Nama : Zaskia Maulidina Mutiara Hati

NIM : 244107060056

Kelas : 1A

JOBSHEET 13 PRAKTIKUM ALGORITMA STRUKTUR DATA

- **Percobaan 1**
  - **Class Mahasiswa24**

jobsheet_13 > Mahasiswa24.java > Mahasiswa24

```java
package jobsheet_13;

public class Mahasiswa24 {

    String nama;
    String nim;
    String kelas;
    double ipk;

    public Mahasiswa24() {
    }

    public Mahasiswa24(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 Node24**

jobsheet_13 > Node24.java > Node24

```java
package jobsheet_13;

public class Node24 {
    Mahasiswa24 mahasiswa;
    Node24 left, right;

    public Node24() {
    }

    public Node24(Mahasiswa24 mahasiswa) {
        this.mahasiswa = mahasiswa;
        left = right = null;
    }
}
```

- **Class BinaryTree24**

jobsheet_13 > BinaryTree24.java > ...

```java
package jobsheet_13;

public class BinaryTree24 {
    Node24 root;

    public BinaryTree24() {
        root = null;
    }

    public boolean isEmpty() {
        return root == null;
    }
    public void add(Mahasiswa24 mahasiswa) {
        Node24 newNode = new Node24(mahasiswa);
        if (isEmpty()) {
            root = newNode;
        } else {
            Node24 current = root;
            Node24 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;
                    }
                }
            }
        }
    }
```

```
    boolean find(double ipk) {
        boolean result = false;
        Node24 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(Node24 node) {
        if (node != null) {
            node.mahasiswa.tampilInformasi();
            traversePreOrder(node.left);
            traversePreOrder(node.right);
        }
    }

    void traverseInOrder(Node24 node) {
        if (node != null) {
            traverseInOrder(node.left);
            node.mahasiswa.tampilInformasi();
            traverseInOrder(node.right);
        }
    }
```

```
    void traversePostOrder(Node24 node) {
        if (node != null) {
            traversePostOrder(node.left);
            traversePostOrder(node.right);
            node.mahasiswa.tampilInformasi();
        }
    }

    Node24 getSuccessor(Node24 del) {
        Node24 successor = del.right;
        Node24 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;
    }
```

```
94      void delete(double ipk) {
95          if (isEmpty()) {
96              System.out.println(x:"Binary tree kosong");
97              return;
98          }
99          //cari node (current) yang akan dihapus
100         Node24 parent = root;
101         Node24 current = root;
102         boolean isLeftChild = false;
103         while (current != null) {
104             if (current.mahasiswa.ipk == ipk) {
105                 break;
106             } else if (ipk < current.mahasiswa.ipk) {
107                 parent = current;
108                 current = current.left;
109                 isLeftChild = true;
110             } else if (ipk > current.mahasiswa.ipk) {
111                 parent = current;
112                 current = current.right;
113                 isLeftChild = false;
114             }
115         }
```

```
116         //penghapusan
117         if (current == null) {
118             System.out.println(x:"Data tidak ditemukan");
119             return;
120         } else {
121             //jika tidak ada anak (leaf), maka node dihapus
122             if (current.left == null && current.right == null) {
123                 if (current == root) {
124                     root = null;
125                 } else {
126                     if (isLeftChild) {
127                         parent.left = null;
128                     } else {
129                         parent.right = null;
130                     }
131                 }
132             } else if (current.left == null) { //jika hanya punya 1 anak (kanan)
133                 if (current == root) {
134                     root = current.right;
135                 } else {
136                     if (isLeftChild) {
137                         parent.left = current.right;
138                     } else {
139                         parent.right = current.right;
140                     }
141                 }
```

```
142             } else if (current.right == null) { //jika hanya punya 1 anak (kiri)
143                 if (current == root) {
144                     root = current.left;
145                 } else {
146                     if (isLeftChild) {
147                         parent.left = current.left;
148                     } else {
149                         parent.right = current.left;
150                     }
151                 }
152             } else { //jika punya dua anak
153                 Node24 successor = getSuccessor(current);
154                 System.out.println(x:"Jika 2 anak, current = ");
155                 successor.mahasiswa.tampilInformasi();
156                 if (current == root) {
157                     root = successor;
158                 } else {
159                     if( isLeftChild) {
160                         parent.left = successor;
161                     } else {
162                         parent.right = successor;
163                     }
164                 }
165                 successor.left = current.left;
166             }
167         }
168     }
169 }
```

- **Class BinaryTreeMain24**

jobsheet_13 > BinaryTreeMain24.java > BinaryTreeMain24

