NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

13.3 Kegiatan Praktikum 1

13.2.1. Percobaan 1

Class Node :

PrakASD_1F_13 > src > P14 > Praktikum1 > Node13.java > ...

```java
package P14.Praktikum1;

Codeium: Refactor | Explain
public class Node13 {
    int data;
    Node13 left, right;

    public Node13() {

    }

    public Node13(int data) {
        this.left = null;
        this.data = data;
        this.right = null;
    }
}

```

Class BinaryTree :

PrakASD_1F_13 > src > P14 > BinaryTree13.java > BinaryTree13 > add(int)

```java
package P14;

Codeium: Refactor | Explain
public class BinaryTree13 {
    Node13 root;

    public BinaryTree13() {
        root = null;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    boolean isEmpty() {
        return root == null;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    void add(int data) {
        if (isEmpty()) {
            root = new Node13(data);
        } else {
            Node13 current = root;
            while (true) {
                if (data < current.data) {
                    if (current.left != null) {
                        current = current.left;
                    } else {
                        current.left = new Node13(data);
                        break;
                    }
                } else {
                    if (current.right != null) {
                        current = current.right;
                    } else {
                        current.right = new Node13(data);
                        break;
                    }
                }
            }
        }
    }

```

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

```java
boolean find(int data) {
    boolean result = false;
    Node13 current =  root;
    while (current != null) {
        if (current.data != data) {
            result = true;
            break;
        } else if (data > current.data) {
            current = current.left;
        } else {
            current = current.right;
        }
    }
    return result;
}

Codeium: Refactor | Explain | Generate Javadoc | ×
void traversePreOrder(Node13 node) {
    if (node != null) {
        System.out.print(" " + node.data);
        traversePreOrder(node.left);
        traversePreOrder(node.right);
    }
}

Codeium: Refactor | Explain | Generate Javadoc | ×
void traversePostOrder(Node13 node) {
    if (node != null) {
        traversePostOrder(node.left);
        traversePostOrder(node.right);
        System.out.print(" " + node.data);
    }
}

Codeium: Refactor | Explain | Generate Javadoc | ×
void traverseInOrder(Node13 node) {
    if (node != null) {
        traverseInOrder(node.left);
        System.out.print(" " + node.data);
        traverseInOrder(node.right);
    }
}
```

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

```
79      Node13 getSuccessor(Node13 del) {
80          Node13 successor = del.right;
81          Node13 successorParent = del;
82          while (successor.left != null) {
83              successorParent = successor;
84              successor = successor.left;
85          }
86          if (successor != del.right) {
87              successorParent.left = successor.right;
88              successor.right = del.right;
89          }
90          return successor;
91      }
92
        Codeium: Refactor | Explain | Generate Javadoc | ×
93      void delete(int data) {
94          if (isEmpty()) {
95              System.out.println(x:"Tree is empty!");
96              return;
97          }
98          // Find node (current) that will be deleted
99          Node13 parent = root;
100         Node13 current = root;
101         boolean isLeftChild = false;
102         while (current != null) {
103             if (current.data == data) {
104                 break;
105             } else if (data < current.data) {
106                 parent = current;
107                 current = current.left;
108                 isLeftChild = true;
109             } else if (data > current.data) {
110                 parent = current;
111                 current = current.right;
112                 isLeftChild = false;
113             }
114         }
```

```
115         // deletion
116         if (current == null) {
117             System.out.println(x:"Could't find data!");
118             return;
119         } else {
120             // if there is no child, simply delete it
121             if (current.left == null && current.right == null) {
122                 if (current == root) {
123                     root = null;
124                 } else {
125                     if (isLeftChild) {
126                         parent.left = null;
127                     } else {
128                         parent.right = null;
129                     }
130                 }
131             } else if (current.left == null) { // if there is 1 child (right)
132                 if (current == root) {
133                     root = current.right;
134                 } else {
135                     if (isLeftChild) {
136                         parent.left = current.right;
137                     } else {
138                         parent.right = current.right;
139                     }
140                 }
141             } else if (current.right == null ) {
142                 if (current == root) {
143                     root = current.left;
144                 } else {
145                     if (isLeftChild) {
146                         parent.left = current.left;
147                     } else {
148                         parent.right = current.left;
149                     }
150                 }
```

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

```
151             } else { // if there is 2 childs
152                 Node13 successor = getSuccessor(current);
153                 if (current == root) {
154                     root = successor;
155                 } else {
156                     if (isLeftChild) {
157                         parent.left = successor;
158                     } else {
159                         parent.right = successor;
160                     }
161                     successor.left = current.left;
162                 }
163             }
164         }
165     }
166 }
167
```

Class Main :

