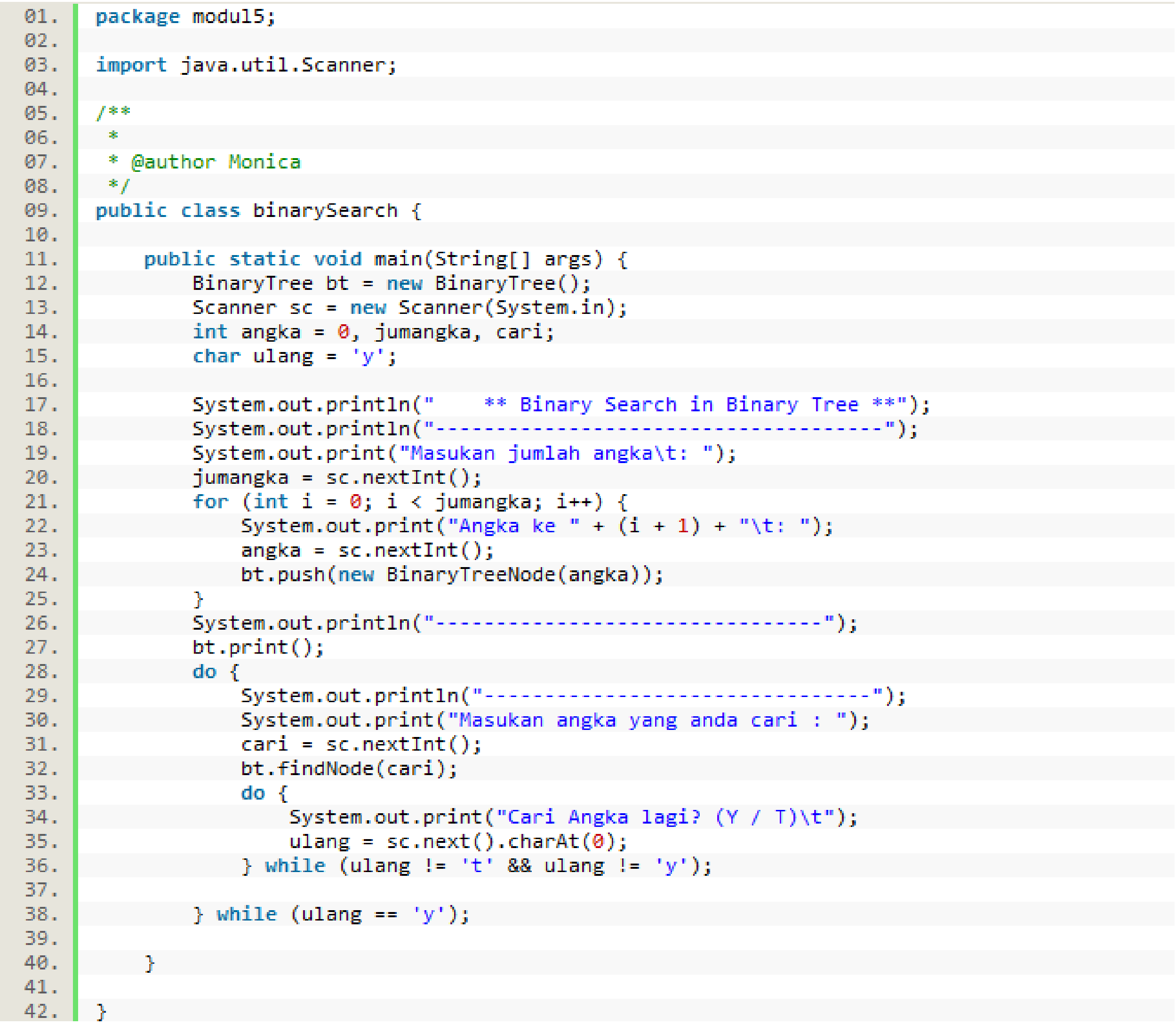
Nama : Monica Tifani Zahara

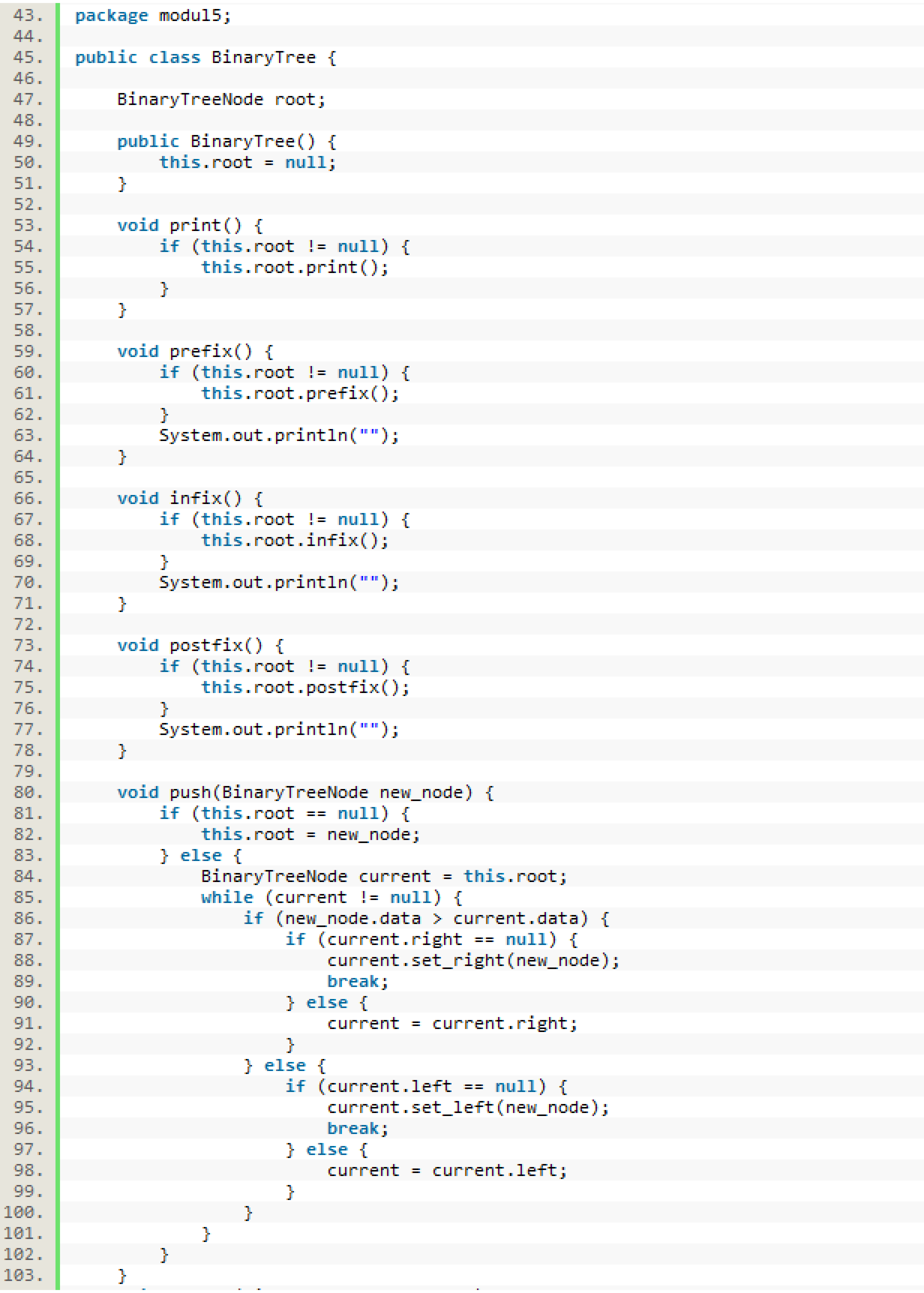
NRP : 171 111 077

Kelas : TI C / Praktikum Pemrograman Dasar 2

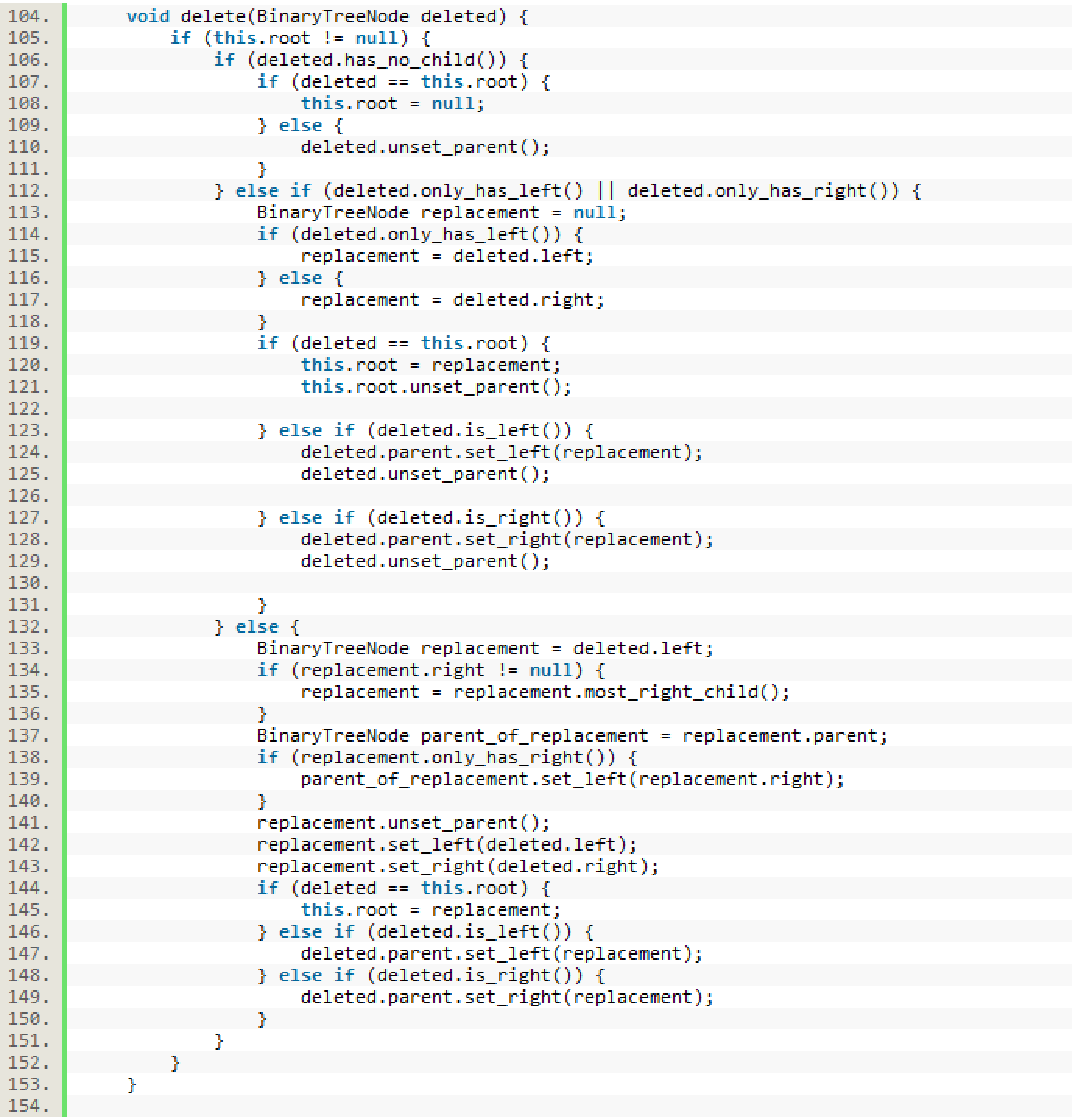
1. Source Code



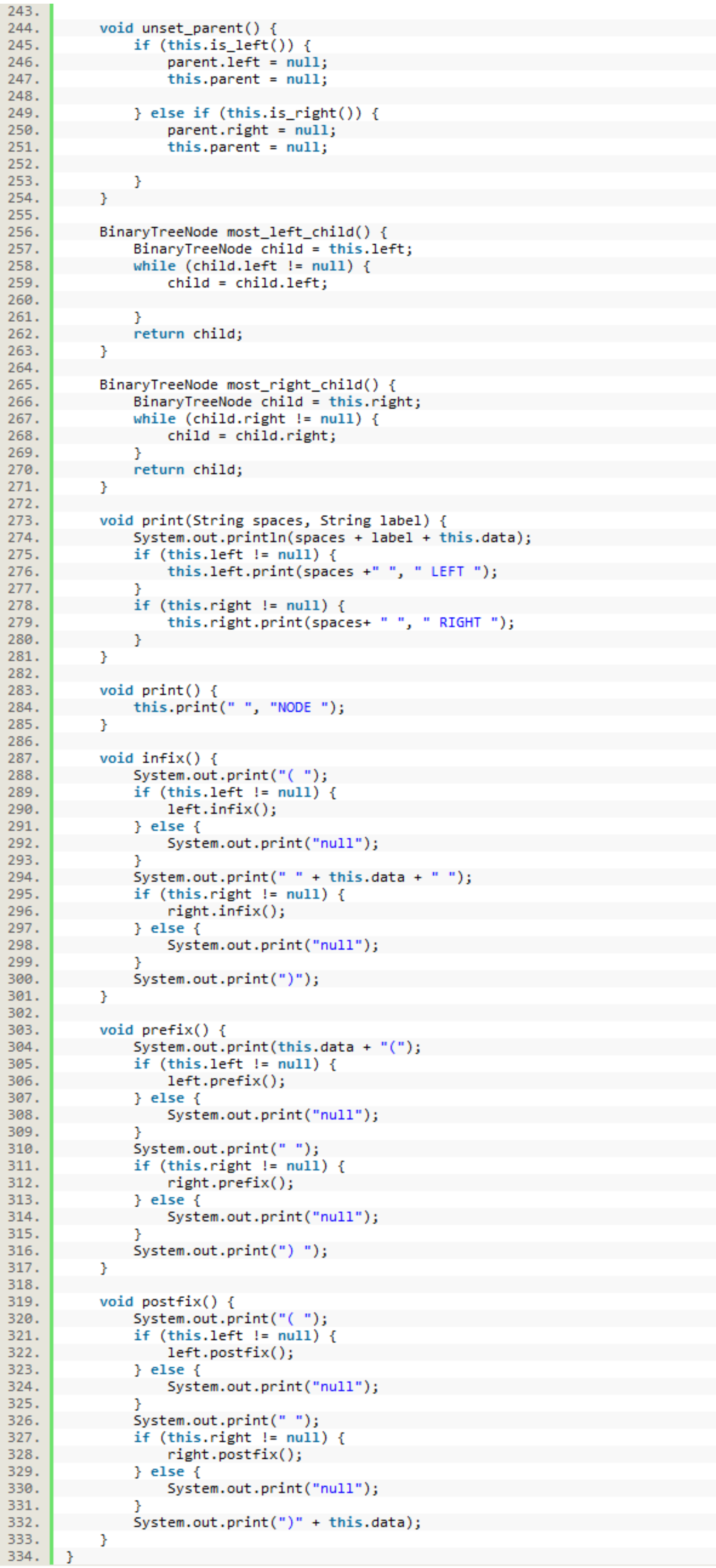
```
package modul5;

import java.util.Scanner;

/**
 *
 * @author Monica
 */
public class binarySearch {

    public static void main(String[] args) {
        BinaryTree bt = new BinaryTree();
        Scanner sc = new Scanner(System.in);
        int angka = 0, jumangka, cari;
        char ulang = 'y';

        System.out.println("    ** Binary Search in Binary Tree **");
        System.out.println("-------------------------------------");
        System.out.print("Masukan jumlah angka\t: ");
        jumangka = sc.nextInt();
        for (int i = 0; i < jumangka; i++) {
            System.out.print("Angka ke " + (i + 1) + "\t: ");
            angka = sc.nextInt();
            bt.push(new BinaryTreeNode(angka));
        }
