Laporan Praktikum

Struktur Data Non Linear DP

Modul 2

Dosen Pengampu

JB. Budi Darmawan S.T., M.Sc.

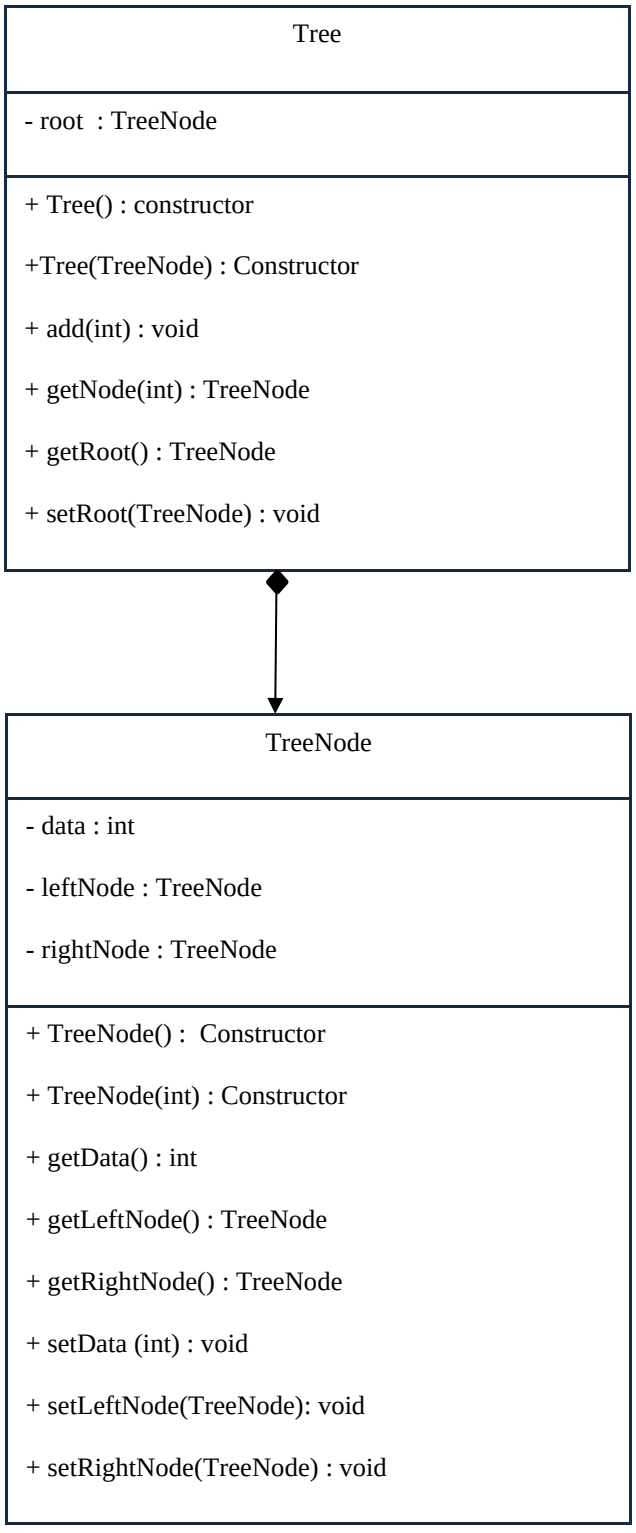
Oleh :

Nama : Maria Gresia Plena Br Purba

NIM : 235314094

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2024

1. Diagram UML



2. Source Code

Tree

add()

```
34      public void add(int x) {
35          TreeNode bantu = root;
36
37          if (isEmpty()) {
38              root = new TreeNode(data: x);
39              return;
40          }
41
42          while (true) {
43              if (x < bantu.getData()) {
44                  if (bantu.leftNode == null) {
45                      bantu.leftNode = new TreeNode(data: x);
46                      return;
47                  } else {
48                      bantu = bantu.leftNode;
49                  }
50              } else if (bantu.rightNode == null) {
51                  bantu.rightNode = new TreeNode(data: x);
52                  return;
53              } else {
54                  bantu = bantu.rightNode;
55              }
56          }
57      }
```

getNode()

```
59      public TreeNode getNode(int key) {
60          TreeNode bantu = root;
61
62          while (bantu != null) {
63              if (key == bantu.getData()) {
64                  return bantu;
65              } else if (key < bantu.getData()) {
66                  bantu = bantu.getLeftNode();
67              } else {
68                  bantu = bantu.getRightNode();
69              }
70          }
71          return bantu;
72      }
```

