Nama : Desy Dwi Puspita
Kelas : SIB1A
NIM : 244107060145
Jobsheet 14 - Tree

Praktikum 1 - Implementasi Binary Search Tree menggunakan Linked List
Kode program :

- Class Mahasiswa03

```java
Pertemuan14 > J Mahasiswa03.java > ...
package Pertemuan14;

public class Mahasiswa03 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa03() {
    }

    public Mahasiswa03(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 Node03

```java
Pertemuan14 > J Node03.java > ...
package Pertemuan14;

public class Node03 {
    Mahasiswa03 mahasiswa;
    Node03 left, right;

    public Node03() {
    }

    public Node03(Mahasiswa03 mahasiswa) {
        this.mahasiswa = mahasiswa;
        this.left = null;
    }
}
```

- Class BinaryTree03

```
Pertemuan14 > BinaryTree03.java > BinaryTree03 > find(double)
```

```java
package Pertemuan14;

public class BinaryTree03 {
    Node03 root;

    public BinaryTree03() {
        root = null;
    }

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

    public void add(Mahasiswa03 mahasiswa) {
        Node03 newNode = new Node03(mahasiswa);
        if (isEmpty()) {
            root = newNode;
        } else {
            Node03 current = root;
            Node03 parent = null;
            while (true) {
                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;
                    }
                }
            }
        }
    }
```

```java
    boolean find(double ipk) {
        boolean result = false;
        Node03 current = root;
        while (current != null) {
            if (current.mahasiswa.ipk == ipk) {
                result = true;
                break;
            } else if (ipk > current.mahasiswa.ipk) {
                current = current.right;
            } else {
                current = current.left;
            }
        }
        return result;
    }

    void traversePreOrder(Node03 node) {
        if (node != null) {
            node.mahasiswa.tampilInformasi();
            traversePreOrder(node.left);
            traversePreOrder(node.right);
        }
    }

    void traverseInOrder(Node03 node) {
        if (node != null) {
            traverseInOrder(node.left);
            node.mahasiswa.tampilInformasi();
            traverseInOrder(node.right);
        }
    }

    void traversePostOrder(Node03 node) {
        if (node != null) {
            traversePostOrder(node.left);
            traversePostOrder(node.right);
            node.mahasiswa.tampilInformasi();
```

```java
    Node03 getSuccessor(Node03 del) {
        Node03 successor = del.right;
        Node03 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;
    }
```

```java
    void delete(double ipk) {
        if (isEmpty()) {
            System.out.println(x:"Binary tree kosong");
            return;
        }
        // cari node (current) yang akan dihapus
        Node03 parent = root;
        Node03 current = root;
        boolean isLeftChild = false;
        while (current != null) {
            if (current.mahasiswa.ipk == ipk) {
                break;
            } else if (ipk < current.mahasiswa.ipk) {
                parent = current;
                current = current.left;
                isLeftChild = true;
            } else if (ipk > current.mahasiswa.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 {
                Node03 successor = getSuccessor(current);
                System.out.println(x:"Jika 2 anak, current = ");
                successor.mahasiswa.tampilInformasi();
                if (current == root) {
                    root = successor;
                } else {
                    if (isLeftChild) {
                        parent.left = successor;
                    } else {
                        parent.right = successor;
                    }
                }
                successor.left = current.left;
            }
        }
    }
}
```

- Class BinaryTreeMain03

```
public class BinaryTreeMain03 {
    Run | Debug
    public static void main(String[] args) {
        BinaryTree03 bst = new BinaryTree03();

        bst.add(new Mahasiswa03(nim:"244160121", nama:"Ali", kelas:"A", ipk:3.57));
        bst.add(new Mahasiswa03(nim:"244160221", nama:"Badar", kelas:"B", ipk:3.85));
        bst.add(new Mahasiswa03(nim:"244160185", nama:"Candra", kelas:"C", ipk:3.21));
        bst.add(new Mahasiswa03(nim:"244160220", nama:"Dewi", kelas:"B", ipk:3.54));

