Nama : Alfina Salsabilla

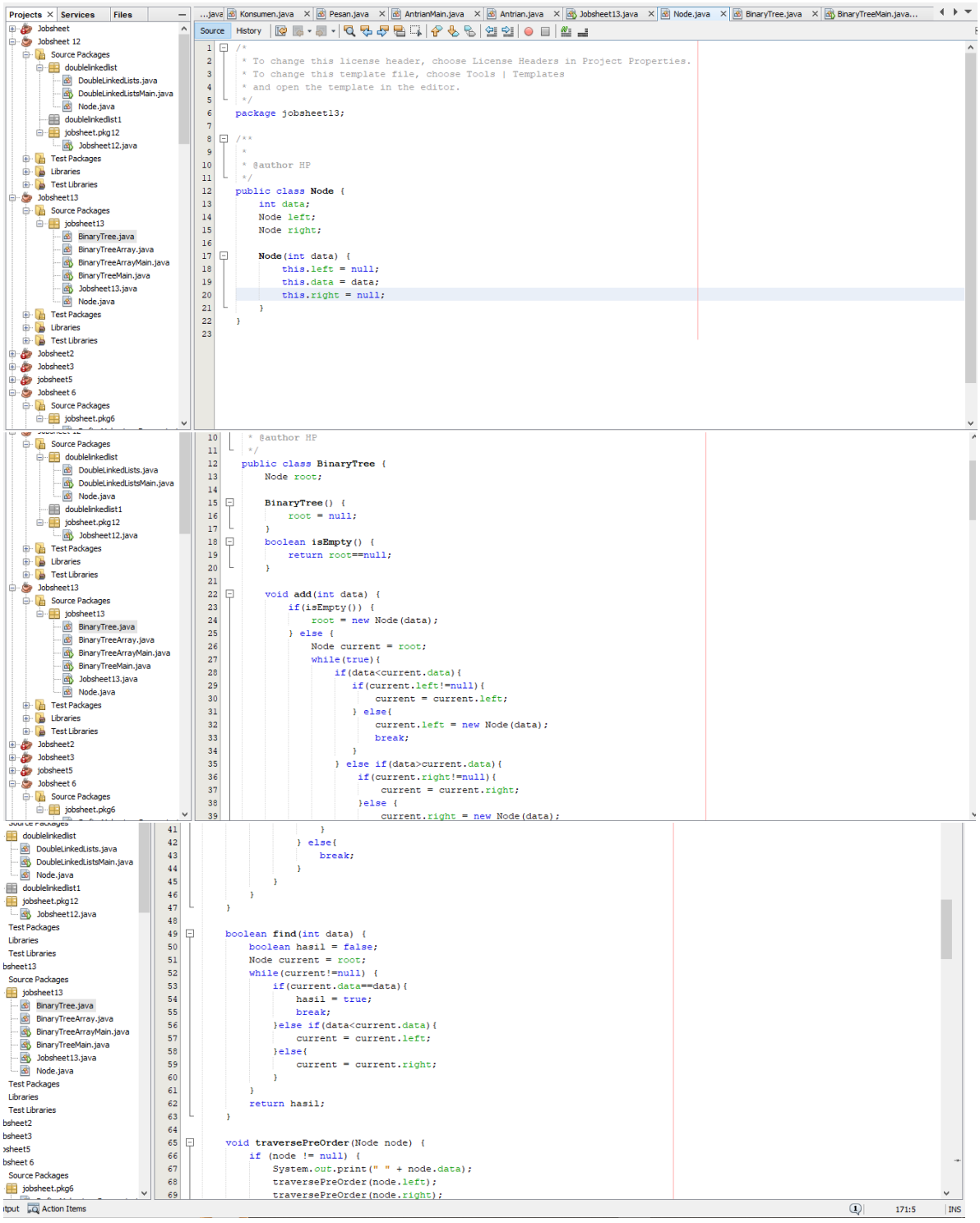
NIM : 2141720044

Kelas / Absen : TI-1G / 03

