Nama : Monica Tifani Zahara

NRP : 171 111 077

Kelas : TI C / Praktikum Pemrograman Dasar 2

1. Source Code

```
01. package modul5;
02. 
03. import java.util.Scanner;
04. 
05. /**
06.  *
07.  * @author Monica
08.  */
09. public class binarySearch {
10. 
11.     public static void main(String[] args) {
12.         BinaryTree bt = new BinaryTree();
13.         Scanner sc = new Scanner(System.in);
14.         int angka = 0, jumangka, cari;
15.         char ulang = 'y';
16. 
17.         System.out.println("    ** Binary Search in Binary Tree **");
18.         System.out.println("---------------------------------------");
19.         System.out.print("Masukan jumlah angka\t: ");
20.         jumangka = sc.nextInt();
21.         for (int i = 0; i < jumangka; i++) {
22.             System.out.print("Angka ke " + (i + 1) + "\t: ");
23.             angka = sc.nextInt();
24.             bt.push(new BinaryTreeNode(angka));
25.         }
26.         System.out.println("---------------------------------");
27.         bt.print();
28.         do {
29.             System.out.println("---------------------------------");
30.             System.out.print("Masukan angka yang anda cari : ");
31.             cari = sc.nextInt();
32.             bt.findNode(cari);
33.             do {
34.                 System.out.print("Cari Angka lagi? (Y / T)\t");
35.                 ulang = sc.next().charAt(0);
36.             } while (ulang != 't' && ulang != 'y');
37. 
38.         } while (ulang == 'y');
39. 
40.     }
41. 
42. }
```

