รายงาน

เรื่อง โครงสร้างข้อมูลในรูปแบบ AVL Tree

จัดทำโดย

นายณัฐพัฒน์ ไทยเจริญ รหัสนิสิต 6630200161

นายณัฐวุฒิ ทรัพย์มี รหัสนิสิต 6630200179

นายธันย์ธนัช สุขรัตน์ รหัสนิสิต 6630200276

นายนิธิพัฒน์ อินทรมงคล รหัสนิสิต 6630200322

นายปัณณธร ศรีทองกูล รหัสนิสิต 6630200373

เสนอ

ผศ.ดร.จิรวรรณ เจริญสุข

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา

โครงสร้างข้อมูลและขั้นตอนวิธี (01418223-65)

คณะวิทยาศาสตร์ ศรีราชา

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา

การศึกษาภาคต้น ปีการศึกษา 2567

คำนำ

รายงานโครงสร้างข้อมูลและขั้นตอนวิธี โครงสร้างข้อมูลในรูปแบบ AVL Tree นั้นสำเร็จ ขึ้นได้โดยได้รับการช่วยเหลืออย่างดียิ่งจาก ผศ.ดร.จิรวรรณ เจริญสุข ที่ปรึกษารายวานที่ได้ให้คำเสนอแนะ แนวคิดและให้ความรู้ในการจัดทำรายงานเล่มนี้ ตลอดจนการแก้ไขข้อบกพร่องต่างๆ มาโดยตลอดจนโครงงานนี้เสร็จสมบูรณ์ คณะผู้จัดทำจึงกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณ ผศ.ดร.จิรวรรณ เจริญสุข ที่คอยให้ความช่วยเหลือด้านการรวบรวมข้อมูลต่าง ๆ และให้คำปรึกษาในการจัดทำรายงาน และขอบคุณคณะเพื่อนร่วมรุ่น วิทยาการคอมพิวเตอร์ ที่ให้กำลังใจ และข้อมูลในการจัดทำรายงานอีกด้วย

ท้ายที่สุดคณะผู้จัดทำหวังเป็นอย่างยิ่งว่า จะเป็นประโยชน์ต่อการศึกษาค้นคว้าและเป็นประโยชน์ต่อผู้ที่สนใจในเรื่อง โครงสร้างข้อมูลในรูปแบบ AVL Tree

คณะผู้จัดทำ

สารบัญ

โครงสร้างโปรแกรม

1.1 Linkedlist Node

```java
package avl_tree;

public class LinkedListNode {
    int value;
    int height;
    LinkedListNode next; // Reference to the next node in the list
    LinkedListNode left, right; // References for AVL Tree structure

    public LinkedListNode(int d) {
        value = d;
        height = 1;
        next = null; // Initialize next as null
    }
}
```

รูปที่ 1.1 โครงสร้างโปรแกรมของ LinkedlistNode

คลาส LinkedListNode ในโค้ดนี้เป็นส่วนหนึ่งของแพ็กเกจ avl_tree ซึ่งดูเหมือนว่าจะถูกออกแบบมาให้ใช้งานได้สองแบบ คือ:

โครงสร้างของลิงก์ลิสต์ (Linked List):

int value: เก็บค่าของ Node

LinkedListNode next: เป็นตัวชี้ไปยังNodeถัดไปในลิงก์ลิสต์

โครงสร้างของ Tree AVL (AVL Tree):

int height: เก็บข้อมูลความสูงของNodeเมื่อใช้งานใน AVL tree ซึ่งจำเป็นสำหรับการปรับสมดุลของ Tree

LinkedListNode left, right: เป็นตัวชี้ไปยังลูกทางซ้ายและขวาของโหนดใน AVL tree

คอนสตรัคเตอร์:

public LinkedListNode(int d): คอนสตรัคเตอร์นี้จะใช้ค่า d ที่รับเข้ามาเพื่อกำหนดให้กับฟิลด์ value และตั้งค่า height เป็น 1 (ซึ่งเป็นความสูงเริ่มต้นสำหรับNodeใน AVL tree) ส่วน next, left และ right จะถูกตั้งค่าเป็น null เพราะตอนเริ่มNodeนี้ยังไม่ได้เชื่อมต่อกับNodeอื่น