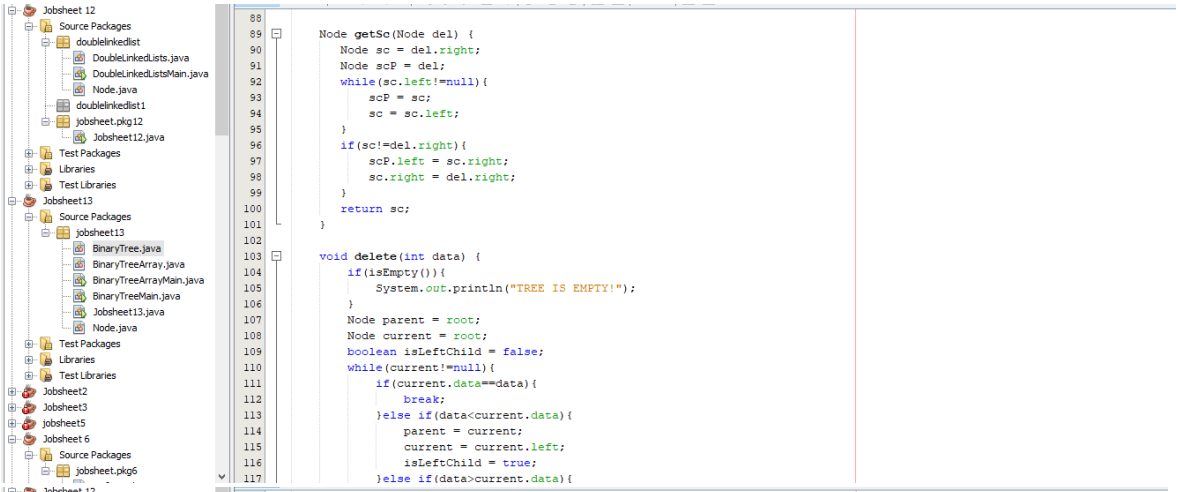
Tugas :

JOBSHEET 14!

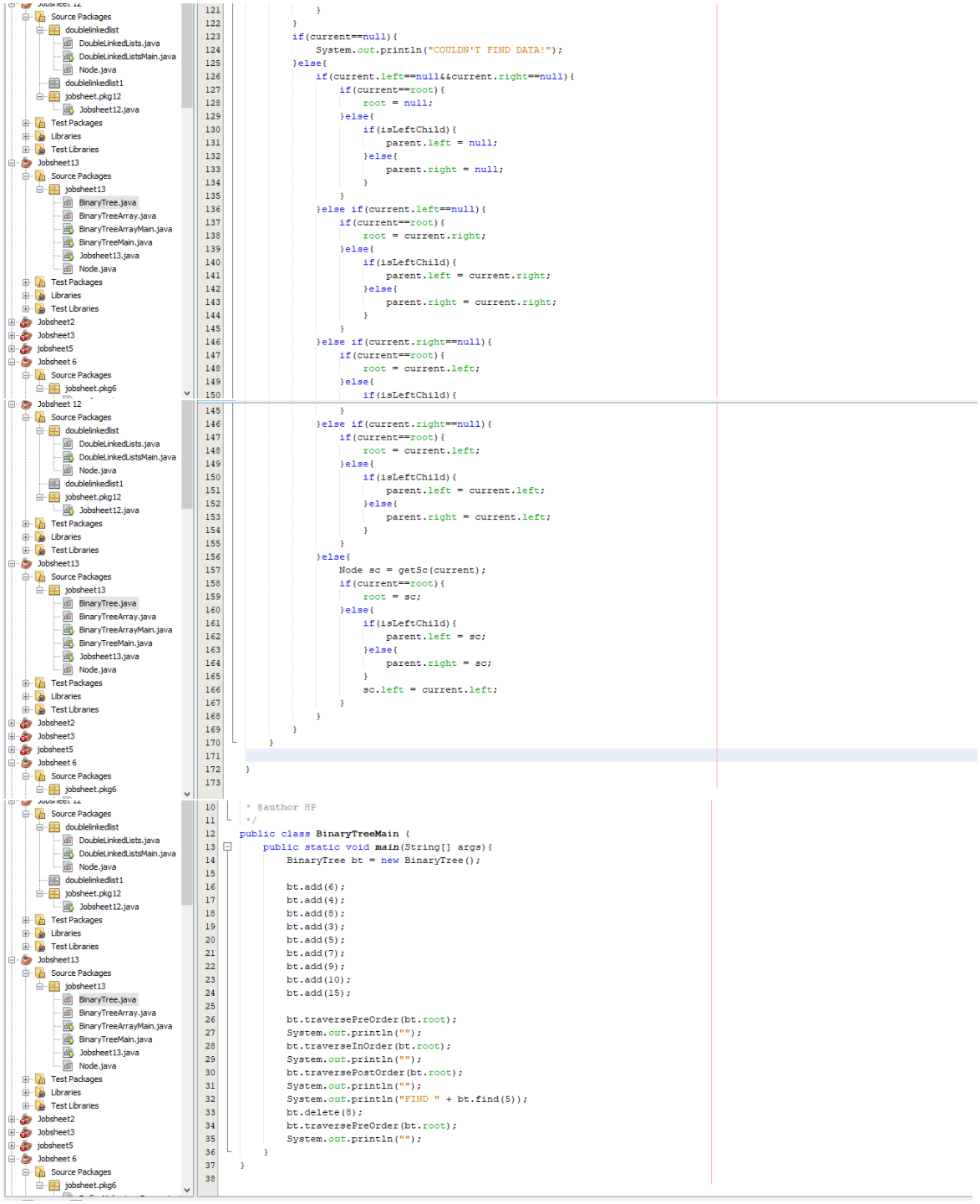
- Praktikum :



