

Panji Iman Baskoro  
171111023  
Praktikum Progdas 2

## Modul 5

BinaryTreeNode.java

```
public class BinaryTreeNode {  
1.  
2.  BinaryTreeNode parent;  
3.  BinaryTreeNode left;  
4.  BinaryTreeNode right;  
5.  int data;  
6.  
7.  BinaryTreeNode(int new_data) {  
8.      this.data = new_data;  
9.      this.parent = null;  
10.     this.left = null;  
11.     this.right = null;  
12. }  
13.  
14.  
15. void set_parent(BinaryTreeNode other) {  
16.     this.parent = other;  
17.     if (other != null) {  
18.         if (other.data > this.data) {  
19.             other.left = this;  
20.         } else {  
21.             other.right = this;  
22.         }  
23.     }  
24. }  
25.  
26. void set_left(BinaryTreeNode other) {  
27.     this.left = other;  
28.     if (other != null) {  
29.         other.parent = this;  
30.     }  
31. }  
32.  
33. void set_right(BinaryTreeNode other) {  
34.     this.right = other;  
35.     if (other != null) {  
36.         other.parent = this;  
37.     }  
38. }  
39. boolean is_left() {  
40.     return this.parent != null && parent.left == this;  
41. }  
42.  
43. boolean is_right() {
```

```
44.     return this.parent != null && parent.right == this;
45. }
46.
47. boolean has_right_and_left() {
48.     return this.left != null && this.right != null;
49. }
50.
51. boolean only_has_left() {
52.     return this.left != null && this.right == null;
53. }
54.
55. boolean only_has_right() {
56.     if (this.right != null || this.left == null) {
57.
58.     }
59.     return this.right != null && this.left == null;
60. }
61.
62. boolean has_no_child() {
63.     return this.left == null && this.right == null;
64. }
65. void unset_parent() {
66.     if (this.is_left()) {
67.         parent.left = null;
68.         this.parent = null;
69.
70.     } else if (this.is_right()) {
71.         parent.right = null;
72.         this.parent = null;
73.
74.     }
75. }
76.
77.
78. BinaryTreeNode most_left_child() {
79.     BinaryTreeNode child = this.left;
80.     while (child.left != null) {
81.         child = child.left;
82.
83.     }
84.     return child;
85. }
86.
87. BinaryTreeNode most_right_child() {
88.     BinaryTreeNode child = this.right;
89.     while (child.right != null) {
90.         child = child.right;
91.     }
92.     return child;
93. }
94.
95. void print(String spaces, String label) {
```

```
96.     System.out.println(spaces + label + this.data);
97.     if (this.left != null) {
98.         this.left.print(spaces + " ", " LEFT ");
99.     }
100.    if (this.right != null) {
101.        this.right.print(spaces+ " ", " RIGHT ");
102.    }
103. }
104.
105. void print() {
106.     this.print(" ", "NODE ");
107. }
108.
109. void infix() {
110.     System.out.print("( ");
111.     if (this.left != null) {
112.         left.infix();
113.     } else {
114.         System.out.print("null");
115.     }
116.     System.out.print(" " + this.data + " ");
117.     if (this.right != null) {
118.         right.infix();
119.     } else {
120.         System.out.print("null");
121.     }
122.     System.out.print(")");
123. }
124.
125. void prefix() {
126.     System.out.print(this.data + "(");
127.     if (this.left != null) {
128.         left.prefix();
129.     } else {
130.         System.out.print("null");
131.     }
132.     System.out.print(" ");
133.     if (this.right != null) {
134.         right.prefix();
135.     } else {
136.         System.out.print("null");
137.     }
138.     System.out.print(") ");
139. }
140. void postfix() {
141.     System.out.print("( ");
142.     if (this.left != null) {
143.         left.postfix();
144.     } else {
145.         System.out.print("null");
146.     }
147.     System.out.print(" ");
```

```
148.     if (this.right != null) {
149.         right.postfix();
150.     } else {
151.         System.out.print("null");
152.     }
153.     System.out.print(" " + this.data);
154. }
155.}
```