การใช้งาน:

คลาสนี้สามารถใช้งานได้ทั้งในโครงสร้าง ลิงก์ลิสต์ (โดยใช้ตัวชี้ next) และในโครงสร้าง Tree AVL (โดย

ใช้ตัวชี้ left และ right) ทำให้เป็นโครงสร้างข้อมูลที่ยืดหยุ่น แต่ควรระวังหากใช้ทั้งสองโครงสร้างพร้อมกัน เพราะอาจทำให้สับสนได้ในบางกรณี

1.2 โครงสร้าง Linkedlist

การสร้างและจัดการโครงสร้างข้อมูล AVL Tree โดยมีการผสมผสานการทำงานของ Linked List ด้วยคลาส LinkedListAVL_Tree ถูกออกแบบเพื่อจัดการโหนดของ AVL Tree ซึ่งเป็น Treeค้นหาที่มีความสมดุลในตัวเอง (Self-balancing Binary Search Tree) โดยมีฟังก์ชันสำหรับเพิ่ม (insert) ลบ (delete) และค้นหา (find) โหนด พร้อมทั้งฟังก์ชันหมุนซ้ายและขวา (leftRotate, rightRotate) เพื่อปรับสมดุลของ Tree
รายละเอียดฟังก์ชัน:

1.2.1 int height (LinkedListNode N)

คืนค่าความสูงของโหนด N ถ้า N เป็น null จะคืนค่า 0

```
int height(LinkedListNode N) {
    return (N == null) ? 0 : N.height;
}
```

รูปที่ 1.2.1 Method Height

1.2.2 int max (int a, int b)

คืนค่ามากที่สุดระหว่าง a และ b

```
int max(int a, int b) {
    return (a > b) ? a : b;
}
```

รูปที่ 1.2.2 Method max

1.2.3 LinkedListNode rightRotate(LinkedListNode y) และ LinkedListNode leftRotate(LinkedListNode x)

ฟังก์ชันสำหรับหมุนNodeไปทางขวาและซ้ายตามลำดับ ใช้เพื่อปรับสมดุลของ Tree

```
LinkedListNode rightRotate(LinkedListNode y) {
    LinkedListNode x = y.left;
    LinkedListNode T2 = x.right;

    x.right = y;
    y.left = T2;

    y.height = max(height(y.left), height(y.right)) + 1;
    x.height = max(height(x.left), height(x.right)) + 1;

    return x;
}
```

รูปที่ 1.2.3 Method LinkedListNode rightRotate(LinkedListNode y)

```
LinkedListNode leftRotate(LinkedListNode x) {
    LinkedListNode y = x.right;
    LinkedListNode T2 = y.left;

    y.left = x;
    x.right = T2;

    x.height = max(height(x.left), height(x.right)) + 1;
    y.height = max(height(y.left), height(y.right)) + 1;

    return y;
}
```

รูปที่ 1.2.3 Method LinkedListNode rightRotate(LinkedListNode x)

1.2.4 int getBalance(LinkedListNode N)

คืนค่าความสมดุลของโหนด N โดยการลบความสูงของ Treeฝั่งขวาออกจากความสูงของ Treeฝั่งซ้าย

```
int getBalance(LinkedListNode N) {
    return (N == null) ? 0 : height(N.left) - height(N.right);
}
```

รูปที่ 1.2.4 Method getBalance

1.2.5 LinkedListNode insert(LinkedListNode node, int value)

ฟังก์ชันสำหรับเพิ่มโหนดใหม่เข้าไปใน Tree AVL ตามค่าของ value และปรับสมดุลหลังจากการแทรก

```
LinkedListNode insert(LinkedListNode node, int value) {
    if (node == null) {
        return new LinkedListNode(value);
    }

    if (value < node.value) {
        node.left = insert(node.left, value);
    } else if (value > node.value) {
        node.right = insert(node.right, value);
    } else {
        return node; // Duplicates not allowed
    }

    node.height = 1 + max(height(node.left), height(node.right));
    int balance = getBalance(node);

    if (balance > 1 && value < node.left.value) {
        return rightRotate(node);
    }
    if (balance < -1 && value > node.right.value) {
        return leftRotate(node);
    }
    if (balance > 1 && value > node.left.value) {
        node.left = leftRotate(node.left);
        return rightRotate(node);
    }
    if (balance < -1 && value < node.right.value) {
        node.right = rightRotate(node.right);
        return leftRotate(node);
    }

    return node;
}
```