        System.out.println("--------------------------------");
        bt.print();
        do {
            System.out.println("--------------------------------");
            System.out.print("Masukan angka yang anda cari : ");
            cari = sc.nextInt();
            bt.findNode(cari);
            do {
                System.out.print("Cari Angka lagi? (Y / T)\t");
                ulang = sc.next().charAt(0);
            } while (ulang != 't' && ulang != 'y');

        } while (ulang == 'y');

    }

}
```



```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;
                    }
                }
            }
        }
    }
```

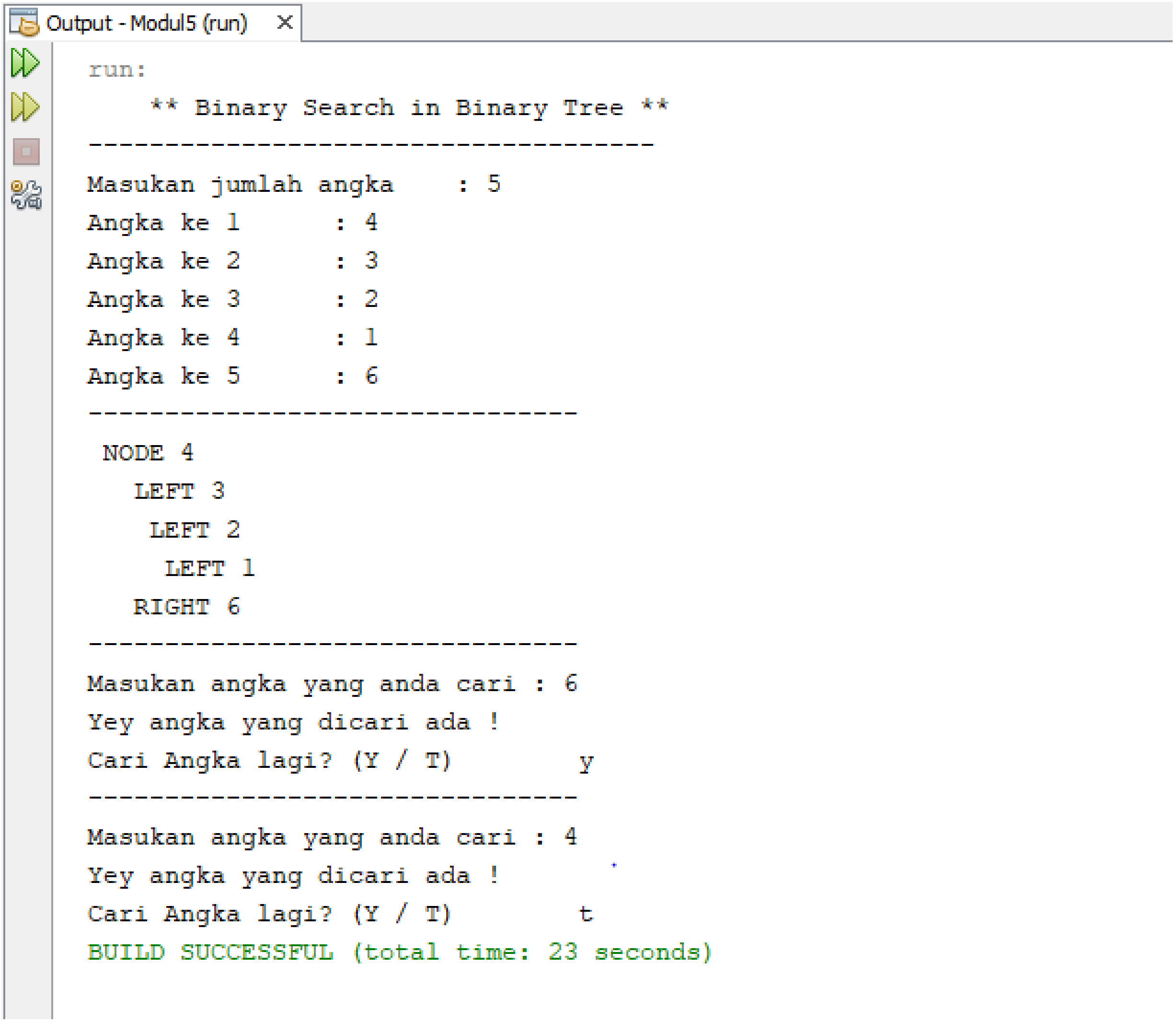


```
    void delete(BinaryTreeNode deleted) {
        if (this.root != null) {
            if (deleted.has_no_child()) {
                if (deleted == this.root) {
                    this.root = null;
                } else {
                    deleted.unset_parent();
                }