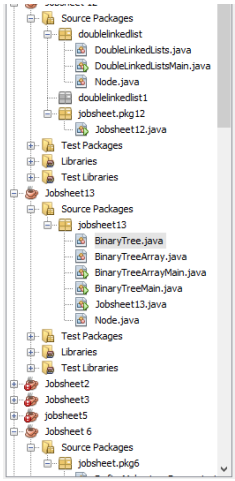

Jobsheet 12
Source Packages
doublelinkedlist
DoubleLinkedLists.java
DoubleLinkedListsMain.java
Node.java
doublelinkedlist1
jobsheet.pkg12
Jobsheet12.java
Test Packages
Libraries
Test Libraries
Jobsheet13
Source Packages
jobsheet13
BinaryTree.java
BinaryTreeArray.java
BinaryTreeArrayMain.java
BinaryTreeMain.java
Jobsheet13.java
Node.java
Test Packages
Libraries
Test Libraries
Jobsheet2
Jobsheet3
jobsheet5
Jobsheet 6
Source Packages
jobsheet.pkg6
64
65 void traversePreOrder(Node node) {
66 if (node != null) {
67 System.out.print(" " + node.data);
68 traversePreOrder(node.left);
69 traversePreOrder(node.right);
70 }
71 }
72
73 void traversePostOrder(Node node){
74 if (node != null) {
75 traversePostOrder(node.left);
76 traversePostOrder(node.right);
77 System.out.print(" " + node.data);
78 }
79 }
80
81 void traverseInOrder(Node node){
82 if (node != null) {
83 traverseInOrder(node.left);
84 System.out.print(" " + node.data);
85 traverseInOrder(node.right);
86 }
87 }
88
89 Node getSc(Node del) {
90 Node sc = del.right;
91 Node scP = del;
92 while(sc.left!=null){
93 scP = sc;


Jobsheet 12
Source Packages
doublelinkedlist
DoubleLinkedLists.java
DoubleLinkedListsMain.java
Node.java
doublelinkedlist1
jobsheet.pkg12
Jobsheet12.java
Test Packages
Libraries
Test Libraries
Jobsheet13
Source Packages
jobsheet13
BinaryTree.java
BinaryTreeArray.java
BinaryTreeArrayMain.java
BinaryTreeMain.java
Jobsheet13.java
Node.java
Test Packages
Libraries
Test Libraries
Jobsheet2
Jobsheet3
jobsheet5
Jobsheet 6
Source Packages
jobsheet.pkg6
88
89 Node getSc(Node del) {
90 Node sc = del.right;
91 Node scP = del;
92 while(sc.left!=null){
93 scP = sc;
94 sc = sc.left;
95 }
96 if(sc!=del.right){
97 scP.left = sc.right;
98 sc.right = del.right;
99 }
100 return sc;
101 }
102
103 void delete(int data) {
104 if(isEmpty()){
105 System.out.println("TREE IS EMPTY!");
106 }
107 Node parent = root;
108 Node current = root;
109 boolean isLeftChild = false;
110 while(current!=null){
111 if(current.data==data){
112 break;
113 }else if(data<current.data){
114 parent = current;
115 current = current.left;
116 isLeftChild = true;
117 }else if(data>current.data){