รูปที่ 1.2.5 Method Linkedlistnode insert

1.2.6 LinkedListNode deleteNode(LinkedListNode root, int value)

ฟังก์ชันสำหรับลบโหนดจาก Tree AVL โดยใช้ค่าของ value และปรับสมดุลหลังจากการลบ

```
LinkedListNode deleteNode(LinkedListNode root, int value) {
        if (root == null) {
            return root;
        }

        if (value < root.value) {
            root.left = deleteNode(root.left, value);
        } else if (value > root.value) {
            root.right = deleteNode(root.right, value);
        } else {
            if ((root.left == null) || (root.right == null)) {
                LinkedListNode temp = (root.left != null) ? root.left : root.right;
                return temp; // Return the non-null child or null
            } else {
                LinkedListNode temp = minValueNode(root.right);
                root.value = temp.value;
                root.right = deleteNode(root.right, temp.value);
            }
        }

        if (root == null) {
            return root;
        }

        root.height = max(height(root.left), height(root.right)) + 1;
        int balance = getBalance(root);

        if (balance > 1 && getBalance(root.left) >= 0) {
            return rightRotate(root);
        }
        if (balance > 1 && getBalance(root.left) < 0) {
            root.left = leftRotate(root.left);
            return rightRotate(root);
        }
        if (balance < -1 && getBalance(root.right) <= 0) {
            return leftRotate(root);
        }
        if (balance < -1 && getBalance(root.right) > 0) {
            root.right = rightRotate(root.right);
            return leftRotate(root);
        }

        return root;
    }
```

รูปที่ 1.2.6 Method deleteNode

1.2.7 LinkedListNode minValueNode(LinkedListNode node)

ฟังก์ชันนี้จะค้นหาและคืนค่าโหนดที่มีค่าต่ำที่สุดใน Treeโดยการเดินไปทางซ้ายสุดของโหนดที่ถูกส่งเข้ามา

```
LinkedListNode minValueNode(LinkedListNode node) {
    LinkedListNode current = node;
    while (current.left != null) {
        current = current.left;
    }
    return current;
}
```

รูปที่ 1.2.7 Method LinkedListNode minValueNode

1.2.8 boolean find(LinkedListNode node, int value)