TreeMain

```java
package TreeBinarySearch;

import java.util.Scanner;

public class TreeMain {

    public static void main(String[] args) {
        Scanner in = new Scanner(source: System.in);
        Tree data = new Tree();

        data.add(x: 42);
        data.add(x: 21);
        data.add(x: 38);
        data.add(x: 27);
        data.add(x: 71);
        data.add(x: 82);
        data.add(x: 55);
        data.add(x: 63);
        data.add(x: 6);
        data.add(x: 2);
        data.add(x: 40);
        data.add(x: 12);

        System.out.println(x: "data yang ditambahkan = 42, 21, 38, 27, 71, 82, 55, 63, 6, 2, 40 dan 12");

        while (true){
            System.out.print(s: "Masukkan angka yang dicari = ");
            int cari = in.nextInt();

            TreeNode kunci = data.getNode(key:cari);
            System.out.println("\ndata yang dicari = " + cari);

            if (kunci != null) {
                System.out.println(x: "data ditemukan");
            } else {
                System.out.println(x: "data tidak ditemukan");
            }
            System.out.print(s: "Ulangi pencarian? (ya / tidak)");
            String pilihan = in.next();

            if (pilihan.equalsIgnoreCase(anotherString: "tidak")) {
                break;
            }
        }
    }
}
```

3. Output

```
run:
data yang ditambahkan = 42, 21, 38, 27, 71, 82, 55, 63, 6, 2, 40 dan 12
Masukkan angka yang dicari = 11

data yang dicari = 11
data tidak ditemukan
Ulangi pencarian? (ya / tidak)ya
Masukkan angka yang dicari = 21

data yang dicari = 21
data ditemukan
Ulangi pencarian? (ya / tidak)tidak
BUILD SUCCESSFUL (total time: 20 seconds)
```

4. Analisa

Tree

Code	Penjelasan

public void add(int x) {
TreeNode bantu = root;
 	
if (isEmpty()) {
root = new TreeNode(x);
return;
}
 	
while (true) {
if (x < bantu.getData()) {
if (bantu.leftNode == null) {
bantu.leftNode = new TreeNode(x);
return;
} else {
bantu = bantu.leftNode;
}
} else if (bantu.rightNode == null) {
bantu.rightNode = new TreeNode(x);
return;
} else {
bantu = bantu.rightNode;
}
}
}
	Memeriksa jika root kosong dengan memanggil method isEmpty(), jika iya, maka akan menambahkan node baru dengan nilai x sebagai root.
	
	Jika nilai x kurang dari node bantu, maka akan memeriksa apakah node kiri dari bantu sama dengan null maka akan menambahkan node baru dengan nilai x ke kiri. Jika tidak, maka pembaruan bergantung kepada perbandingan nilai.
	
	Jika nilai node kanan dari bantu bernilai null, maka nilai dari x akan ditambahkan ke node kanan dari bantu. Jika tidak memenuhi syarat-syarat tersebut, maka bantu akan diperbarui untuk menunjuk ke node kanan.

public TreeNode getNode(int key) {
TreeNode bantu = root;
 	
while (bantu != null) {
if (key == bantu.getData()) {
return bantu;
} else if (key < bantu.getData()) {
bantu = bantu.getLeftNode();
} else {
bantu = bantu.getRightNode();
}
}
return bantu;
}
	Method ini digunakan untuk mencari node dengan nilai tertentu. Terdapat loop yang akan terus berjalan selama nilai bantu tidak sama dengan null. Jika nilai key sama dengan node bantu, maka node tersebut akan dikembalikan. Selain itu, jika nilai key lebih kecil dari node bantu, maka pencarian akan dilanjutkan ke subtree kiri. Jika tidak memenuhi, maka pencarian akan dilanjutkan ke subtree kanan.

TreeNode

Code	Penjelasan
private int data;	Mendeklarasikan atribut data dengan tipe data integer yang bersifat private
private TreeNode leftNode;	Mendeklarasikan atribut leftNode dengan tipe data TreeNode yang bersifat private
private TreeNode rightNode;	Mendeklarasikan atribut rightNode dengan tipe data TreeNode yang bersifat private

public TreeNode() {
this(0);
}
	Berfungsi untuk memanggil constructor yang lain. Lalu, menginisialisasikan data dengan nilai 0

public TreeNode(int data) {
this.data = data;
leftNode = null;
rightNode = null;
}
	Menginisialisasi data dengan nilai yang ada. Menginisialisasi leftNode, serta rightNode dengan null

public int getData() {
return data;
}
	Method ini berfungsi untuk mengembalikan nilai dari data

public TreeNode getLeftNode() {
return leftNode;
}
	Method in berfungsi untuk mengembalikan nilai dari leftNode

public TreeNode getRightNode() {
return rightNode;
}
	Method in berfungsi untuk mengembalikan nilai dari rightNode

public void setData(int data) {
this.data = data;
}
	Method ini berfungsi untuk mengubah nilai data saat ini dengan data baru yang ingin diinput

public void setLeftNode(TreeNode lefftNode) {
this.leftNode = leftNode;
}
	Method ini berfungsi untuk mengubah nilai leftNode saat ini dengan leftNode baru yang ingin diinput

public void setRightNode(TreeNode rightNode) {
this.rightNode = rightNode;
}
	Method ini berfungsi untuk mengubah nilai rightNode saat ini dengan rightNode baru yang ingin diinput

5. Referensi

-