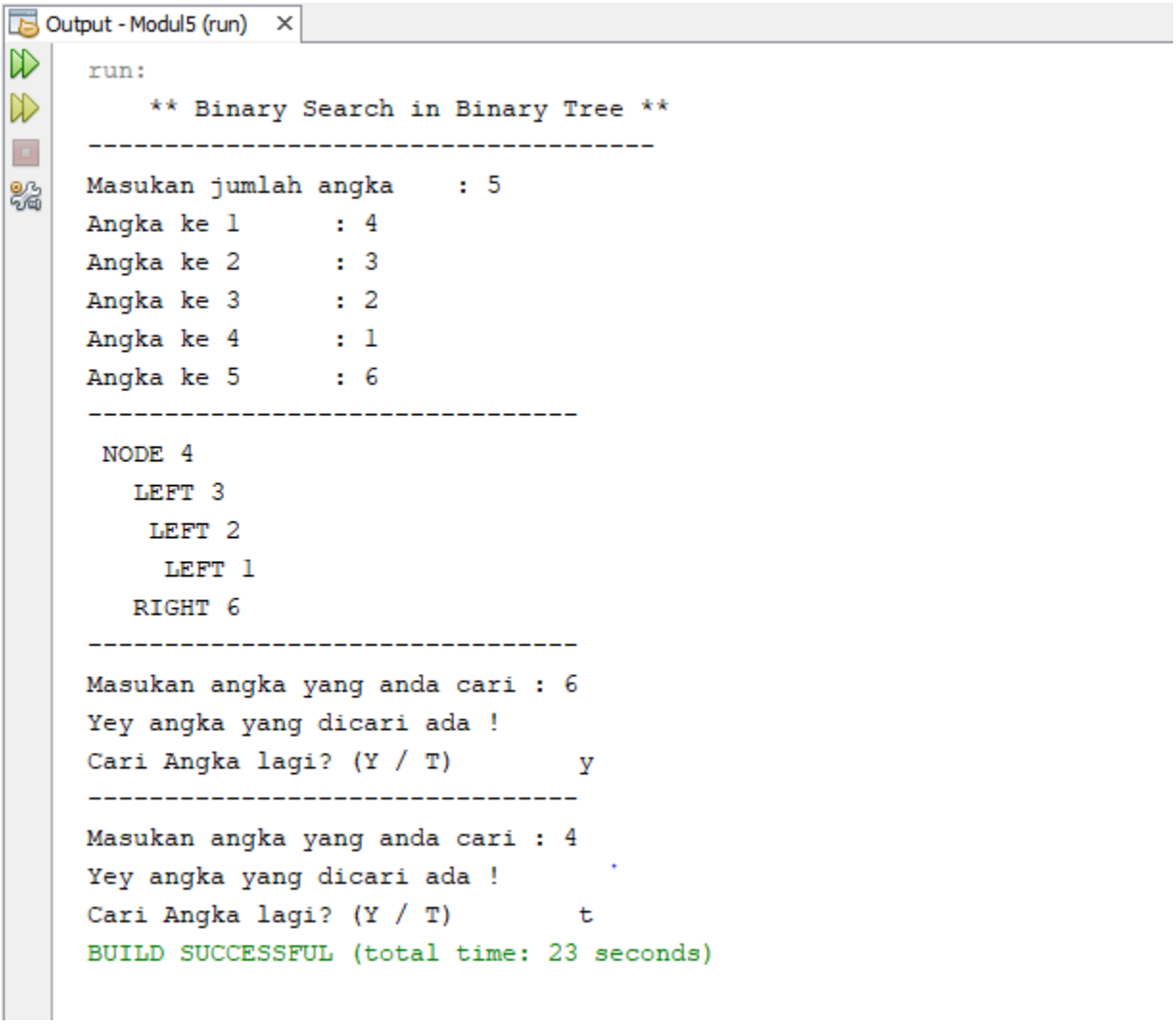
```java
package modul5;

public class BinaryTree {

    BinaryTreeNode root;

    public BinaryTree() {
        this.root = null;
    }

    void print() {
        if (this.root != null) {
            this.root.print();
        }
    }

    void prefix() {
        if (this.root != null) {
            this.root.prefix();
        }
        System.out.println("");
    }

    void infix() {
        if (this.root != null) {
            this.root.infix();
        }
        System.out.println("");
    }

    void postfix() {
        if (this.root != null) {
            this.root.postfix();
        }
        System.out.println("");
    }

    void push(BinaryTreeNode new_node) {
        if (this.root == null) {
            this.root = new_node;
        } else {
            BinaryTreeNode current = this.root;
            while (current != null) {
                if (new_node.data > current.data) {
                    if (current.right == null) {
                        current.set_right(new_node);
                        break;
                    } else {
                        current = current.right;
                    }
                } else {
                    if (current.left == null) {
                        current.set_left(new_node);
                        break;
                    } else {
                        current = current.left;
                    }
                }
            }
        }
    }
```

```
104.     void delete(BinaryTreeNode deleted) {
105.         if (this.root != null) {
106.             if (deleted.has_no_child()) {
107.                 if (deleted == this.root) {
108.                     this.root = null;
109.                 } else {
110.                     deleted.unset_parent();
111.                 }
112.             } else if (deleted.only_has_left() || deleted.only_has_right()) {
113.                 BinaryTreeNode replacement = null;
114.                 if (deleted.only_has_left()) {
115.                     replacement = deleted.left;
116.                 } else {
117.                     replacement = deleted.right;
118.                 }
119.                 if (deleted == this.root) {
120.                     this.root = replacement;
121.                     this.root.unset_parent();
122. 
123.                 } else if (deleted.is_left()) {
124.                     deleted.parent.set_left(replacement);
125.                     deleted.unset_parent();
126. 
127.                 } else if (deleted.is_right()) {
128.                     deleted.parent.set_right(replacement);
129.                     deleted.unset_parent();
130. 
131.                 }
132.             } else {
133.                 BinaryTreeNode replacement = deleted.left;
134.                 if (replacement.right != null) {
135.                     replacement = replacement.most_right_child();
136.                 }
137.                 BinaryTreeNode parent_of_replacement = replacement.parent;
138.                 if (replacement.only_has_right()) {
139.                     parent_of_replacement.set_left(replacement.right);
140.                 }
141.                 replacement.unset_parent();
142.                 replacement.set_left(deleted.left);
143.                 replacement.set_right(deleted.right);
144.                 if (deleted == this.root) {
145.                     this.root = replacement;
146.                 } else if (deleted.is_left()) {
147.                     deleted.parent.set_left(replacement);
148.                 } else if (deleted.is_right()) {
149.                     deleted.parent.set_right(replacement);
150.                 }
151.             }
152.         }
153.     }
154. 
```

```
    void findNode(int key) {
        if (this.root == null) {
            System.out.println("Binary Tree Kosong");
        } else {
            BinaryTreeNode current = this.root;
            while (current != null) {
                if (key == current.data) {
                    System.out.println("Yey angka yang dicari ada !");
                    break;
                }
                if (key > current.data) {
                   current = current.right;
                } else {
                    current = current.left;
                }
            }
        }

    }

package modul5;

public class BinaryTreeNode {

    BinaryTreeNode parent;
    BinaryTreeNode left;