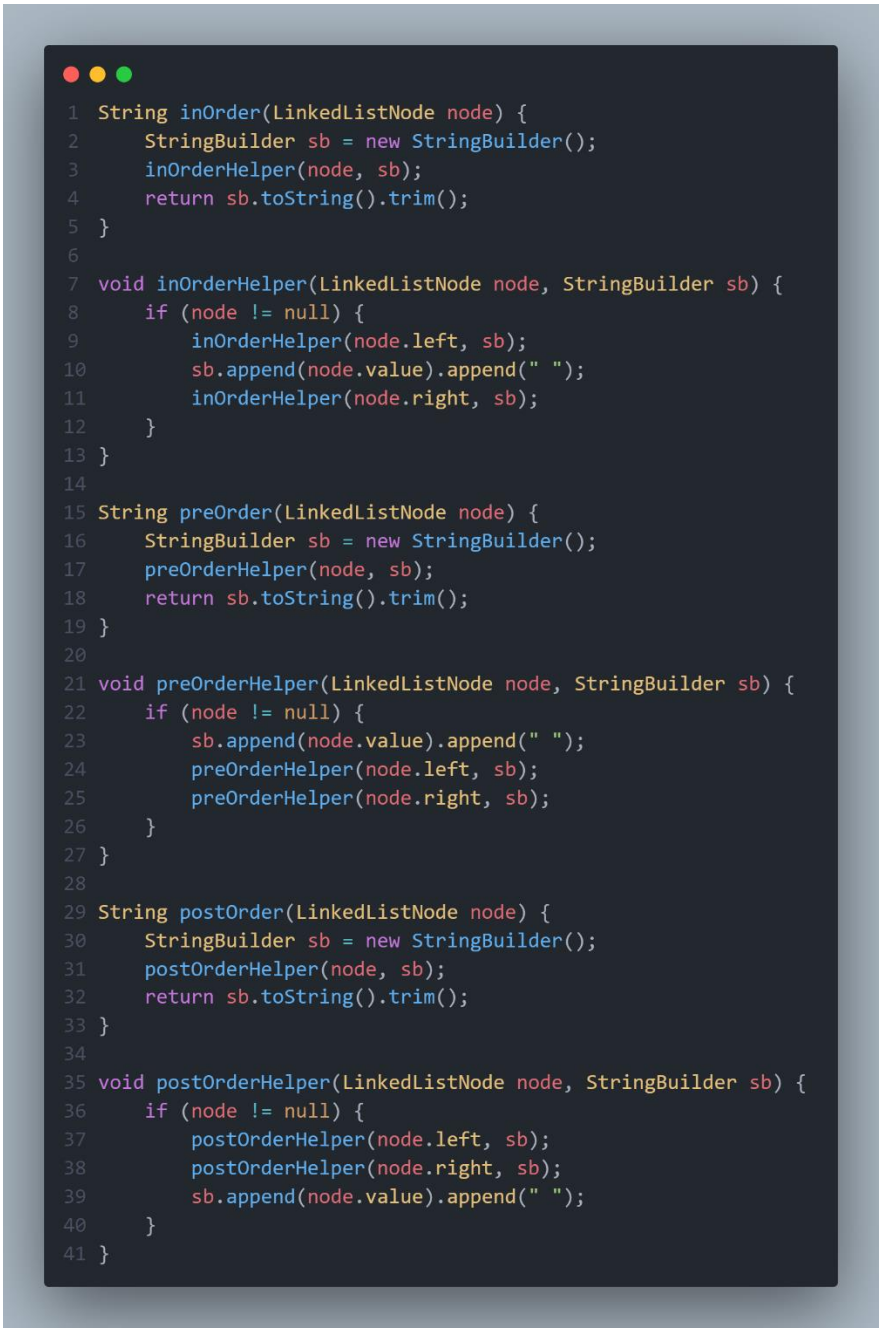
ฟังก์ชันสำหรับค้นหาค่าที่ตรงกับ value ใน Tree คืนค่า true ถ้าพบ และ false ถ้าไม่พบ

```
boolean find(LinkedListNode node, int value) {
    if (node == null) {
        return false;
    }
    if (node.value == value) {
        return true;
    }
    return (value < node.value) ? find(node.left, value) : find(node.right, value);
}
```

รูปที่ 1.2.8 Method find

1.2.9 ฟังก์ชันการเดินทางใน Tree (Traversal Functions)

String inOrder(LinkedListNode node), String preOrder(LinkedListNode node), String postOrder(LinkedListNode node):ฟังก์ชันสำหรับการเดินทางใน Treeแบบเรียงลำดับ (in-order), ก่อนลำดับ (pre-order), และหลังลำดับ (post-order) โดยคืนค่าเป็นสตริง



```java
String inOrder(LinkedListNode node) {
    StringBuilder sb = new StringBuilder();
    inOrderHelper(node, sb);
    return sb.toString().trim();
}

void inOrderHelper(LinkedListNode node, StringBuilder sb) {
    if (node != null) {
        inOrderHelper(node.left, sb);
        sb.append(node.value).append(" ");
        inOrderHelper(node.right, sb);
    }
}

String preOrder(LinkedListNode node) {
    StringBuilder sb = new StringBuilder();
    preOrderHelper(node, sb);
    return sb.toString().trim();
}

void preOrderHelper(LinkedListNode node, StringBuilder sb) {
    if (node != null) {
        sb.append(node.value).append(" ");
        preOrderHelper(node.left, sb);
        preOrderHelper(node.right, sb);
    }
}

String postOrder(LinkedListNode node) {
    StringBuilder sb = new StringBuilder();
    postOrderHelper(node, sb);
    return sb.toString().trim();
}

void postOrderHelper(LinkedListNode node, StringBuilder sb) {
    if (node != null) {
        postOrderHelper(node.left, sb);
        postOrderHelper(node.right, sb);
        sb.append(node.value).append(" ");
    }
}
```

