

**Worksheet pertemuan 9 - 1**  
**Algoritma dan Struktur Data**  
**Tree**

**NIM: 20523164**

**Nama: Fajrun Shubhi**

**A. Membuat Folder Untuk Menyimpan Hasil Praktikum**

1. Siapkan folder kosong dengan nama menggunakan NIM masing-masing. Jika folder NIM pada pertemuan sebelumnya mau dimanfaatkan, jangan lupa pindahkan dulu isinya ke folder lain sebagai arsip.
2. Folder ini akan dijadikan tempat untuk menyimpan semua pdf dari worksheet ini beserta fail praktikum lainnya.

**B. Membuat class Node**

1. Siapkan class ArrayList yang dulu pernah dibuat, kita akan menggunakannya lagi di latihan ini
2. Buat sebuah class dengan nama **Node**
3. Kemudian salin tempel kode pogram di bawah ini

```
public class Node<E>{
    private E data = null;
    private ArrayList<Node> children = new ArrayList<>();
    private Node parent = null;
    private int level = 0;

    public Node(E data) {
        this.data = data;
    }

    public Node(){

    }

    public void addChild(Node child) {
        child.setParent(this);
        this.children.add(child);
    }

    public void addChild(E data) {
        Node<E> newChild = new Node<>(data);
        this.addChild(newChild);
    }

    private void setParent(Node parent) {
        this.parent = parent;
    }
}
```

```

        this.level = parent.getLevel() + 1;
    }

    public ArrayList<Node> getChildren() {
        return children;
    }

    public int getNumChild() {
        return this.children.size();
    }

    public int getLevel(){
        return this.level;
    }

    public Object getData() {
        return data;
    }

    public void setData(E data) {
        this.data = data;
    }

    public Node getParent() {
        return parent;
    }

    public boolean isLeaf() {
        return this.children.size() == 0;
    }
}

```

### C. Membuat Class Tree

1. Buat sebuah class dengan nama **Tree**
2. Kemudian salin tempel kode program di bawah ini

```

public class Tree<E> {

    public Node<E> rootNode = new Node<>();
    public int treeSize = 1;

    public Tree(E data) {
        this.rootNode.setData(data);
    }
}

```

```

public int size(){
    return this.treeSize;
}

public void addChild(E parentData, Node<E> child){
    Node<E> parent = getNode(parentData);
    parent.addChild(child);
    this.treeSize++;
}

public void addChild(E parentData, E dataChild){
    Node<E> newChild = new Node<>(dataChild);
    Node<E> parent = getNode(parentData);
    parent.addChild(newChild);
    this.treeSize++;
}

public Node<E> getNode(E data){
    return preOrderGetNode(this.rootNode, data);
}

public void draw(){
    this.preOrderDraw(this.rootNode, 0);
}

private Node<E> preOrderGetNode(Node<E> node, E data){
    if(node.getData() == data){
        return node;
    } else if(node.isLeaf() == false){
        for(int i=0;i<node.getNumChild();i++){
            Node<E> child = (Node<E>)
node.getChildren().get(i);
            Node<E> returnNode = preOrderGetNode(child, data);
            if(returnNode != null)
                return returnNode;
        }
    }
    return null;
}

private void preOrderDraw(Node<E> node, int depth){
    for(int i=0;i<depth;i++){
        System.out.print("--");
    }
    System.out.println(" " + node.getData());
    for(int i=0;i<node.getNumChild();i++){

```

```

        Node<E> child = (Node<E>) node.getChildren().get(i);
        preOrderDraw(child, depth+1);
    }
}
}

```

#### D. Membuat dan Menjalankan Main Method

1. Silakan buat sebuah main method class dengan nama **Main**
2. Kemudian salin tempel kode program di bawah ini

```

public class Main {

    public static void main(String[] args) {

        Tree<String> pohonKu = new Tree<>("Parent");

        pohonKu.addChild("Parent", "Child 1");
        pohonKu.addChild("Parent", "Child 2");

        Node<String> childNode3 = new Node<>("Child 3");
        pohonKu.addChild("Parent", childNode3);

        pohonKu.addChild("Child 1", "Grandchild 1");

        Node<String> grandchildNode = new Node<>("Grandchild 2");
        pohonKu.addChild("Child 3", grandchildNode);

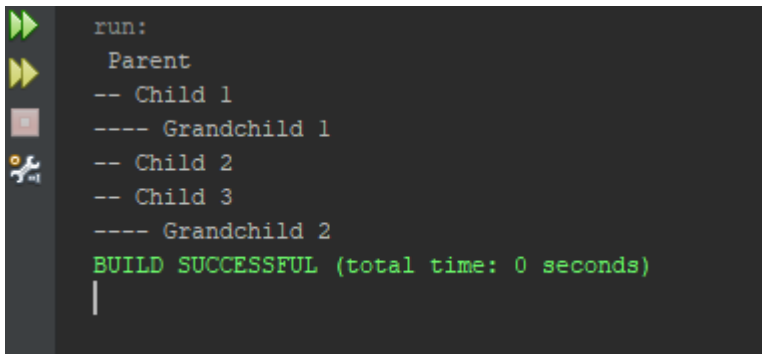
        pohonKu.draw();

    }

}

```

3. Jalankan *main method* tersebut dan hasil tangkapan layar keluaran dari program silakan letakkan di bawah ini



```

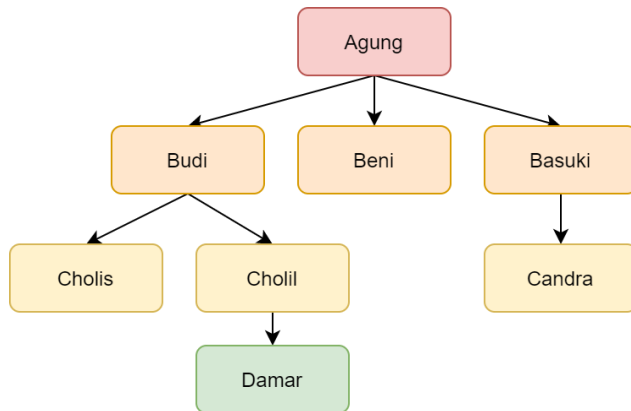
run:
  Parent
  -- Child 1
  ---- Grandchild 1
  -- Child 2
  -- Child 3
  ---- Grandchild 2
BUILD SUCCESSFUL (total time: 0 seconds)

```



## E. Membuat Tree

1. Dengan menggunakan class Tree di atas, buatlah objek Tree yang merepresentasikan pohon keluarga berikut di Main.java:



2. Tuliskan kode yang digunakan untuk membuat objek tersebut:

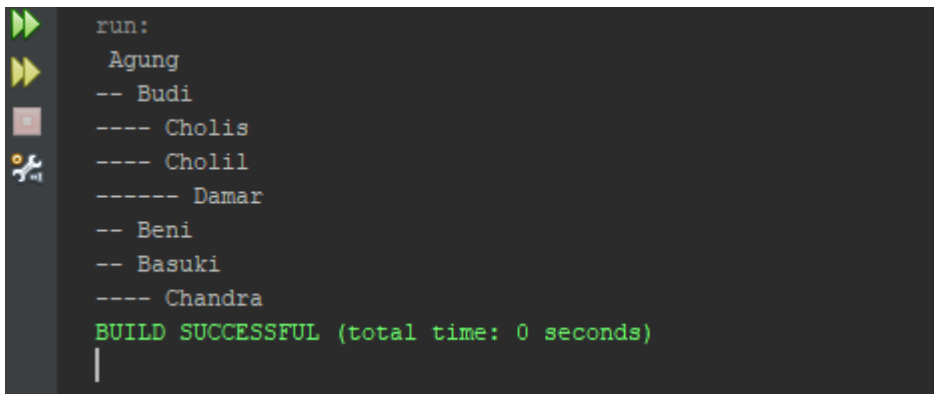
```
Tree<String> nama = new Tree <>("Agung");

nama.addChild("Agung", "Budi");
nama.addChild("Agung", "Beni");
Node<String> anak3 = new Node<>("Basuki");
nama.addChild("Agung", anak3);

nama.addChild("Budi", "Cholis");
Node<String> anakBudi2 = new Node<>("Cholil");
nama.addChild("Budi", anakBudi2);

nama.addChild("Cholil", "Damar");
nama.addChild("Basuki", "Chandra");
```

3. Tambahkan void draw() untuk menampilkan Tree, lalu jalankan program tersebut. Hasil tangkapan layar keluaran dari program silakan letakkan di bawah ini



```
run:
  Agung
  -- Budi
  ---- Cholis
  ---- Cholil
  ----- Damar
  -- Beni
  -- Basuki
  ---- Chandra
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Tambahkan kode program Anda untuk menampilkan anak-anaknya “Budi”:

```
nama.addChild("Budi", "Cholis");
Node<String> anakBudi2 = new Node<>("Cholil");
nama.addChild("Budi", anakBudi2);
```

**\*Catatan**

- Jangan lupa simpan juga fail worksheet ini (yang sudah diisi) sebagai fail pdf di folder NIM anda.
- Sertakan juga fail **Main.java** di dalam folder yang Anda gunakan
- Kompres folder ini sebagai fail ZIP kemudian kumpulkan di classroom atau ruang pengumpulan lain di kelas masing-masing.