PrakASD_1F_13 > src > P14 > BinaryTreeMain13.java > ...

```
1   package P14;
2
    Codeium: Refactor | Explain
3   public class BinaryTreeMain13 {
        Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | ×
4       public static void main(String[] args) {
5           BinaryTree13 bt = new BinaryTree13();
6           bt.add(data:6);
7           bt.add(data:4);
8           bt.add(data:8);
9           bt.add(data:3);
10          bt.add(data:5);
11          bt.add(data:7);
12          bt.add(data:9);
13          bt.add(data:10);
14          bt.add(data:15);
15          System.out.print(s:"PreOrder Traversal : ");
16          bt.traversePreOrder(bt.root);
17          System.out.println(x:"");
18          System.out.print(s:"InOrder Traversal : ");
19          bt.traverseInOrder(bt.root);
20          System.out.println(x:"");
21          System.out.print(s:"PostOrder Traversal : ");
22          bt.traversePostOrder(bt.root);
23          System.out.println(x:"");
24          System.out.println("Find Node : " + bt.find(data:5));
25          System.out.println(x:"Delete Node 8 ");
26          bt.delete(data:8);
27          System.out.println(x:"");
28          System.out.print(s:"PreOrder Traversal : ");
29          bt.traversePreOrder(bt.root);
30          System.out.println(x:"");
31      }
32  }
33
```

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

Output :

```
PreOrder Traversal :  6 4 3 5 8 7 9 10 15
InOrder Traversal :  3 4 5 6 7 8 9 10 15
PostOrder Traversal :  3 5 4 7 15 10 9 8 6
Find Node : true
Delete Node 8

PreOrder Traversal :  6 4 3 5 9 7 10 15
PS E:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

Question :

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
   Jawab : Karena dalam Binary Search Tree proses pencariannya lebih cepat dan efisien, lalu pengaturan node yang sistematis dan pengurangan ruang pencarian yang efektif pada setiap Langkah, sedangkan Binary tree biasa tidak memiliki pengaturan khusus.

2. Untuk apakah di class **Node**, kegunaan dari atribut **left** dan **right**?
   Jawab : untuk mengetahui child node kiri dan kanan dari suatu node pada struktur data pohon.

3. a. Untuk apakah kegunaan dari atribut root di dalam class **BinaryTree**?
   b. Ketika objek tree pertama kali dibuat, apakah nilai dari **root**?
   Jawab : a. untuk menyimpan referensi ke node akar(root node) dari binary tree.
   b. nilai dari root adalah null, karena belum ada node yang dibuat dan struktur pohon masih kosong.

4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
   Jawab : proses yang terjadi adalah node baru akan menjadi root dari pohon (tree), jadi node baru yang ditambahkan akan menjadi awal dari struktur pohon.

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

```
if(data < current.data) {
        if(current.left != null) {
           current = current.left;
        } else {
           current.left = new Node(data).
           break;
        }
}
```

   Jawab :

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

- pertama pengecekan apakah data yang ditambahkan lebih kecil daripada data node saat ini. Jika true, maka menjalankan kode di dalam blok if.
- kedua pengecekan apakah node saat ini memiliki child node kiri. Jika true, maka node saat ini akan digeser ke child node kiri.
- Saat penggeseran ke child node kiri akan dilakukan secara rekursif sampai bertemu node yang memiliki child node kiri null.
- lalu jika node saat ini tidak memiliki child node kiri, maka akan terjadi penambahan node baru sebagai child node kiri.
- Dan break untuk keluar dari loop jika sudah menemukan node yang memiliki child node kiri null.

NAMA	: Gilang Purnomo
NIM	: 2341720042
NO ABSEN	: 13
KELAS	: 1F
MATERI	: Tree

13.3 Kegiatan Praktikum 2

13.3.1 Tahapan Percobaan

Class BinaryTreeArray :

PrakASD_1F_13 > src > P14 > Praktikum2 > BinaryTreeArray13.java > ...

```java
package P14.Praktikum2;

Codeium: Refactor | Explain
public class BinaryTreeArray13 {
    int[] data;
    int idxLast;

    public BinaryTreeArray13() {
        data = new int[10];
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    void populateData(int data[], int idxLast) {
        this.data = data;
        this.idxLast = idxLast;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    void traverseInOrder(int idxStart) {
        if (idxStart <= idxLast) {
            traverseInOrder(2 * idxStart + 1);
            System.out.print(data[idxStart] + " ");
            traverseInOrder(2 * idxStart + 2);
        }
    }
}
```

Class Main :

PrakASD_1F_13 > src > P14 > Praktikum2 > BinaryTreeArrayMain13.java > ...