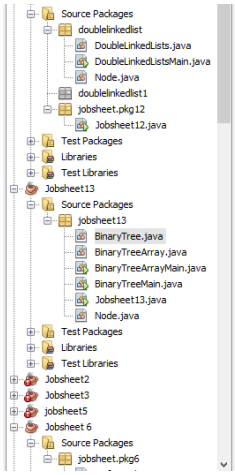

Jobsheet 12
Source Packages
doublelinkedlist
DoubleLinkedLists.java
DoubleLinkedListsMain.java
Node.java
doublelinkedlist1
jobsheet.pkg12
Jobsheet12.java
Test Packages
Libraries
Test Libraries
Jobsheet13
Source Packages
jobsheet13
BinaryTree.java
BinaryTreeArray.java
BinaryTreeArrayMain.java
BinaryTreeMain.java
Jobsheet13.java
Node.java
Test Packages
Libraries
Test Libraries
Jobsheet2
Jobsheet3
jobsheet5
Jobsheet 6
Source Packages
jobsheet.pkg6
103 void delete(int data) {
104 if(isEmpty()){
105 System.out.println("TREE IS EMPTY!");
106 }
107 Node parent = root;
108 Node current = root;
109 boolean isLeftChild = false;
110 while(current!=null){
111 if(current.data==data){
112 break;
113 }else if(data<current.data){
114 parent = current;
115 current = current.left;
116 isLeftChild = true;
117 }else if(data>current.data){
118 parent = current;
119 current = current.right;
120 isLeftChild = false;
121 }
122 }
123 if(current==null){
124 System.out.println("COULDN'T FIND DATA!");
125 }else{
126 if(current.left==null&&current.right==null){
127 if(current==root){
128 root = null;
129 }else{
130 if(isLeftChild){
131 parent.left = null;
132 }else{



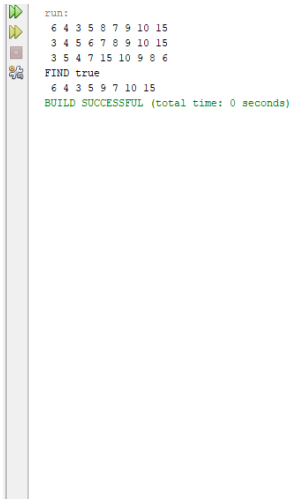

```
            }
        }
        if(current==null){
            System.out.println("COULDN'T FIND DATA!");
        }else{
            if(current.left==null&&current.right==null){
                if(current==root){
                    root = null;
                }else{
                    if(isLeftChild){
                        parent.left = null;
                    }else{
                        parent.right = null;
                    }
                }
            }else if(current.left==null){
                if(current==root){
                    root = current.right;
                }else{
                    if(isLeftChild){
                        parent.left = current.right;
                    }else{
                        parent.right = current.right;
                    }
                }
            }else if(current.right==null){
                if(current==root){
                    root = current.left;
                }else{
                    if(isLeftChild){
                    }
                }else if(current.right==null){
                    if(current==root){
                        root = current.left;
                    }else{
                        if(isLeftChild){
                            parent.left = current.left;
                        }else{
                            parent.right = current.left;
                        }
                    }
                }else{
                    Node sc = getSc(current);
                    if(current==root){
                        root = sc;
                    }else{
                        if(isLeftChild){
                            parent.left = sc;
                        }else{
                            parent.right = sc;
                        }
                        sc.left = current.left;
                    }
                }
            }
        }

    }

 * @author HP
 */
public class BinaryTreeMain {
    public static void main(String[] args){
        BinaryTree bt = new BinaryTree();

        bt.add(6);
        bt.add(4);
        bt.add(8);
        bt.add(3);
        bt.add(5);
        bt.add(7);
        bt.add(9);
        bt.add(10);
        bt.add(15);

        bt.traversePreOrder(bt.root);
        System.out.println("");
        bt.traverseInOrder(bt.root);
        System.out.println("");
        bt.traversePostOrder(bt.root);
        System.out.println("");
        System.out.println("FIND " + bt.find(5));
        bt.delete(8);
        bt.traversePreOrder(bt.root);
        System.out.println("");
    }
}
```



