PRAKTIKUM 25
TRAVERSAL BINARY TREE

A. TUJUAN

Mahasiswa diharapkan mampu :

1. Memahami konsep dari pembacaan Binary Tree dengan traversal Inorder, Preorder dan PostOrder
2. Mengimplementasikan pembacaan Binary Tree dengan traversal Inorder, Preorder dan PostOrder

B. DASAR TEORI

Struktur pohon (tree) biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Contoh penggunaan struktur pohon :

- Silsilah keluarga
- Hasil pertandingan yang berbentuk turnamen
- Struktur organisasi dari sebuah perusahaan

Sebuah binary tree adalah sebuah pengorganisasian secara hirarki dari beberapa buah simpul, dimana masing-masing simpul tidak mempunyai anak lebih dari 2. Simpul yang berada di bawah sebuah simpul dinamakan anak dari simpul tersebut. Simpul yang berada di atas sebuah simpul dinamakan induk dari simpul tersebut.

Algoritma Binary Tree Traversal

Proses traversal adalah proses melakukan kunjungan pada setiap node pada suatu binary tree tepat satu kali. Dengan melakukan kunjungan secara lengkap, maka akan didapatkan urutan informasi secara linier yang tersimpan dalam sebuah binary tree. Terdapat tiga teknik rekursif untuk binary tree traversals ,yaitu:

1. Mengunjungi simpul akar (root),
2. Melakukan traversal subpohon kiri (left subtree), dan
3. Melakukan traversal subpohon kanan (right subtree).

Yang membedakan antara teknik satu dengan yang lain adalah proses pengurutan tugas mereka.

Terdapat tiga cara untuk melakukan kunjungan itu, yaitu:

1. **Traversal preorder (depth first order)**

   Dilaksanakan dengan jalan mencetak isi node yang dikunjungi lalu melakukan kunjungan ke subtree kiri dan selanjutnya ke subtree kanan. Algoritma umum traversal preorder adalah sebagai berikut:

   - ✓ Jika tree kosong, maka keluar
   - ✓ Proses node root.
   - ✓ Traverse subtree kiri secara preorder.
   - ✓ Traverse subtree kanan secara preorder.

2. **Traversal inorder (symmetric order)**

   Dilaksanakan dengan jalan melakukan kunjungan ke subtree kiri, mencetak isi node yang dikunjungi, lalu melakukan kunjungan ke subtree kanan. Algoritma umum traversal inorder adalah sebagai berikut:

   - ✓ Jika tree kosong, maka keluar.
   - ✓ Traverse subtree kiri secara inorder.
   - ✓ Proses node root.
   - ✓ Traverse subtree kanan secara inorder.

3. **Traversal postorder**

   Dilaksanakan dengan jalan melakukan kunjungan ke subtree kiri, lalu ke subtree kanan, dan selanjutnya mencetak isi node yang dikunjungi. Algoritma umum traversal inorder adalah sebagai berikut:

   - ✓ Jika tree kosong, maka keluar.
   - ✓ Traverse subtree kiri secara postorder.
   - ✓ Traverse subtree kanan secara postorder.
   - ✓ Proses node root.

Semua algoritma traversal (preorder, inorder, postorder) yang diberikan di atas berupa algoritma rekursif, dan sebenarnya dapat dikerjakan secara iteratif dengan bantuan stack.

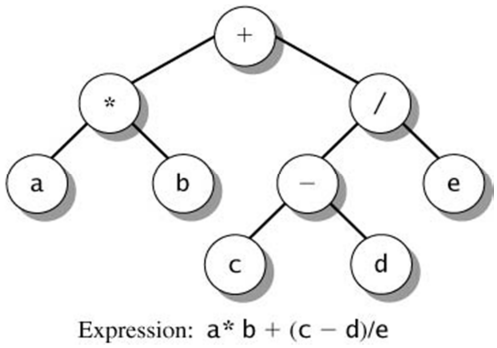
Contoh: diberikan ekspresi matematika ((A * B) - (C ^ D)) + (E / F), apabila digambarkan dalam bentuk binary tree.

Apabila binary tree di atas dikunjungi:

- secara preorder akan menghasilkan: + - * A B ^ C D / E F
- secara inorder akan menghasilkan: A * C - C ^ D + E / F
- secara postorder akan menghasilkan: A B + C D ^ - E F / +

C. TUGAS PENDAHULUAN

Terdapat Binary Tree yang merepresentasikan sebuah notasi aritmatika a * b + (c – d) / e



