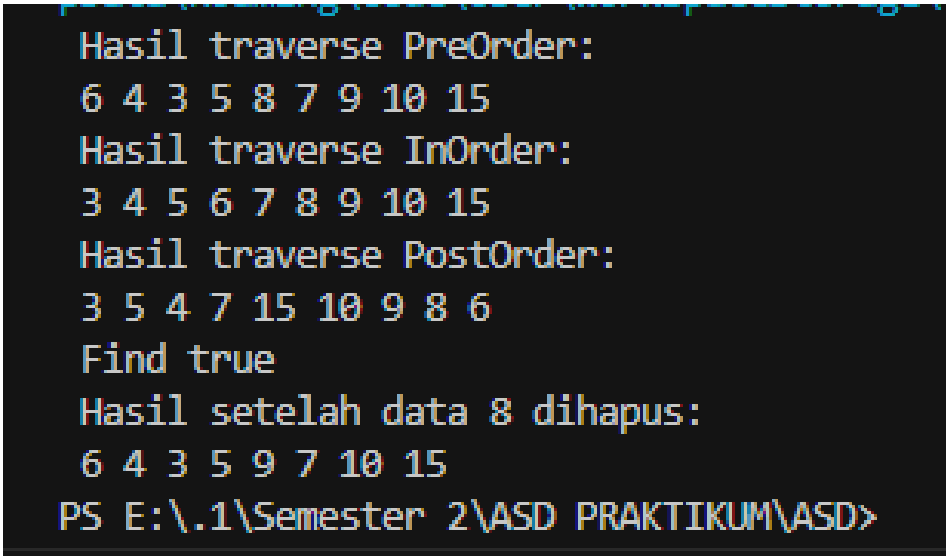
LAPORAN PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA

PRAKTIKUM 12
TREE

Oleh:
JESSICA AMELIA
2341760185
SIB 1A/16

PROGRAM STUDI SISTEM INFORMASI BISNIS
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2023/2024

Praktikum 1



1. **Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?**
   Jawab : Karena penempatan data pada binary search tree, dimana karakteristik left-child harus lebih kecil daripada parent dan parent harus lebih kecil daripada right-child membuat setiap langkah pencarian data lebih terarah dan efisien. Sehingga Binary search tree dibuat untuk mengatasi kelemahan pada binary tree biasa, yaitu kesulitan dalam searching/pencarian node tertentu

2. **Pada class Node, jelaskan kegunaan atribut left, right, dan data?**
   Jawab : Kegunaan atribut left, right, dan data digunakan untuk menyimpan informasi dan menghubungkan node satu sama lain dalam struktur pohon. Dimana data digunakan untuk menyimpan nilai atau informasi yang dimiliki node, left digunakan untuk referensi atau pointer ke anak kiri dari node saat ini, dan right digunakan referensi atau pointer ke anak kanan dari node saat ini.
3. **Ketika objek binary tree pertama kali dibuat, apakah nilai dari root?**
   Jawab : Nilai dari root adalah null, hal ini karena pada saat pembuatan objek binary tree belum ada node dimasukkan ke dalam pohon. Inisialisasi root dengan null menandakan bahwa pohon masih kosong.
4. **Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan dilakukan?**
   Jawab : Pertama, kita periksa apakah pohon kosong atau tidak dengan memeriksa apakah root memiliki nilai null. Jika pohon kosong melakukan inisialisasi root maka node baru tersebut akan menjadi root dari pohon. Kita hanya perlu membuat node baru dan menetapkan node tersebut sebagai 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 : Pada kode program tersebut pertama dilakukan yaitu mengevaluasi data dengan kondisi (data<current.data) apakah nilai data yang ingin ditambahkan lebih kecil daripada nilai data dari current (node saat ini yang sedang diproses). Jika kondisi terpenuhi maka masuk ke pengecekan selanjutnya dengan kondisi (current.left != ,null) apakah left-child dari current sudah terisi atau tidak, jika sudah terisi (tidak null), maka kita harus bergerak ke node left-child tersebut untuk melanjutkan proses penambahan node baru. Jika kondisi sebelumnya terpenuhi (anak kiri tidak null) kita bergerak ke node left-child saat ini dengan menerapkan current menjadi current.left (current = current.left). Ini memungkinkan untuk melanjutkan pencarian posisi yang tepat untuk menyisipkan node baru. Jika kondisi sebelumnya terpenuhi (left-child null) maka kita telah menemukan posisi yang tepat untuk menyisipkan node baru. (current.left = new Node(data) pada baris ini, kita membuat node baru dengan nilai data yang diberikan dan menetapkannya sebagai left-child dari current.