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

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

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

        bst.add(new Mahasiswa03(nim:"244160131", nama:"Devi", kelas:"A", ipk:3.72));
        bst.add(new Mahasiswa03(nim:"244160205", nama:"Ehsan", kelas:"D", ipk:3.37));
        bst.add(new Mahasiswa03(nim:"244160170", nama:"Fizi", kelas:"B", ipk:3.46));
        System.out.println(x:"\nDaftar semua mahasiswa setelah penambahan 3 mahasiswa:");
        System.out.println(x:"InOrder Traversal:");
        bst.traverseInOrder(bst.root);
        System.out.println(x:"\nPreOrder Traversal:");
        bst.traversePreOrder(bst.root);
        System.out.println(x:"\nPostOrder Traversal:");
        bst.traversePostOrder(bst.root);

        System.out.println(x:"\nPenghapusan data mahasiswa");
        bst.delete(ipk:3.57);
        System.out.println(x:"\nDaftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):");
        bst.traverseInOrder(bst.root);
    }
}
```

Output :

```
Daftar semua mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85

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

Daftar semua mahasiswa setelah penambahan 3 mahasiswa:
InOrder Traversal:
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85

PreOrder Traversal:
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72

PostOrder Traversal:
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
```

```
Penghapusan data mahasiswa
Jika 2 anak, current =
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72

Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
```

Pertanyaan percobaan :

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
   karena setiap node disusun dengan aturan nilai yang lebih kecil berada di kiri dan yang lebih besar di kanan, sehingga pencarian dapat langsung mengarah ke subtree yang relevan tanpa harus memeriksa semua node.
2. Untuk apakah di class **Node**, kegunaan dari atribut **left** dan **right**?
   Atribut left dan right digunakan untuk menunjuk ke anak kiri dan kanan, sehingga membentuk struktur pohon biner dan memungkinkan traversal atau manipulasi data
3. a. Untuk apakah kegunaan dari atribut **root** di dalam class **BinaryTree**?
   Atribut root digunakan sebagai titik awal (akar) untuk semua operasi pada pohon, seperti penambahan, pencarian, atau penghapusan data.
   b. Ketika objek tree pertama kali dibuat, apakah nilai dari **root**?
   Ketika objek tree pertama kali dibuat, nilai root adalah null karena pohon masih kosong.
4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
   Ketika tree masih kosong dan akan ditambahkan sebuah node baru, proses yang terjadi adalah node baru tersebut langsung menjadi root dari tree, karena belum ada node lain sebagai akar atau anak.
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;
    }
}
```

   Baris tersebut mencari posisi yang sesuai dalam tree berdasarkan IPK. Jika posisi kiri atau kanan kosong, node baru ditambahkan sebagai anak dari parent.
6. Jelaskan langkah-langkah pada method **delete()** saat menghapus sebuah node yang memiliki dua anak. Bagaimana method **getSuccessor()** membantu dalam proses ini?
   Jika node yang dihapus punya dua anak, maka dicari successor (node terkecil di subtree kanan) untuk menggantikannya. Method getSuccessor() menemukan node ini dan mengatur ulang koneksi agar struktur tree tetap benar.

Praktikum 2 - Implementasi Binary Tree dengan Array

Kode Program :

- Class BinaryTreeArray03

```java
package Pertemuan14;

public class BinaryTreeArray03 {
    Mahasiswa03[] dataMahasiswa;
    int idxLast;

    public BinaryTreeArray03() {
        this.dataMahasiswa = new Mahasiswa03[10];
    }

    void populateData (Mahasiswa03 datMhs[], int idxLast) {
        this.dataMahasiswa = datMhs;
        this.idxLast = idxLast;
    }

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

- Class BinaryTreeArrayMain03

```java
package Pertemuan14;

public class BinaryTreeArrayMain03 {
    Run | Debug
    public static void main(String[] args) {
        BinaryTreeArray03 bta = new BinaryTreeArray03();