รูปที่ 1.2.9 Method การเรียงลำดับ

1.2.10 String printTree()

ฟังก์ชันสำหรับพิมพ์โครงสร้าง Treeในรูปแบบ pre-order, in-order, และ post-order ในรูปของสตริง

```
String printTree() {
    return "Pre-order: " + preOrder(root) + "\n"
         + "In-order: " + inOrder(root) + "\n"
         + "Post-order: " + postOrder(root);
}
```

รูปที่ 1.2.10 Method print

1.3 โครงสร้างโปรแกรมการแสดงผล

คลาส TreeDisplayPanel ซึ่งเป็นส่วนหนึ่งของแพ็กเกจ avl_tree โดยใช้ Java Swing เพื่อแสดงผล Tree AVL (AVL Tree) ในรูปแบบกราฟิกใน GUI

```
package avl_tree;

import javax.swing.*;
import java.awt.*;

public class TreeDisplayPanel extends JPanel {
    private LinkedListNode root;

    public void setTree(LinkedListNode root) {
        this.root = root;
        repaint(); // Refresh the panel
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (root != null) {
            drawTree(g, root, getWidth() / 2, 30, 100);
        }
    }

    private void drawTree(Graphics g, LinkedListNode node, int x, int y, int xOffset) {
        if (node != null) {
            g.drawOval(x - 15, y - 15, 30, 30);
            g.drawString(String.valueOf(node.value), x - 5, y + 5);

            if (node.left != null) {
                g.drawLine(x, y, x - xOffset, y + 50);
                drawTree(g, node.left, x - xOffset, y + 50, xOffset / 2);
            }

            if (node.right != null) {
                g.drawLine(x, y, x + xOffset, y + 50);
                drawTree(g, node.right, x + xOffset, y + 50, xOffset / 2);
            }
        }
    }
}
```

รูปที่ 1.3 โครงสร้าง GUI AVL Tree

รายละเอียดโค้ด:

คลาส TreeDisplayPanel สืบทอดจาก JPanel
JPanel เป็นคลาสใน Java Swing ที่ใช้สำหรับสร้างพื้นที่วาดรูปหรือวางองค์ประกอบต่างๆ ใน GUI โค้ดนี้ใช้ JPanel เพื่อแสดงโหนดของ Tree AVL

ฟิลด์ LinkedListNode root:
ตัวแปร root เป็นโหนดรากของ Tree AVL ที่จะถูกแสดงผลในกราฟิก

เมธอด setTree(LinkedListNode root):
ใช้สำหรับกำหนด Treeที่จะถูกแสดงผล โดยรับโหนดราก (root) ของ Tree จากนั้นจะเรียกใช้ repaint() เพื่อสั่งให้พื้นที่วาดภาพของ JPanel ทำการรีเฟรชและแสดงผลใหม่

เมธอด paintComponent(Graphics g)
เมธอดนี้ถูกเรียกโดยอัตโนมัติเมื่อ JPanel ต้องการการวาดภาพใหม่หรือรีเฟรช ซึ่งในที่นี้มันจะเรียกใช้ฟังก์ชัน drawTree() เพื่อวาด Treeในตำแหน่งที่กำหนด

ถ้า root ไม่เป็น null จะเริ่มต้นวาด Treeจากจุดกึ่งกลางของ JPanel (โดยใช้ getWidth() / 2 สำหรับตำแหน่ง x) และเริ่มต้นที่ตำแหน่ง y = 30