            } else if (deleted.only_has_left() || deleted.only_has_right()) {
                BinaryTreeNode replacement = null;
                if (deleted.only_has_left()) {
                    replacement = deleted.left;
                } else {
                    replacement = deleted.right;
                }
                if (deleted == this.root) {
                    this.root = replacement;
                    this.root.unset_parent();

                } else if (deleted.is_left()) {
                    deleted.parent.set_left(replacement);
                    deleted.unset_parent();

                } else if (deleted.is_right()) {
                    deleted.parent.set_right(replacement);
                    deleted.unset_parent();

                }
            } else {
                BinaryTreeNode replacement = deleted.left;
                if (replacement.right != null) {
                    replacement = replacement.most_right_child();
                }
                BinaryTreeNode parent_of_replacement = replacement.parent;
                if (replacement.only_has_right()) {
                    parent_of_replacement.set_left(replacement.right);
                }
                replacement.unset_parent();
                replacement.set_left(deleted.left);
                replacement.set_right(deleted.right);
                if (deleted == this.root) {
                    this.root = replacement;
                } else if (deleted.is_left()) {
                    deleted.parent.set_left(replacement);
                } else if (deleted.is_right()) {
                    deleted.parent.set_right(replacement);
                }
            }
        }
    }

    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;
    }
```



```java

    void unset_parent() {
        if (this.is_left()) {
            parent.left = null;
            this.parent = null;

        } else if (this.is_right()) {
            parent.right = null;
            this.parent = null;