## BinaryTree .java

```
1. public class BinaryTree {
2.
3.     BinaryTreeNode root;
4.
5.     public BinaryTree() {
6.         this.root = null;
7.     }
8.
9.     void print() {
10.        if (this.root != null) {
11.            this.root.print();
12.        }
13.    }
14.
15.    void prefix() {
16.        if (this.root != null) {
17.            this.root.prefix();
18.        }
19.        System.out.println("");
20.    }
21.
22.    void infix() {
23.        if (this.root != null) {
24.            this.root.infix();
25.        }
26.        System.out.println("");
27.    }
28.    void postfix() {
29.        if (this.root != null) {
30.            this.root.postfix();
31.        }
32.        System.out.println("");
33.    }
34.    void push(BinaryTreeNode new_node) {
35.        if (this.root == null) {
36.            this.root = new_node;
37.        } else {
38.            BinaryTreeNode current = this.root;
39.            while (current != null) {
```

```

40.         if (new_node.data > current.data) {
41.             if (current.right == null) {
42.                 current.set_right(new_node);
43.                 break;
44.             } else {
45.                 current = current.right;
46.             }
47.         } else {
48.             if (current.left == null) {
49.                 current.set_left(new_node);
50.                 break;
51.             } else {
52.                 current = current.left;
53.             }
54.         }
55.     }
56. }
57. }
58.
59. void delete(BinaryTreeNode deleted) {
60.     if (this.root != null) {
61.         if (deleted.has_no_child()) {
62.             if (deleted == this.root) {
63.                 this.root = null;
64.             } else {
65.                 deleted.unset_parent();
66.             }
67.         } else if (deleted.only_has_left() || deleted.only_has_right()) {
68.             BinaryTreeNode replacement = null;
69.             if (deleted.only_has_left()) {
70.                 replacement = deleted.left;
71.             } else {
72.                 replacement = deleted.right;
73.             }
74.             if (deleted == this.root) {
75.                 this.root = replacement;
76.                 this.root.unset_parent();
77.             }
78.             else if (deleted.is_left()) {
79.                 deleted.parent.set_left(replacement);
80.                 deleted.unset_parent();
81.             }
82.             else if (deleted.is_right()) {
83.                 deleted.parent.set_right(replacement);
84.                 deleted.unset_parent();
85.             }
86.         }
87.     } else {
88.         BinaryTreeNode replacement = deleted.left;
89.         if (replacement.right != null) {
90.             replacement = replacement.most_right_child();
91.         }

```

```
92.         BinaryTreeNode parent_of_replacement = replacement.parent;
93.         if (replacement.only_has_right()) {
94.             parent_of_replacement.set_left(replacement.right);
95.         }
96.         replacement.unset_parent();
97.         replacement.set_left(deleted.left);
98.         replacement.set_right(deleted.right);
99.         if (deleted == this.root) {
100.             this.root = replacement;
101.         } else if (deleted.is_left()) {
102.             deleted.parent.set_left(replacement);
103.         } else if (deleted.is_right()) {
104.             deleted.parent.set_right(replacement);
105.         }
106.     }
107. }
108. }
109.
110. void caricari(int key) {
111.     if (this.root == null) {
112.         System.out.println("Binary Tree Kosong");
113.     } else {
114.         BinaryTreeNode current = this.root;
115.         while (current != null) {
116.             if (key == current.data) {
117.                 System.out.println("wowo angka yang dicari ada !");
118.                 break;
119.             }
120.             if (key > current.data) {
121.                 current = current.right;
122.             } else {
123.                 current = current.left;
124.             }
125.         }
126.     }
127.
128. }
129.
130. }
```

## binarySearch.java

```
1. import java.util.Scanner;
2.
3. public class binarySearch {
4.
5.     public static void main(String[] args) {
6.         BinaryTree bt = new BinaryTree();
7.         Scanner sc = new Scanner(System.in);
8.         int angka = 0, jumangka, cari;
9.         char ulang = 'y';
10.
11.         System.out.println("    ** Binary Search in Binary Tree **");
12.         System.out.println("-----");
13.         System.out.print("Masukan jumlah angka\t: ");
14.         jumangka = sc.nextInt();
15.         for (int i = 0; i < jumangka; i++) {
16.             System.out.print("Angka ke " + (i + 1) + "\t: ");
17.             angka = sc.nextInt();
18.             bt.push(new BinaryTreeNode(angka));
19.         }
20.         System.out.println("-----");
21.         bt.print();
22.         do {
23.             System.out.println("-----");
24.             System.out.print("Masukan angka yang anda cari : ");
25.             cari = sc.nextInt();
26.             bt.caricari(cari);
27.             do {
28.                 System.out.print("Cari Angka lagi? (Y / T)\t");
29.                 ulang = sc.next().charAt(0);
30.             } while (ulang != 't' && ulang != 'y');
31.
32.         } while (ulang == 'y');
33.
34.     }
35.
36. }
```

output :

```
budosen@budosen-pc:/mnt/b2c7efbf-ef52-437d-8ca7-e46ea5
ertemuan 5/gatus$ java binarySearch
** Binary Search in Binary Tree **
-----
Masukan jumlah angka      : 3
Angka ke 1                : 7
Angka ke 2                : 5
Angka ke 3                : 3
-----
NODE 7
  LEFT 5
    LEFT 3
-----
Masukan angka yang anda cari : 5
wowo angka yang dicari ada !
Cari Angka lagi? (Y / T)
```

Terimakasih