```
1   package jobsheet_13;
2
3   public class BinaryTreeMain24 {
        Run | Debug
4       public static void main(String[] args) {
5           BinaryTree24 bst = new BinaryTree24();
6
7           bst.add(new Mahasiswa24(nim:"244160121", nama:"Ali", kelas:"A", ipk:3.57));
8           bst.add(new Mahasiswa24(nim:"244160221", nama:"Badar", kelas:"B", ipk:3.85));
9           bst.add(new Mahasiswa24(nim:"244160185", nama:"Candra", kelas:"C", ipk:3.21));
10          bst.add(new Mahasiswa24(nim:"244160220", nama:"Dewi", kelas:"B", ipk:3.54));
11
12          System.out.println(x:"\nDaftar semua mahasiswa (in order transversal) :");
13          bst.traverseInOrder(bst.root);
14
15          System.out.println(x:"\nPencarian data mahasiswa:");
16          System.out.print(s:"Cari mahasiswa dengan IPK 3.54: ");
17          String hasilCari = bst.find(ipk:3.54) ? "Ditemukan" : "Tidak ditemukan";
18          System.out.println(hasilCari);
19
20          System.out.println(x:"Cari mahasiswa dengan IPK 3.22: ");
21          hasilCari = bst.find(ipk:3.22) ? "Ditemukan" : "Tidak ditemukan";
22          System.out.println(hasilCari);
```

```
        bst.add(new Mahasiswa24(nim:"244160131", nama:"Devi", kelas:"A", ipk:3.72));
        bst.add(new Mahasiswa24(nim:"244160205", nama:"Ehsan", kelas:"D", ipk:3.37));
        bst.add(new Mahasiswa24(nim:"244160170", nama:"Fizi", kelas:"B", ipk:3.46));
        System.out.println(x:"\nDaftar semua mahasiswa setelah penambahan 3 mahasiswa:");
        System.out.println(x:"\nInOrder transversal:");
        bst.traverseInOrder(bst.root);
        System.out.println(x:"\nPreOrder transversal:");
        bst.traversePreOrder(bst.root);
        System.out.println(x:"\nPostOrder transversal:");
        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 transversal) :
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 transversal:
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 transversal:
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 transversal:
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 1**

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
   - Karena pada binary search tree data disusun secara terurut, data yang nilai nya lebih kecil dari node induknya selalu berada di sebelah kiri, dan data yang lebih besar dari node induknya selalu berada di sebelah kanan, sehingga membuat proses pencarian tidak perlu mengecek ke-semua node, cukup membandingkan dan memilih arah yang benar (kanan, kiri).

2. Untuk apakah di class Node, kegunaan dari atribut left dan right?
   - Atribut left dan right di class Node berfungsi untuk menyambungkan node satu dengan node lainnya supaya bisa membentuk struktur pohon. Tanpa adanya atribut ini, node nya akan berdiri sendiri, tidak mempunyai hubungan, dan tidak bisa disebut Tree.

3. a. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?
   - Root adalah penanda node paling atas (awal) dalam binary tree. Fungsi root yaitu sebagai titik awal akses keseluruh pohon dan untuk menyimpan struktur pohon.

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

4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
   - Proses add(), ketika tree masih kosong dan ditambahkan node baru, maka node baru tersebut menjadi rootnya.