Gambar 1. Binary Tree dengan notasi aritmatika a * b + (c – d) / e

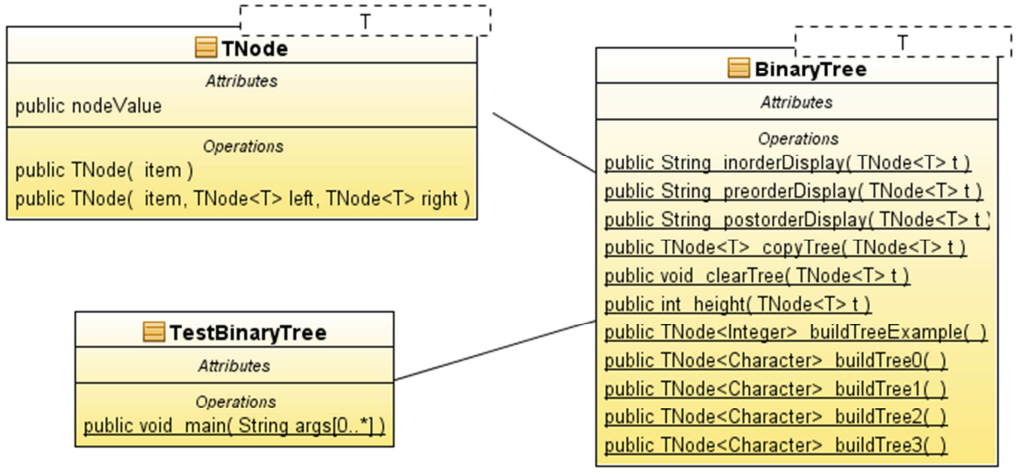
Buatlah digram rekursif untuk pembacaan Binary Tree menggunakan Traversal PostOrder

D. PERCOBAAN

Pada percobaan dan latihan ini seperti ditunjukkan pada gambar 2, terdapat Class TNode, Class Binary Tree dan Class Test BinaryTree. Class TNode merepresentasikan sebuah Node pada Binary Tree. Pada Class Binary Tree terdapat method :

- preorderDisplay(TNode<T> t) : untuk traversal Tree dengan cara Preorder
- inorderDisplay(TNode<T> t) : untuk traversal Tree dengan cara Inorder
- postorderDisplay(TNode<T> t) : untuk traversal Tree dengan cara Postorder
- buildTreeExample(), buildTree0(),buildTree1(),buildTree2(),buildTree3() : method untuk membangun tree berdasarkan studi kasus.
- copyTree(TNode<T> t) : untuk mengkopi sebuah Binary Tree
- clearTree(TNode<T> t) : untuk menghapus Binary Tree
- height(TNode<T> t) : untuk menghitung kedalaman Binary Tree

Class Test BinaryTree untuk menguji class Binary Tree yang sudah dibuat.



Gambar 2. Class Diagram dari praktikum Binary Tree

Percobaan 1 : Pembacaan Binary Tree dengan Traversal Inorder

```
public class BinaryTree<T> {

   public static <T> String inorderDisplay(TNode<T> t) {
        String s = "";

        if (t != null) {
            s += inorderDisplay(t.left);   // descend left
            s += t.nodeValue + "  ";       // display the node
            s += inorderDisplay(t.right);  // descend right
        }

        return s;
    }
}
```

Percobaan 2 : Pembacaan Binary Tree dengan Traversal Preorder

```
public class BinaryTree<T> {
   public static <T> String preorderDisplay(TNode<T> t) {
        String s = "";

        if (t != null) {
            s += t.nodeValue + "  ";       // display the node
            s += preorderDisplay(t.left);   // descend left
            s += preorderDisplay(t.right);  // descend right
        }
        return s;
   }
}
```

Percobaan 3: Pembacaan Binary Tree dengan Traversal Postorder

```
public class BinaryTree<T> {

   public static <T> String postorderDisplay(TNode<T> t) {
        String s = "";

        if (t != null) {
            s += postorderDisplay(t.left);   // descend left
            s += postorderDisplay(t.right);  // descend right
            s += t.nodeValue + "  ";       // display the node
        }
        return s;
    }
}
```

Selanjutnya ujilah method `inorderDisplay(TNode<T> t), preorderDisplay(TNode<T> t), postorderDisplay(TNode<T> t).`

```
public class TestBinaryTree {
    public static void main(String[] args) {
     TNode<Character> root = BinaryTree.buildTree0();
     System.out.println("Preorder : " + BinaryTree.preorderDisplay(root));
     System.out.println("Inorder : " + BinaryTree.inorderDisplay(root));
     System.out.println("Postorder : " +
BinaryTree.postorderDisplay(root));
    }
}
```

E. LATIHAN

Latihan 1 : Untuk studi kasus pada gambar 1, buatlah Binary Tree secara manual. Lakukan pembacaan Binary Tree dengan Traversal Inorder, Preorder dan Postorder.

Latihan 2 : Buatlah Binary Tree dari notasi infix berikut. Selanjutnya lakukan pembacaan Binary Tree dengan Traversal Inorder, Preorder dan Postorder.

- **A + B – C + D**
- **(A + B) * (C-D)**
- **A + (B-C) / D * E**
- **A * B - C ^ D + E / F**

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.