        }
    }

    BinaryTreeNode most_left_child() {
        BinaryTreeNode child = this.left;
        while (child.left != null) {
            child = child.left;

        }
        return child;
    }

    BinaryTreeNode most_right_child() {
        BinaryTreeNode child = this.right;
        while (child.right != null) {
            child = child.right;
        }
        return child;
    }

    void print(String spaces, String label) {
        System.out.println(spaces + label + this.data);
        if (this.left != null) {
            this.left.print(spaces +" ", " LEFT ");
        }
        if (this.right != null) {
            this.right.print(spaces+ " ", " RIGHT ");
        }
    }

    void print() {
        this.print(" ", "NODE ");
    }

    void infix() {
        System.out.print("( ");
        if (this.left != null) {
            left.infix();
        } else {
            System.out.print("null");
        }
        System.out.print(" " + this.data + " ");
        if (this.right != null) {
            right.infix();
        } else {
            System.out.print("null");
        }
        System.out.print(")");
    }

    void prefix() {
        System.out.print(this.data + "(");
        if (this.left != null) {
            left.prefix();
        } else {
            System.out.print("null");
        }
        System.out.print(" ");
        if (this.right != null) {
            right.prefix();
        } else {
            System.out.print("null");
        }
        System.out.print(") ");
    }

    void postfix() {
        System.out.print("( ");
        if (this.left != null) {
            left.postfix();
        } else {
            System.out.print("null");
        }
        System.out.print(" ");
        if (this.right != null) {
            right.postfix();
        } else {
            System.out.print("null");
        }
        System.out.print(")" + this.data);
    }
}
```

2. Output



```
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)
```