```java
package P14.Praktikum2;

Codeium: Refactor | Explain
public class BinaryTreeArrayMain13 {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        BinaryTreeArray13 bta = new BinaryTreeArray13();
        int[] data = {6, 4, 8, 3, 5, 7, 9, 0, 0, 0};
        int idxLast = 6;
        bta.populateData(data, idxLast);
        System.out.print(s:"\nInOrder Traversal : ");
        bta.traverseInOrder(idxStart:0);
        System.out.println(x:"\n");
    }
}
```

Output :

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

```
InOrder Traversal : 3 4 5 6 7 8 9

PS E:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

Question :

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class **BinaryTreeArray**?
   Jawab :
   - data : digunakan untuk menyimpan data dari setiap node dalam binary tree dalam bentuk array.
   - idxLast : Menyimpan indeks dari elemen terakhir dalam array, atribut ini bisa membantu saat proses pencarian, penambahan node baru, dan menghapus node dalam binary tree.
   -
2. Apakah kegunaan dari method **populateData()**?
   Jawab : untuk mengisikan atribut data dengan nilai yang sudah diberikan serta menetapkan nilai idxLast untuk menandai bahwa elemen akhir dalam array data.

3. Apakah kegunaan dari method **traverseInOrder()**?
   Jawab : digunakan untuk traversal pada binary tree dengan cara in-order yang melakukan pengunjungan node dalam urutan: node kiri, root, dan node kanan.

4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masing - masing?
   Jawab : jadi left child dari I berada di indeks : 2*2 + 1 = 5, sendangkan right child berada di indeks 2*2+2 = 6.

5. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?
   Jawab : untuk menandai indeks terakhir dalam array data, memastikan operasi traversal dan manipulasi tree dilakukan pada elemen yang valid, serta digunakan untuk batas ukuran binary tree yang direpresentasikan oleh array.

NAMA	: Gilang Purnomo
NIM	: 2341720042
NO ABSEN	: 13
KELAS	: 1F
MATERI	: Tree

13.4 Tugas Praktikum

1. Buat method di dalam class **BinaryTree** yang akan menambahkan node dengan cara rekursif.

```
39      void add(int data) {
40          root = addRecursive(root, data);
41      }
42
43      public Node13 addRecursive(Node13 current, int data) {
44          if (current == null) {
45              return new Node13(data);
46          }
47          if (data < current.data) {
48              current.left = addRecursive(current.left, data);
49          } else if (data > current.data) {
50              current.right = addRecursive(current.right, data);
51          }
52          return current;
53      }
```

Main :

PrakASD_1F_13 > src > P14 > Praktikum1 > BinaryTreeMain13.java > ...

```
1   package P14.Praktikum1;
2
3   public class BinaryTreeMain13 {
        Run | Debug
4       public static void main(String[] args) {
5           BinaryTree13 bt = new BinaryTree13();
6           bt.add(data:6);
7           bt.add(data:4);
8           bt.add(data:8);
9           bt.add(data:3);
10          bt.add(data:5);
11          bt.add(data:7);
12          bt.add(data:9);
13          bt.add(data:10);
14          bt.add(data:15);
15          System.out.print(s:"PreOrder Traversal : ");
16          bt.traversePreOrder(bt.root);
17          System.out.println(x:"");
18          System.out.print(s:"InOrder Traversal : ");
19          bt.traverseInOrder(bt.root);
20          System.out.println(x:"");
21          System.out.print(s:"PostOrder Traversal : ");
22          bt.traversePostOrder(bt.root);
23          System.out.println(x:"");
24          System.out.println("Find Node : " + bt.find(data:5));
25          System.out.println(x:"Delete Node 8 ");
26          bt.delete(data:8);
27          System.out.println(x:"");
28          System.out.print(s:"PreOrder Traversal : ");
29          bt.traversePreOrder(bt.root);
30          System.out.println(x:"");
```

Output :

```
PreOrder Traversal :  6 4 3 5 8 7 9 10 15
InOrder Traversal :  3 4 5 6 7 8 9 10 15
PostOrder Traversal :  3 5 4 7 15 10 9 8 6
Find Node : true
Delete Node 8

PreOrder Traversal :  6 4 3 5 9 7 10 15
PS E:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

NAMA	: Gilang Purnomo
NIM	: 2341720042
NO ABSEN	: 13
KELAS	: 1F
MATERI	: Tree

2. Buat method di dalam class **BinaryTree** untuk menampilkan nilai paling kecil dan yang paling besar yang ada di dalam tree.