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;
    }
}
```

   - `parent = current;` : Menyimpan node yang sekarang ke dalam variable parent

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

   - : Jika ipk mahasiswa baru lebih kecil dari ipk mahasiswa sekarang (current), maka Gerak ke kiri(current = current.left), jika kiri ini kosong (current == null) maka tempat ini cocok untuk masukin node baru.

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

   -

: Jika ipk mahasiswa baru lebih besar dari ipk mahasiswa sekarang (current), maka Gerak ke kanan(current = current.right), jika kanan ini kosong (current == null) maka tempat ini cocok untuk masukin node baru.

6. Jelaskan langkah-langkah pada method delete() saat menghapus sebuah node yang memiliki dua anak. Bagaimana method getSuccessor() membantu dalam proses ini?
   - **Cari successor** → pakai method `getSuccessor(current)`
     (yaitu node terkecil di subtree kanan)
   - **Gantikan node yang dihapus dengan successor**
     (successor dipindah ke posisi node yang dihapus)
   - **Hubungkan anak kiri node lama ke successor**
     (karena successor tidak punya anak kiri, aman disambung)

- **Percobaan 2**
  - **Class BinaryTreeArray24**

```
jobsheet_13 > BinaryTreeArray24.java > BinaryTreeArray24
1   package jobsheet_13;
2
3   public class BinaryTreeArray24 {
4       Mahasiswa24[] dataMahasiswa;
5       int idxLast;
6
7       public BinaryTreeArray24() {
8           this.dataMahasiswa = new Mahasiswa24[10];
9       }
10
11      void populateData (Mahasiswa24 dataMhs[], int idxLast) {
12          this.dataMahasiswa = dataMhs;
13          this.idxLast = idxLast;
14      }
15
16      void traversaInOrder(int idxStart) {
17          if (idxStart <= idxLast) {
18              if (dataMahasiswa[idxStart] != null) {
19                  traversaInOrder(2 * idxStart + 1);
20                  dataMahasiswa[idxStart].tampilInformasi();
21                  traversaInOrder(2 * idxStart + 2); // Kanan
22              }
23          }
24      }
25  }
```

  - **Class BinaryTreeArrayMain24**

```
jobsheet_13 > BinaryTreeArrayMain24.java > BinaryTreeArrayMain24
1   package jobsheet_13;
2
3   public class BinaryTreeArrayMain24 {
        Run | Debug
4       public static void main(String[] args) {
5           BinaryTreeArray24 bta = new BinaryTreeArray24();
6           Mahasiswa24 mhs1 = new Mahasiswa24(nim:"244160121", nama:"Ali", kelas:"A", ipk:3.57);
7           Mahasiswa24 mhs2 = new Mahasiswa24(nim:"244160185", nama:"Candra", kelas:"C", ipk:3.41);
8           Mahasiswa24 mhs3 = new Mahasiswa24(nim:"244160221", nama:"Badar", kelas:"B", ipk:3.75);
9           Mahasiswa24 mhs4 = new Mahasiswa24(nim:"244160220", nama:"Dewi", kelas:"B", ipk:3.35);
10
11          Mahasiswa24 mhs5 = new Mahasiswa24(nim:"244160131", nama:"Devi", kelas:"A", ipk:3.48);
12          Mahasiswa24 mhs6 = new Mahasiswa24(nim:"244160205", nama:"Ehsan", kelas:"D", ipk:3.61);
13          Mahasiswa24 mhs7 = new Mahasiswa24(nim:"244160185", nama:"Fizi", kelas:"B", ipk:3.86);
14
15          Mahasiswa24[] dataMahasiswas = {mhs1, mhs2, mhs3, mhs4, mhs5, mhs6, mhs7, null, null, null};
16          int idxLast = 6;
17          bta.populateData(dataMahasiswas, idxLast);
18          System.out.println(x:"\nInOrder Traversal Mahasiswa:");
19          bta.traversaInOrder(idxStart:0);
20
21      }
22  }
```

- **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:244160185 Nama: Fizi Kelas: B IPK: 3.86
```

- **Pertanyaan 2**

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?
   - Atribut idxLast pada class BinaryTreeArray24 berfungsi untuk menyimpan indeks terakhir (tertinggi) dari elemen array dataMahasiswa yang berisi data (tidak kosong) dalam representasi binary tree berbasis array.

2. Apakah kegunaan dari method populateData()?
   - Fungsi method populateData() untuk mengisi data Tree dari array eksternal dan menandai batas akhir isi tree.