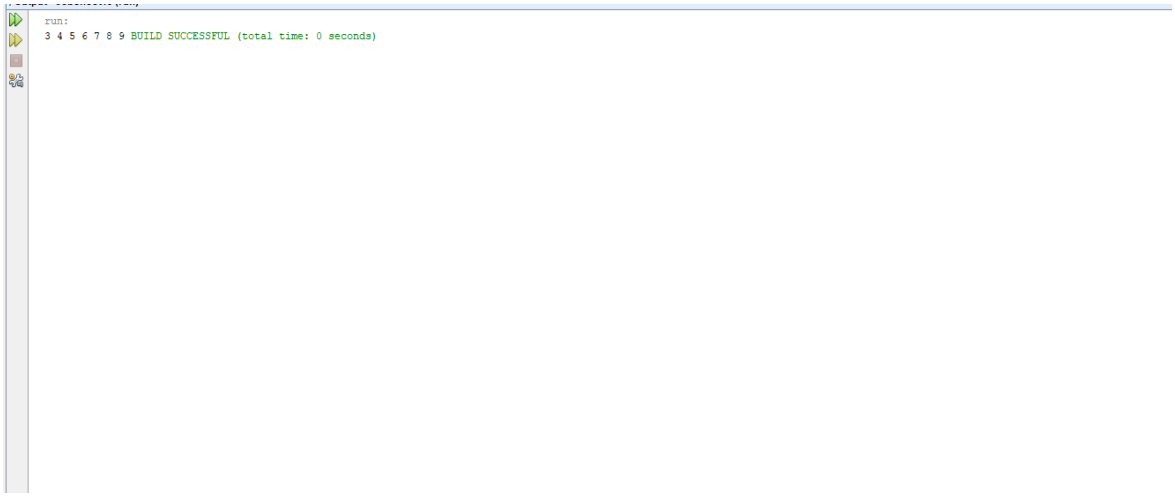

```java
/**
 *
 * @author HP
 */
public class BinaryTreeArray {
    int[] data;
    int idxLast;

    BinaryTreeArray(){
        data = new int[10];
    }

    void populateData(int data[], int idxLast){
        this.data = data;
        this.idxLast = idxLast;
    }

    void traverseInOrder(int idxStart){
        if(idxStart<=idxLast){
            traverseInOrder(2*idxStart+1);
            System.out.print(data[idxStart] + " ");
            traverseInOrder(2*idxStart+2);
        }
    }
}
```




```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package jobsheet13;

/**
 *
 * @author HP
 */
public class BinaryTreeArrayMain {
    public static void main(String[] args) {
        BinaryTreeArray bta = new BinaryTreeArray();
        int[] data = {6, 4, 8, 3, 5, 7, 9, 0, 0, 0};
        int idxLast = 6;
        bta.populateData(data, idxLast);
        bta.traverseInOrder(0);
    }
}
```



```
run:
 6 4 3 5 8 7 9 10 15
 3 4 5 6 7 8 9 10 15
 3 5 4 7 15 10 9 8 6
FIND true
 6 4 3 5 9 7 10 15
BUILD SUCCESSFUL (total time: 0 seconds)
```




- Tugas :

awaban Pertanyaan Percobaan Jobsheet 13

## Pertanyaan Percobaan 2.1.2

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?

**Jawab: karena pada Binary Search Tree terdapat aturan bahwa setiap child node sebelah kiri selalu lebih kecil nilainya dari pada root node. Begitu pula sebaliknya, setiap child node sebelah kanan selalu lebih besar nilainya daripada root node, yang dapat memberikan proses efisiensi pada proses searching.**

2. Untuk apakah di class Node, kegunaan dari atribut left dan right?

**Jawab: untuk menentukan leftchild dan reightchild. kegunaan left dan right sama seperti next dan prev**

3. a. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?

**Jawab: kegunaan dari root adalah untuk menentukan nilai paling atas**

b. Ketika objek tree pertama kali dibuat, apakah nilai dari root?

**Jawab: null atau kosong**

4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?

**Jawab: proses penginputan node ke dalam root**

5. Perhatikan method add(), di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil untuk apa baris program tersebut?

```
if(data<current.data){
 if(current.left!=null){
 current = current.left;
 }else{
 current.left = new Node(data);
 break;
 }
```

**Jawab: jika data baru kurang dari data lama maka di lakukan pengecekan lagi apakah data kiri bernilai sama dengan null, jika iya data lama akan maka akan masuk ke dalam data kiri, jika tidak maka data kiri di ganti dengan data yang baru saja di masukkan, setelah itu break**

## Pertanyaan Percobaan 13.2.1

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?

**Jawab: data untuk mendeklarasikan banyaknya nilai array dan IdxLast untuk menentukan alamat agar tidak eror waktu add**

2. Apakah kegunaan dari method populateData()?

**Jawab: untuk menginput data agar dikenali indexnya**

3. Apakah kegunaan dari method traverseInOrder()?

**Jawab:untuk mengeprint secara inOrder atau mengeprint seluruh data pada tree secara rekursif mulai dari sebelah kiri**

4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masin-masing?

**Jawab: left = 1 dan right 3**

5. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?

**Jawab: untuk melimit index agar hanya menjadi 6**