return right;
}

/* Function to set data to node */
public void setData(E d) {
    data = d;
}

/* Function to get data from node */
public E getData() {
    return data;
}
```

- Fungsi Fungsi untuk menjelajahi setiap node dan mencetak node menggunakan **getData**