```
183     public int minValue(Node13 node) {
184         int minValue = node.data;
185         while (node.left != null) {
186             minValue = node.left.data;
187             node = node.left;
188         }
189         return minValue;
190     }
191
192     public int maxValue(Node13 node) {
193         int maxValue = node.data;
194         while (node.right != null) {
195             maxValue = node.right.data;
196             node = node.right;
197         }
198         return maxValue;
199     }
```

Main :

```
32          System.out.println("Min value   : " + bt.minValue(bt.root));
33          System.out.println("Max value   : " + bt.maxValue(bt.root));
```

Output :

```
PreOrder Traversal :  6 4 3 5 8 7 9 10 15
InOrder Traversal :  3 4 5 6 7 8 9 10 15
PostOrder Traversal :  3 5 4 7 15 10 9 8 6
Find Node : true
Delete Node 8

PreOrder Traversal :  6 4 3 5 9 7 10 15
Min value   : 3
Max value   : 15
PS E:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

3. Buat method di dalam class **BinaryTree** untuk menampilkan data yang ada di leaf.

```
201     public void displayleaf(Node13 node) {
202         if (node == null) {
203             return;
204         }
205         if (node.left == null && node.right == null) {
206             System.out.print(node.data + " ");
207         }
208         displayleaf(node.left);
209         displayleaf(node.right);
210     }
```

Main :

```
34          System.out.print(s:"Data leaf : ");
35          bt.displayleaf(bt.root);
```

Output :

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

```
PreOrder Traversal :  6 4 3 5 8 7 9 10 15
InOrder Traversal :  3 4 5 6 7 8 9 10 15
PostOrder Traversal :  3 5 4 7 15 10 9 8 6
Find Node : true
Delete Node 8

PreOrder Traversal :  6 4 3 5 9 7 10 15
Min value  : 3
Max value  : 15
Data leaf : 3 5 7 15
PS E:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

4. Buat method di dalam class **BinaryTree** untuk menampilkan berapa jumlah leaf yang ada di dalam tree.

```
212     public int countLeaf(Node13 node) {
213         if (node == null) {
214             return 0;
215         }
216         if (node.left == null && node.right == null) {
217             return 1;
218         }
219         return countLeaf(node.left) + countLeaf(node.right);
220     }
```

Main :

```
36          System.out.println(x:"");
37          System.out.println("Jumlah leaf : " + bt.countLeaf(bt.root));
```

Output :

```
PreOrder Traversal :  6 4 3 5 8 7 9 10 15
InOrder Traversal :  3 4 5 6 7 8 9 10 15
PostOrder Traversal :  3 5 4 7 15 10 9 8 6
Find Node : true
Delete Node 8

PreOrder Traversal :  6 4 3 5 9 7 10 15
Min value  : 3
Max value  : 15
Data leaf : 3 5 7 15
Jumlah leaf : 4
PS E:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

5. Modifikasi class **BinaryTreeArray**, dan tambahkan :
   • method **add(int data)** untuk memasukan data ke dalam tree

```
17      void add(int data) {
18          if (idxLast == this.data.length - 1) {
19              System.out.println(x:"Data sudah penuh");
20          }
21          idxLast++;
22          this.data[idxLast] = data;
23      }
```

   • method **traversePreOrder()** dan **traversePostOrder().**

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : Tree

```
33      void traversePreOrder(int idxStart) {
34          if (idxStart <= idxLast) {
35              System.out.print(data[idxStart] + " ");
36              traversePreOrder(2 * idxStart + 1);
37              traversePreOrder(2 * idxStart + 2);
38          }
39      }
40
41      void traversePostOrder(int idxStart) {
42          if (idxStart <= idxLast) {
43              traversePostOrder(2 * idxStart + 1);
44              traversePostOrder(2 * idxStart + 2);
45              System.out.print(data[idxStart] + " ");
46          }
47      }
```

Main :

PrakASD_1F_13 > src > P14 > Praktikum2 > BinaryTreeArrayMain13.java > ...

```
1   package P14.Praktikum2;
2
3   public class BinaryTreeArrayMain13 {
        Run | Debug
4       public static void main(String[] args) {
5           BinaryTreeArray13 bta = new BinaryTreeArray13();
6           int[] data = {6, 4, 8, 3, 5, 7, 9, 0, 0, 0};
7           int idxLast = 6;
8           bta.populateData(data, idxLast);
9           System.out.print(s:"\nInOrder Traversal : ");
10          bta.traverseInOrder(idxStart:0);
11          System.out.println(x:"\n");
12
13          System.out.print(s:"PreOrder Traversal : ");
14          bta.traversePreOrder(idxStart:0);
15          System.out.println(x:"\n");
16
17          System.out.print(s:"PostOrder Traversal : ");
18          bta.traversePostOrder(idxStart:0);
19          System.out.println(x:"\n");
20
21          System.out.print(s:"Add 10 to tree : ");
22          bta.add(data:10);
23          bta.traverseInOrder(idxStart:0);
24          System.out.println(x:"\n");
25      }
26  }
27
```

Output :

```
InOrder Traversal : 3 4 5 6 7 8 9

PreOrder Traversal : 6 4 3 5 8 7 9

PostOrder Traversal : 3 5 4 7 9 8 6

Add 10 to tree : 10 3 4 5 6 7 8 9

PS E:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```