3. Apakah kegunaan dari method traverseInOrder()?
   - Method ini digunakan untuk melakukan traversal in-order pada binary tree yang disimpan dalam bentuk array.

4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masing-masing?
   - Kiri : 2 * 2 + 1 = 5
   - Kanan : 2 * 2 + 2 = 6

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

   - idxLast = 6 menandai indeks terakhir yang berisi data valid dalam array dataMahasiswas. Artinya, dari indeks 0 sampai 6 sudah ada objek Mahasiswa24 (isi data), sedangkan indeks setelah 6 adalah null alias kosong.

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 dipakai karena itu rumus untuk posisi anak kiri dan kanan di binary tree yang disimpan dalam array secara level-order. Jadi, dari node di idxStart, anak kiri ada di 2*idxStart+1 dan anak kanan di 2*idxStart+2. Ini memastikan traversal rekursif sesuai struktur pohon meski pakai array.

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

```
169         //Menambahkan node secara rekursif
170     public void addRekursif(Mahasiswa24 mahasiswa) {
171         root = addRekursifRek(root, mahasiswa);
172     }
173
174     Node24 addRekursifRek(Node24 current, Mahasiswa24 mahasiswa) {
175         if (current == null) {
176             return new Node24(mahasiswa);
177         }
178         if (mahasiswa.ipk < current.mahasiswa.ipk) {
179             current.left = addRekursifRek(current.left, mahasiswa);
180         } else {
181             current.right = addRekursifRek(current.right, mahasiswa);
182         }
183         return current;
184     }
```

-
```
Menambahkan mahasiswa dengan rekursif:
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:244160300 Nama: Gita Kelas: C IPK: 3.6
NIM:244160131 Nama: Devi Kelas: A IPK: 3.72
NIM:244160221 Nama: Badar Kelas: B IPK: 3.85
```

-

  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.

```
186         //Mencari IPK minimum
187     public Mahasiswa24 cariMinIPK() {
188         if (isEmpty()) return null;
189         Node24 current = root;
190             while (current.left != null) {
191                 current = current.left;
192             }
193         return current.mahasiswa;
194     }
195
196         //Mencari IPK maksimum
197     public Mahasiswa24 cariMaxIPK() {
198         if (isEmpty()) return null;
199         Node24 current = root;
200             while (current.right != null) {
201                 current = current.right;
202             }
203         return current.mahasiswa;
204     }
```

-
```
Mahasiswa dengan IPK terkecil:
Nama: Candra, IPK: 3.21

Mahasiswa dengan IPK terbesar:
Nama: Badar, 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.

```
206     void tampilRekursifDiAtas(Node24 node, double ipkBatas) {
207         if (node == null) return;
208         tampilRekursifDiAtas(node.left, ipkBatas);
209         if (node.mahasiswa.ipk > ipkBatas) {
210             System.out.println("Nama: " + node.mahasiswa.nama + ", IPK: " + node.mahasiswa.ipk);
211         }
212         tampilRekursifDiAtas(node.right, ipkBatas);
213     }
214         // Menampilkan mahasiswa dengan IPK di atas batas tertentu
215     public void tampilMahasiswaIPKdiAtas(double ipkBatas) {
216         tampilRekursifDiAtas(root, ipkBatas);
217     }
```

-

```
Mahasiswa dengan IPK di atas 3.50:
Nama: Dewi, IPK: 3.54
Nama: Gita, IPK: 3.6
Nama: Devi, IPK: 3.72
Nama: Badar, IPK: 3.85
```

-

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

```
29      public void add(Mahasiswa24 dataMahasiswa) {
30          if (idxLast < this.dataMahasiswa.length - 1) {
31              idxLast++;
32              this.dataMahasiswa[idxLast] = dataMahasiswa;
33          } else {
34              System.out.println(x:"Tree is full");
35          }
36      }
```

   • method traversePreOrder()

```
38      public void traversePreOrder(int idxStart) {
39          if (idxStart <= idxLast && dataMahasiswa[idxStart] != null) {
40              System.out.println(dataMahasiswa[idxStart]);
41              traversePreOrder(2 * idxStart + 1);
42              traversePreOrder(2 * idxStart + 2);
43          }
44      }
```