        Mahasiswa03 mhs1 = new Mahasiswa03(nim:"244160121", nama:"Ali", kelas:"A", ipk:3.57);
        Mahasiswa03 mhs2 = new Mahasiswa03(nim:"244160185", nama:"Candra", kelas:"C", ipk:3.41);
        Mahasiswa03 mhs3 = new Mahasiswa03(nim:"244160221", nama:"Badar", kelas:"B", ipk:3.75);
        Mahasiswa03 mhs4 = new Mahasiswa03(nim:"244160220", nama:"Dewi", kelas:"B", ipk:3.35);

        Mahasiswa03 mhs5 = new Mahasiswa03(nim:"244160131", nama:"Devi", kelas:"A", ipk:3.48);
        Mahasiswa03 mhs6 = new Mahasiswa03(nim:"244160205", nama:"Ehsan", kelas:"D", ipk:3.61);
        Mahasiswa03 mhs7 = new Mahasiswa03(nim:"244160170", nama:"Fizi", kelas:"B", ipk:3.86);

        Mahasiswa03[] dataMahasiswa = {mhs1, mhs2, mhs3, mhs4, mhs5, mhs6, mhs7, null, null, null};
        int idxLast = 6;
        bta.populateData(dataMahasiswa, idxLast);
        System.out.println(x:"\nInorder Traversal Mahasiswa: ");
        bta.traverseInOrder(idxStart:0);
    }
}
```

Output :

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

Pertanyaan percobaan :

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class **BinaryTreeArray**?
   Data digunakan untuk menyimpan data mahasiswa dalam bentuk struktur pohon biner. idxLast digunakan untuk menunjukkan indeks terakhir dari data yang valid di dalam array tersebut.
2. Apakah kegunaan dari method **populateData()**?
   berfungsi untuk mengisi array dataMahasiswa dengan data mahasiswa yang diberikan dan menetapkan nilai idxLast sebagai batas data yang valid.
3. Apakah kegunaan dari method **traverseInOrder()**?
   Method ini melakukan penelusuran data mahasiswa secara in-order pada pohon biner yang direpresentasikan oleh array, yaitu mengunjungi anak kiri, menampilkan data saat ini, lalu anak kanan secara rekursif.
4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan right child masing-masing?
   Left child berada di indeks 2 * 2 + 1 = 5
   Right child berada di indeks 2 * 2 + 2 = 6
5. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?
   berfungsi untuk menetapkan batas indeks terakhir data mahasiswa yang valid dalam array dataMahasiswa atau data yang valid hanya sampai indeks ke-6, sehingga proses traversal atau operasi lain tidak akan mengakses indeks di luar batas ini.
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?
   Indeks 2*idxStart+1 dan 2*idxStart+2 digunakan karena dalam penyimpanan pohon biner pada array, anak kiri selalu berada di posisi 2*idxStart+1 dan anak kanan di posisi 2*idxStart+2. Cara ini memudahkan untuk mengakses anak-anak node hanya dengan perhitungan indeks, sesuai struktur pohon biner tanpa perlu pointer.

Tugas Praktikum

1. Buat method di dalam class BinaryTree03 yang akan menambahkan node dengan cara rekursif (addRekursif()).

```
181     public void addRekursif(Mahasiswa03 mahasiswa) {
182         root = addRekursif(root, mahasiswa);
183     }
184
185     private Node03 addRekursif(Node03 current, Mahasiswa03 mahasiswa) {
186         if (current == null) {
187             return new Node03(mahasiswa);
188         }
189         if (mahasiswa.ipk < current.mahasiswa.ipk) {
190             current.left = addRekursif(current.left, mahasiswa);
191         } else {
192             current.right = addRekursif(current.right, mahasiswa);
193         }
194         return current;
195     }
```

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.

```
197     public void cariMinIPK() {
198         if (isEmpty()) {
199             System.out.println(x:"Tree kosong");
200             return;
201         }
202         Node03 current = root;
203         while (current.left != null) {
204             current = current.left;
205         }
206         current.mahasiswa.tampilInformasi();
207     }
208
209     public void cariMaxIPK() {
210         if (isEmpty()) {
211             System.out.println(x:"Tree kosong");
212             return;
213         }
214         Node03 current = root;
215         while (current.right != null) {
216             current = current.right;
217         }
218         current.mahasiswa.tampilInformasi();
219     }
```

Main :

```
36          // Tambahan: Menampilkan IPK terkecil dan terbesar
37          System.out.println(x:"\nMahasiswa dengan IPK terkecil:");
38          bst.cariMinIPK();
39
40          System.out.println(x:"\nMahasiswa dengan IPK terbesar:");
41          bst.cariMaxIPK();
```

Output :

```
Mahasiswa dengan IPK terkecil:
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21

Mahasiswa dengan IPK terbesar:
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
```

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.

```java
public void tampilMahasiswaIPKdiAtas(double ipkBatas) {
    tampilMahasiswaIPKdiAtas(root, ipkBatas);
}

private void tampilMahasiswaIPKdiAtas(Node03 node, double ipkBatas) {
    if (node != null) {
        tampilMahasiswaIPKdiAtas(node.left, ipkBatas);
        if (node.mahasiswa.ipk > ipkBatas) {
            node.mahasiswa.tampilInformasi();
        }
        tampilMahasiswaIPKdiAtas(node.right, ipkBatas);
    }
}
```

Main

```java
// Tambahan: Menampilkan mahasiswa dengan IPK di atas 3.50
System.out.println(x:"\nMahasiswa dengan IPK di atas 3.50:");
bst.tampilMahasiswaIPKdiAtas(ipkBatas:3.50);

System.out.println(x:"\nPenghapusan data mahasiswa (IPK = 3.57)");
bst.delete(ipk:3.57);

System.out.println(x:"\nDaftar semua mahasiswa setelah penghapusan (in order traversal):");
bst.traverseInOrder(bst.root);
```

Output :

```
Mahasiswa dengan IPK di atas 3.50:
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
```

4. Modifikasi class BinaryTreeArray00 di atas, dan tambahkan :
   • method add(Mahasiswa data) untuk memasukan data ke dalam binary tree

```java
//Tambahan tugas no 4 menambahkan method add
void add(Mahasiswa03 data) {
    if (idxLast >= dataMahasiswa.length - 1) {
        System.out.println(x:"Tree Penuh, tidak bisa menambahkan data.");
        return;
    }
    idxLast++;
    dataMahasiswa[idxLast] = data;
}
```

• method traversePreOrder()

```
36      //Tambahan tugas no 4 menambahkan method traversePreOrder()
37      void traversePreOrder(int idxStart) {
38          if (idxStart <= idxLast && dataMahasiswa[idxStart] != null){
39              dataMahasiswa[idxStart].tampilInformasi();
40              traversePreOrder(2 * idxStart + 1);
41              traversePreOrder(2 * idxStart + 2);
42          }
43      }
44  }
```

Main

```
22          // Tugas no 4 Menambahkan mahasiswa baru menggunakan method add()
23          Mahasiswa03 mhs8 = new Mahasiswa03(nim:"244160300", nama:"Desy", kelas:"A", ipk:3.90);
24          bta.add(mhs8);
25
26          // Tugas no 4 Menambahkan mahasiswa baru menggunakan method traversePreOrder()
27          System.out.println(x:"\nSetelah tambah 1 mahasiswa (Desy), Preorder traversal:");
28          bta.traversePreOrder(idxStart:0);
```

Output :

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

Setelah tambah 1 mahasiswa (Desy), Preorder traversal:
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.41
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM: 244160300 Nama: Desy Kelas: D IPK: 3.91
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.48
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.75
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.61
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.86
```