    BinaryTreeNode right;
    int data;

    BinaryTreeNode(int new_data) {
        this.data = new_data;
        this.parent = null;
        this.left = null;
        this.right = null;
    }

    void set_parent(BinaryTreeNode other) {
        this.parent = other;
        if (other != null) {
            if (other.data > this.data) {
                other.left = this;
            } else {
                other.right = this;
            }
        }
    }

    void set_left(BinaryTreeNode other) {
        this.left = other;
        if (other != null) {
            other.parent = this;
        }
    }

    void set_right(BinaryTreeNode other) {
        this.right = other;
        if (other != null) {
            other.parent = this;
        }
    }

    boolean is_left() {
        return this.parent != null && parent.left == this;
    }


    boolean is_right() {
        return this.parent != null && parent.right == this;
    }

    boolean has_right_and_left() {
        return this.left != null && this.right != null;
    }

    boolean only_has_left() {
        return this.left != null && this.right == null;
    }

    boolean only_has_right() {
        if (this.right != null || this.left == null) {

        }
        return this.right != null && this.left == null;
    }

    boolean has_no_child() {
        return this.left == null && this.right == null;
    }
```

```
243.
244.     void unset_parent() {
245.         if (this.is_left()) {
246.             parent.left = null;
247.             this.parent = null;
248.
249.         } else if (this.is_right()) {
250.             parent.right = null;
251.             this.parent = null;
252.
253.         }
254.     }
255.
256.     BinaryTreeNode most_left_child() {
257.         BinaryTreeNode child = this.left;
258.         while (child.left != null) {
259.             child = child.left;
260.
261.         }
262.         return child;
263.     }
264.
265.     BinaryTreeNode most_right_child() {
266.         BinaryTreeNode child = this.right;
267.         while (child.right != null) {
268.             child = child.right;
269.         }
270.         return child;
271.     }
272.
273.     void print(String spaces, String label) {
274.         System.out.println(spaces + label + this.data);
275.         if (this.left != null) {
276.             this.left.print(spaces +" ", " LEFT ");
277.         }
278.         if (this.right != null) {
279.             this.right.print(spaces+ " ", " RIGHT ");
280.         }
281.     }
282.
283.     void print() {
284.         this.print(" ", "NODE ");
285.     }
286.
287.     void infix() {
288.         System.out.print("( ");
289.         if (this.left != null) {
290.             left.infix();
291.         } else {
292.             System.out.print("null");
293.         }
294.         System.out.print(" " + this.data + " ");
295.         if (this.right != null) {
296.             right.infix();
297.         } else {
298.             System.out.print("null");
299.         }
300.         System.out.print(")");
301.     }
302.
303.     void prefix() {
304.         System.out.print(this.data + "(");
305.         if (this.left != null) {
306.             left.prefix();
307.         } else {
308.             System.out.print("null");
309.         }
310.         System.out.print(" ");
311.         if (this.right != null) {
312.             right.prefix();
313.         } else {
314.             System.out.print("null");
315.         }
316.         System.out.print(") ");
317.     }
318.
319.     void postfix() {
320.         System.out.print("( ");
321.         if (this.left != null) {
322.             left.postfix();
323.         } else {
324.             System.out.print("null");
325.         }
326.         System.out.print(" ");
327.         if (this.right != null) {
328.             right.postfix();
329.         } else {
330.             System.out.print("null");
331.         }
332.         System.out.print(")" + this.data);
333.     }
334. }
```

2. Output


Output - Modul5 (run)
run:
** Binary Search in Binary Tree **
Masukan jumlah angka : 5
Angka ke 1 : 4
Angka ke 2 : 3
Angka ke 3 : 2
Angka ke 4 : 1
Angka ke 5 : 6
NODE 4
LEFT 3
LEFT 2
LEFT 1
RIGHT 6
Masukan angka yang anda cari : 6
Yey angka yang dicari ada !
Cari Angka lagi? (Y / T) y
Masukan angka yang anda cari : 4
Yey angka yang dicari ada !
Cari Angka lagi? (Y / T) t
BUILD SUCCESSFUL (total time: 23 seconds)