Tugas Praktikum

1. Buat method di dalam class BinaryTree untuk menampilkan data yang ada pada leaf.

```
// Metode untuk menampilkan data pada leaf
public void printLeaf(Node node) {
    if (node == null) {
        return;
    }
    if (node.left == null && node.right == null) { // Cek apakah node merupakan leaf
        System.out.print(" " + node.data);
    }
    // Rekursif: lanjutkan ke anak kiri dan anak kanan
    printLeaf(node.left);
    printLeaf(node.right);
}
```

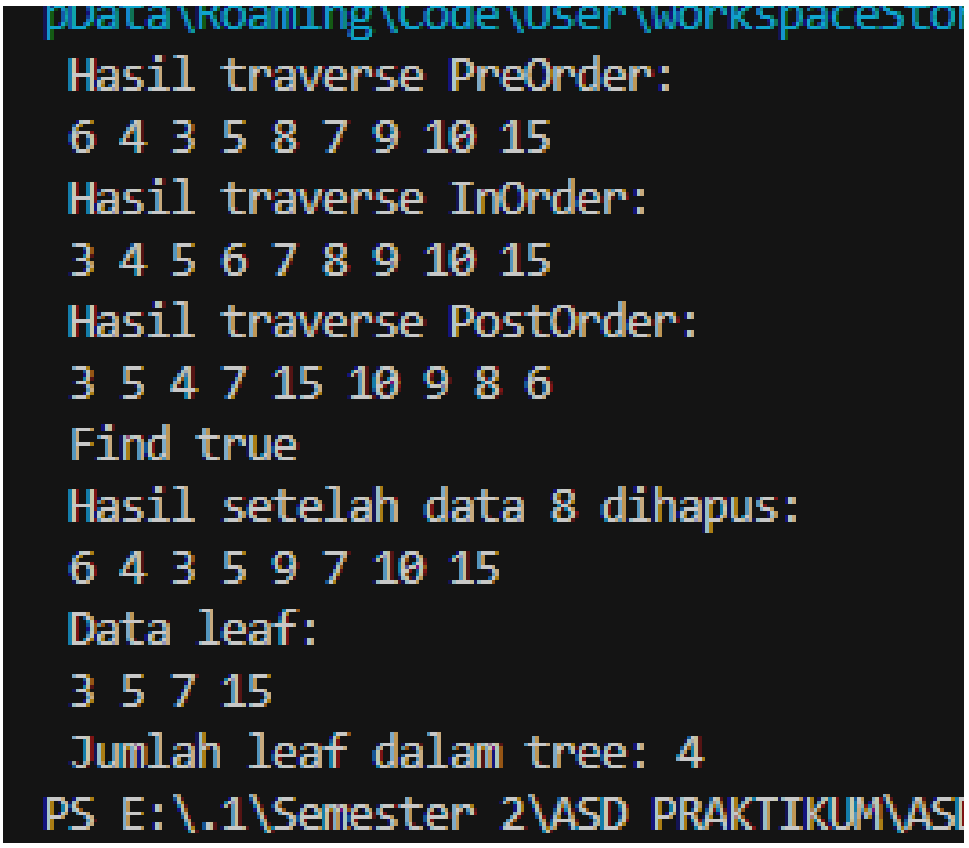
2. Buat method di dalam class BinaryTree untuk menampilkan berapa jumlah leaf yang ada di dalam tree.

```java
// Metode untuk menghitung jumlah leaf
public int countLeaf(Node node) {
    if (node == null) {
        return 0;
    }
    if (node.left == null && node.right == null) { // Jika node merupakan leaf, return 1
        return 1;
    }
    // Rekursif: jumlah leaf dari anak kiri dan anak kanan
    return countLeaf(node.left) + countLeaf(node.right);
}
```

Main dan Output dari dua method diatas

```java
public class BinaryTreeMain {
    public static void main(String[] args) {
        bt.add(data:6);
        bt.add(data:4);
        bt.add(data:8);
        bt.add(data:3);
        bt.add(data:5);
        bt.add(data:7);
        bt.add(data:9);
        bt.add(data:10);
        bt.add(data:15);

        System.out.println(x:" Hasil traverse PreOrder: ");
        bt.traversePreOrder(bt.root);
        System.out.println(x:"");
        System.out.println(x:" Hasil traverse InOrder: ");
        bt.traverseInOrder(bt.root);
        System.out.println(x:"");
        System.out.println(x:" Hasil traverse PostOrder: ");
        bt.traversePostOrder(bt.root);
        System.out.println(x:"");
        System.out.println(" Find " + bt.find(data:5));
        System.out.println(x:" Hasil setelah data 8 dihapus: ");
        bt.delete(data:8);
        bt.traversePreOrder(bt.root);
        System.out.println(x:"");
        System.out.println(x:" Data leaf: ");
        bt.printLeaf(bt.root);
        System.out.println(x:"");
        System.out.println(" Jumlah leaf dalam tree: " + bt.countLeaf(bt.root));
```


pData\Roaming\Code\User\workspaceSto
Hasil traverse PreOrder:
6 4 3 5 8 7 9 10 15
Hasil traverse InOrder:
3 4 5 6 7 8 9 10 15
Hasil traverse PostOrder:
3 5 4 7 15 10 9 8 6
Find true
Hasil setelah data 8 dihapus:
6 4 3 5 9 7 10 15
Data leaf:
3 5 7 15
Jumlah leaf dalam tree: 4
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASI