POLITEKNIK NEGERI MALANG

TEKNOLOGI INFORMASI

TEKNIK INFORMATIKA

Mohammad Ariq Baihaqi

244107020161

TI – 1A

16

14.2.1 Percobaan 1

Class Mahasiswa16

```
package Minggu15;
public class Mahasiswa16 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa16() {

    }

    public Mahasiswa16(String nim, String nama, String kelas, double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    public void tampilInformasi() {
        System.out.println("NIM:" + this.nim+" " +
                "Nama: " + this.nama + " " +
                "Kelas: " + this.kelas + " " +
                "IPK: " + this.ipk);
    }

}
```

Class Node16

```
package Minggu15;
public class Node16 {
    Mahasiswa16 mahasiswa16;
    Node16 left, right;

    public Node16() {

    }

    public Node16(Mahasiswa16 mahasiswa16) {
        this.mahasiswa16 = mahasiswa16;
        left = right = null;
    }
```

Class BinaryTree16

```
package Minggu15;
public class BinaryTree16 {
    Node16 root;

    public BinaryTree16() {
        root = null;
    }

    public boolean isEmpty() {
        return root == null;
    }

    public void add(Mahasiswa16 mahasiswa) {
        Node16 newNode = new Node16(mahasiswa);
        if (isEmpty()) {
            root = newNode;
        } else {
            Node16 current = root;
            Node16 parent = null;
            while (true) {
```

```
parent = current;
                if (mahasiswa.ipk < current.mahasiswa16.ipk) {
                    current = current.left;
                    if (current == null) {
                        parent.left = newNode;
                        return;
                    }
                } else {
                    current = current.right;
                    if (current == null) {
                        parent.right = newNode;
                        return;
                    }
                }
            }
        }
    }

    public boolean find(double ipk) {
        boolean result = false;
        Node16 current = root;
        while (current != null) {
            if (current.mahasiswa16.ipk == ipk) {
                result = true;
                break;
            } else if (ipk > current.mahasiswa16.ipk) {
                current = current.right;
            } else {
                current = current.left;
            }
        }
        return result;
    }

    public void traversePreOrder(Node16 node) {
        if (node != null) {
            node.mahasiswa16.tampilInformasi();
            traversePreOrder(node.left);
            traversePreOrder(node.right);
        }
    }
```

```
Node16 getSuccessor(Node16 del) {
        Node16 successor = del.right;
        Node16 successorParent = del;
        while (successor.left != null) {
            successorParent = successor;
            successor = successor.left;
        }
        if (successor != del.right) {
            successorParent.left = successor.right;
            successor.right = del.right;
        }
        return successor;
    }

     public void delete(double ipk) {
        if (isEmpty()) {
            System.out.println("x:Binary tree kosong");
            return;
        }

        // cari node (current) yang akan dihapus
        Node16 parent = root;
        Node16 current = root;
        boolean isLeftChild = false;
        while (current != null) {
            if (current.mahasiswa16.ipk == ipk) {
                break;
            } else if (ipk < current.mahasiswa16.ipk) {
                parent = current;
                current = current.left;
                isLeftChild = true;
            } else if (ipk > current.mahasiswa16.ipk) {
                parent = current;
                current = current.right;
                isLeftChild = false;
            }
        }
    //penghapusan
        if (current == null) {
            System.out.println("x:Data tidak ditemukan");
            return;
        } else {
            //jika tidak ada anak (leaf), maka node dihapus
```

```
if (current.left == null && current.right == null) {
                if (current == root) {
                    root = null;
                } else {
                    if (isLeftChild) {
                        parent.left = null;
                    } else {
                        parent.right = null;
                    }
                }
            }
            //jika hanya punya 1 anak (kanan)
            else if (current.left == null) {
                if (current == root) {
                    root = current.right;
                } else {
                    if (isLeftChild) {
                        parent.left = current.right;
                    } else {
                        parent.right = current.right;
                    }
                }
            }
            //jika hanya punya 1 anak (kiri)
            else if (current.right == null) {
                if (current == root) {
                    root = current.left;
                } else {
                    if (isLeftChild) {
                        parent.left = current.left;
                    } else {
                        parent.right = current.left;
                    }
                }
            }
            //jika punya 2 anak
            else {
                Node16 successor = getSuccessor(current);
                System.out.println("x:Jika 2 anak, current = ");
                current.mahasiswa16.tampilInformasi();
                if (current == root) {
                    root = successor;
                } else {
```

```
Node16 successor = getSuccessor(current);
                System.out.println("x:Jika 2 anak, current = ");
                current.mahasiswa16.tampilInformasi();
                if (current == root) {
                    root = successor;
                } else {
                    if (isLeftChild) {
                        parent.left = successor;
                    } else {
                        parent.right = successor;
                    }
                }
                successor.left = current.left;
            }
        }
    }
}
```

Class BinarTreeMain16

```
package Minggu15;
public class BinaryTreeMain16 {
   public static void main(String[] args) {
      BinaryTree16 bst = new BinaryTree16();

      bst.add(new Mahasiswa16("244160121", "Ali", "kelas:A", 3.57));
      bst.add(new Mahasiswa16("244160221", "Badar", "kelas:B", 3.85));
      bst.add(new Mahasiswa16("244160185", "Candra", "kelas:C", 3.21));
      bst.add(new Mahasiswa16("244160201", "Dewi", "kelas:B", 3.54));

      System.out.println("\nx:Daftar semua mahasiswa (in order traversal):");
      bst.traverseInOrder(bst.root);

      System.out.println("\nx:Pencarian data mahasiswa:");
      System.out.print("x:Cari mahasiswa dengan ipk: 3.54 : ");
      String hasilCari = bst.find(3.54) ? "Ditemukan" : "Tidak ditemukan";
      System.out.println(hasilCari);
```

```
System.out.print("x:Cari mahasiswa dengan ipk: 3.22 : ");
        hasilCari = bst.find(3.22) ? "Ditemukan" : "Tidak ditemukan";
        System.out.println(hasilCari);

        bst.add(new Mahasiswa16("244160131", "Devi", "kelas:A", 3.72));
        bst.add(new Mahasiswa16("244160205", "Ehlan", "kelas:D", 3.37));
        bst.add(new Mahasiswa16("244160170", "Fizi", "kelas:B", 3.46));

        System.out.println("\nx:Daftar semua mahasiswa setelah penambahan 3 mahasiswa:");
        System.out.println("x:InOrder Traversal:");
        bst.traverseInOrder(bst.root);
        System.out.println("x:PreOrder Traversal:");
        bst.traversePreOrder(bst.root);
        System.out.println("x:PostOrder Traversal:");
        bst.traversePostOrder(bst.root);

        System.out.println("\nx:Penghapusan data mahasiswa:");
        bst.delete(3.57);

        System.out.println("\nx:Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order
traversal):");
        bst.traverseInOrder(bst.root);
    }
}
```

OUTPUT

```
x:Daftar semua mahasiswa (in order traversal):
NIM:244160185 Nama: Candra Kelas: kelas:C IPK: 3.21
NIM:244160201 Nama: Dewi Kelas: kelas:B IPK: 3.54
NIM:244160121 Nama: Ali Kelas: kelas:A IPK: 3.57
NIM:244160221 Nama: Badar Kelas: kelas:B IPK: 3.85

x:Pencarian data mahasiswa:
x:Cari mahasiswa dengan ipk: 3.54 : Ditemukan
x:Cari mahasiswa dengan ipk: 3.22 : Tidak ditemukan

x:Daftar semua mahasiswa setelah penambahan 3 mahasiswa:
```

```
x:InOrder Traversal:
NIM:244160185 Nama: Candra Kelas: kelas:C IPK: 3.21
NIM:244160205 Nama: Ehlan Kelas: kelas:D IPK: 3.37
NIM:244160170 Nama: Fizi Kelas: kelas:B IPK: 3.46
NIM:244160201 Nama: Dewi Kelas: kelas:B IPK: 3.54
NIM:244160121 Nama: Ali Kelas: kelas:A IPK: 3.57
NIM:244160131 Nama: Devi Kelas: kelas:A IPK: 3.72
NIM:244160221 Nama: Badar Kelas: kelas:B IPK: 3.85
```

```
x:PreOrder Traversal:
NIM:244160121 Nama: Ali Kelas: kelas:A IPK: 3.57
NIM:244160185 Nama: Candra Kelas: kelas:C IPK: 3.21
NIM:244160201 Nama: Dewi Kelas: kelas:B IPK: 3.54
NIM:244160205 Nama: Ehlan Kelas: kelas:D IPK: 3.37
NIM:244160170 Nama: Fizi Kelas: kelas:B IPK: 3.46
NIM:244160221 Nama: Badar Kelas: kelas:B IPK: 3.85
NIM:244160131 Nama: Devi Kelas: kelas:A IPK: 3.72
x:PostOrder Traversal:
NIM:244160170 Nama: Fizi Kelas: kelas:B IPK: 3.46
NIM:244160205 Nama: Ehlan Kelas: kelas:D IPK: 3.37
NIM:244160201 Nama: Dewi Kelas: kelas:B IPK: 3.54
NIM:244160185 Nama: Candra Kelas: kelas:C IPK: 3.21
NIM:244160131 Nama: Devi Kelas: kelas:A IPK: 3.72
NIM:244160221 Nama: Badar Kelas: kelas:B IPK: 3.85
NIM:244160121 Nama: Ali Kelas: kelas:A IPK: 3.57
```

```
x:Penghapusan data mahasiswa:
x:Jika 2 anak, current =
NIM:244160121 Nama: Ali Kelas: kelas:A IPK: 3.57

x:Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):
NIM:244160185 Nama: Candra Kelas: kelas:C IPK: 3.21
NIM:244160205 Nama: Ehlan Kelas: kelas:D IPK: 3.37
NIM:244160170 Nama: Fizi Kelas: kelas:B IPK: 3.46
NIM:244160201 Nama: Dewi Kelas: kelas:B IPK: 3.54
NIM:244160131 Nama: Devi Kelas: kelas:A IPK: 3.72
NIM:244160221 Nama: Badar Kelas: kelas:B IPK: 3.85
```

14.2.2 Pertanyaan Percobaan

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

- karena adanya properti pengurutan yang ketat pada BST.

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

- Membangun Struktur pohon, Navigasi/Traversal Pohon

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

- Titik masuk ke pohon, penentu kondisi kosong

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

- Bukan, untuk objek BinaryTree pertama kali dibuat, nilai dari atribut root adalah null

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

- Proses Add, untuk menambahkan node baru

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

```
parent = current;
if (mahasiswa.ipk < current.mahasiswa.ipk) {
    current = current.left;
    if (current == null) {
        parent.left = newNode;
        return;
    }
} else {
    current = current.right;
    if (current == null) {
        parent.right = newNode;
        return;
    }
}
```

- Digunakan untuk menambahkan node bar uke dalam Binary Search Tree(BST)

6. Jelaskan langkah-langkah pada method delete() saat menghapus sebuah node yang memiliki dua anak. Bagaimana method getSuccessor() membantu dalam proses ini?

- Ketika sebuah node (current) yang akan dihapus memiliki dua anak (baik current.left maupun current.right tidak null), algoritma umumnya melibatkan penggantian node yang akan dihapus dengan "successor"-nya.

14.3.1 Tahapan Percobaan

Class BinaryTree16

```
package Minggu15;

public class BinaryTreeArray16 {

    Mahasiswa16[] dataMahasiswa;

    int idxLast;


    public BinaryTreeArray16() {

        this.dataMahasiswa = new Mahasiswa16[10];

    }


    void populateData(Mahasiswa16 dataMhs[], int idxLast) {

        this.dataMahasiswa = dataMhs;

        this.idxLast = idxLast;

    }
```

Class BinaryTreeArrayMain16

```
package Minggu15;
public class BinaryTreeArrayMain16 {
    public static void main(String[] args) {
        BinaryTreeArray16 bta = new BinaryTreeArray16();

        Mahasiswa16 mhs1 = new Mahasiswa16("244160121", "Ali", "kelas:A", 3.57);
        Mahasiswa16 mhs2 = new Mahasiswa16("244160185", "Candra", "kelas:C", 3.41);
        Mahasiswa16 mhs3 = new Mahasiswa16("244160221", "Badar", "kelas:B", 3.75);
        Mahasiswa16 mhs4 = new Mahasiswa16("244160220", "Dewi", "kelas:B", 3.35);

        Mahasiswa16 mhs5 = new Mahasiswa16("244160131", "Devi", "kelas:A", 3.48);
        Mahasiswa16 mhs6 = new Mahasiswa16("244160205", "Ehlan", "kelas:D", 3.61);
        Mahasiswa16 mhs7 = new Mahasiswa16("244160170", "Fizi", "kelas:B", 3.86);

        Mahasiswa16[] dataMahasiswas = {mhs1, mhs2, mhs3, mhs4, mhs5, mhs6, mhs7,
null, null, null};
        int idxLast = 6;

        bta.populateData(dataMahasiswas, idxLast);

        System.out.println("x:\\nInorder Traversal Mahasiswa: ");
        bta.traverseInOrder(0);
    }
}
```

OUTPUT

```
x:\nInorder Traversal Mahasiswa:
NIM:244160220 Nama: Dewi Kelas: kelas:B IPK: 3.35
NIM:244160185 Nama: Candra Kelas: kelas:C IPK: 3.41
NIM:244160131 Nama: Devi Kelas: kelas:A IPK: 3.48
NIM:244160121 Nama: Ali Kelas: kelas:A IPK: 3.57
NIM:244160205 Nama: Ehlan Kelas: kelas:D IPK: 3.61
NIM:244160221 Nama: Badar Kelas: kelas:B IPK: 3.75
NIM:244160170 Nama: Fizi Kelas: kelas:B IPK: 3.86
```

14.3.2 Pertanyaan Percobaan

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

- digunakan untuk menyimpan node-node dari Binary Tree.

2. Apakah kegunaan dari method populateData()?

- digunakan untuk menginisialisasi atau mengisi array dataMahasiswa dan mengatur nilai idxLast

3. Apakah kegunaan dari method traverseInOrder()?

- untuk menjelajahi (traversal) Binary Tree dengan urutan In-Order.

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

jika suatu node berada di **indeks 2**:

- Posisi left child adalah 2×2+1=4+1=indeks 5.
- Posisi right child adalah 2×2+2=4+2=indeks 6.

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

- untuk menentukan batas akhir data yang valid dalam array

6. Mengapa indeks `2*idxStart+1` dan `2*idxStart+2` digunakan dalam pemanggilan rekursif, dan apa kaitannya dengan struktur pohon biner yang disusun dalam array?

- karena mereka adalah rumus standar untuk merepresentasikan struktur pohon biner lengkap dalam sebuah array

14.4 Tugas Praktikum

1. Buat method di dalam class BinaryTree00 yang akan menambahkan node dengan cara rekursif (addRekursif()).

2. Buat method di dalam class BinaryTree00 untuk menampilkan data mahasiswa dengan IPK paling kecil dan IPK yang paling besar (cariMinIPK() dan cariMaxIPK()) yang ada di dalam binary search tree.

3. Buat method dalam class BinaryTree00 untuk menampilkan data mahasiswa dengan IPK di atas suatu batas tertentu, misal di atas 3.50 (tampilMahasiswaIPKdiAtas(double ipkBatas)) yang ada di dalam binary search tree.

4. Modifikasi class BinaryTreeArray00 di atas, dan tambahkan :

• method add(Mahasiswa data) untuk memasukan data ke dalam binary tree

• method traversePreOrder()

Class BinaryTree16

```
public void addRekursif(Mahasiswa16 mahasiswa) {
        root = addRekursif(root, mahasiswa);
    }

    private Node16 addRekursif(Node16 current, Mahasiswa16 mahasiswa) {
        if (current == null) {
            return new Node16(mahasiswa);
        }

        if (mahasiswa.ipk < current.mahasiswa16.ipk) {
            current.left = addRekursif(current.left, mahasiswa);
        } else if (mahasiswa.ipk > current.mahasiswa16.ipk) {
            current.right = addRekursif(current.right, mahasiswa);
        } else {
            System.out.println("x:Mahasiswa dengan IPK " + mahasiswa.ipk + " sudah
ada, tidak ditambahkan.");
        }
        return current;
    }
```

```
public Mahasiswa16 cariMinIPK() {
        if (isEmpty()) {
            System.out.println("x:Binary tree kosong, tidak ada IPK minimum.");
            return null;
        }
        Node16 current = root;
        while (current.left != null) {
            current = current.left;
        }
        return current.mahasiswa16;
    }

     public Mahasiswa16 cariMaxIPK() {
        if (isEmpty()) {
            System.out.println("x:Binary tree kosong, tidak ada IPK maksimum.");
            return null;
        }
        Node16 current = root;
        while (current.right != null) {
            current = current.right;
        }
        return current.mahasiswa16;
    }

public void tampilMahasiswaIPKdiAtas(double ipkBatas) {
        System.out.println("\nx:Mahasiswa dengan IPK di atas " + ipkBatas + ":");
        tampilMahasiswaIPKdiAtas(root, ipkBatas);
    }

    private void tampilMahasiswaIPKdiAtas(Node16 node, double ipkBatas) {
        if (node == null) {
            return;
        }
    }
}
```

Class BinaryTreeArray

```
public void add(Mahasiswa16 data) {
        if (idxLast == dataMahasiswa.length - 1) {
            System.out.println("x:Array pohon penuh, tidak bisa menambahkan data
lagi.");
            return;
        }
        if (idxLast == -1) {
            dataMahasiswa[0] = data;
            idxLast = 0;
            return;
        }
    }

public void traversePreOrder() {
        System.out.println("x:PreOrder Traversal (Array):");
        traversePreOrder(0);
    }

    private void traversePreOrder(int idxStart) {
        if (idxStart <= idxLast) {
            if (dataMahasiswa[idxStart] != null) {
                dataMahasiswa[idxStart].tampilInformasi();
                traversePreOrder(2 * idxStart + 1);
                traversePreOrder(2 * idxStart + 2);
            }
        }
    }
}
```

Link Github : https://github.com/Ariqq16/semester2