เมธอด drawTree(Graphics g, LinkedListNode node, int x, int y, int xOffset)
เป็นเมธอดหลักสำหรับการวาด Tree AVL แบบกราฟิก โดยวาดโหนดแต่ละโหนดในตำแหน่ง x, y ที่ระบุ

การวาดโหนด
วาดวงกลม (drawOval) ที่ตำแหน่ง (x - 15, y - 15) ขนาด 30x30 เพื่อแทนโหนด
วาดค่าของโหนด (node.value) ด้วยการแสดงสตริงที่ตรงกลางของวงกลม (drawString)

การวาดเส้นเชื่อมต่อ (Edge)
ถ้าโหนดนั้นมีลูกทางซ้าย (node.left), จะวาดเส้นจากโหนดปัจจุบันไปยังตำแหน่งลูกทางซ้ายที่อยู่ต่ำลงและเยื้องไปทางซ้าย (x - xOffset, y + 50)
ถ้าโหนดนั้นมีลูกทางขวา (node.right), จะวาดเส้นไปยังตำแหน่งลูกทางขวาในลักษณะเดียวกัน (x + xOffset, y + 50)

การลดระยะห่าง: ในการวาดโหนดลูกของโหนดปัจจุบัน ระยะทางแนวนอน (xOffset) จะถูกลดลงครึ่งหนึ่งในแต่ละระดับเพื่อให้ Treeถูกจัดเรียงอย่างสวยงาม

การทำงานโดยรวม

โค้ดนี้ช่วยให้เราสามารถแสดงผล Tree AVL ได้ในรูปแบบกราฟิก ซึ่งจะแสดงโหนดเป็นวงกลมและมีเส้นเชื่อมโยงระหว่างโหนดแม่และโหนดลูก โดยใช้วิธีการวาดภาพผ่าน Graphics ของ Java Swing

1.4 โครงสร้างโปรแกรม AVL Tree แบบสมบูรณ์

Java Swing GUI ที่ใช้สำหรับการจัดการกับโครงสร้าง Tree (AVL Tree) โดยใช้คอนเทนเนอร์ JPanel ของ Swing ในการแสดงผล และใช้ KeyListeners เพื่อตอบสนองการกดปุ่มบนแป้นพิมพ์

```java
public class Base extends javax.swing.JPanel {

    private LinkedListAVL_Tree avlTree = new LinkedListAVL_Tree();
    private TreeDisplayPanel displayPanel = new TreeDisplayPanel();

    public Base() {
        initComponents();
        displayPanel.setBackground(Color.WHITE);
        Display.setLayout(new BorderLayout());
        Display.add(displayPanel, BorderLayout.CENTER);

        // Ensure that ShowTraverse has space
        addKeyListeners();
    }

    private void addKeyListeners() {
        insertBox.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
                if (evt.getKeyCode() == java.awt.event.KeyEvent.VK_ENTER) {
                    insert_BtnActionPerformed(null);
                }
            }
        });

        deleteBox.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
                if (evt.getKeyCode() == java.awt.event.KeyEvent.VK_ENTER) {
                    delete_BtnActionPerformed(null);
                }
            }
        });

        findBox.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
                if (evt.getKeyCode() == java.awt.event.KeyEvent.VK_ENTER) {
                    find_BtnActionPerformed(null);
                }
            }
        });
    }
```

รูปที่ 1.4.1 – 1.4.4 การสร้างฟังก์ชั่น GUI รูปแบบที่ 1

1.4.1 การสร้างคลาส Base

คลาส Base เป็นการขยายคลาส JPanel ของ Swing ทำให้สามารถแสดงผลกราฟิกใน GUI ได้

1.4.2 การประกาศตัวแปร

avlTree: ตัวแปรที่เก็บโครงสร้าง Tree AVL

displayPanel: ตัวแปรที่ใช้แสดงผลของโครงสร้าง Treeบนหน้าจอ

1.4.3 คอนสตรัคเตอร์ของคลาส Base

เรียกใช้ initComponents() เพื่อเตรียมการเริ่มต้น

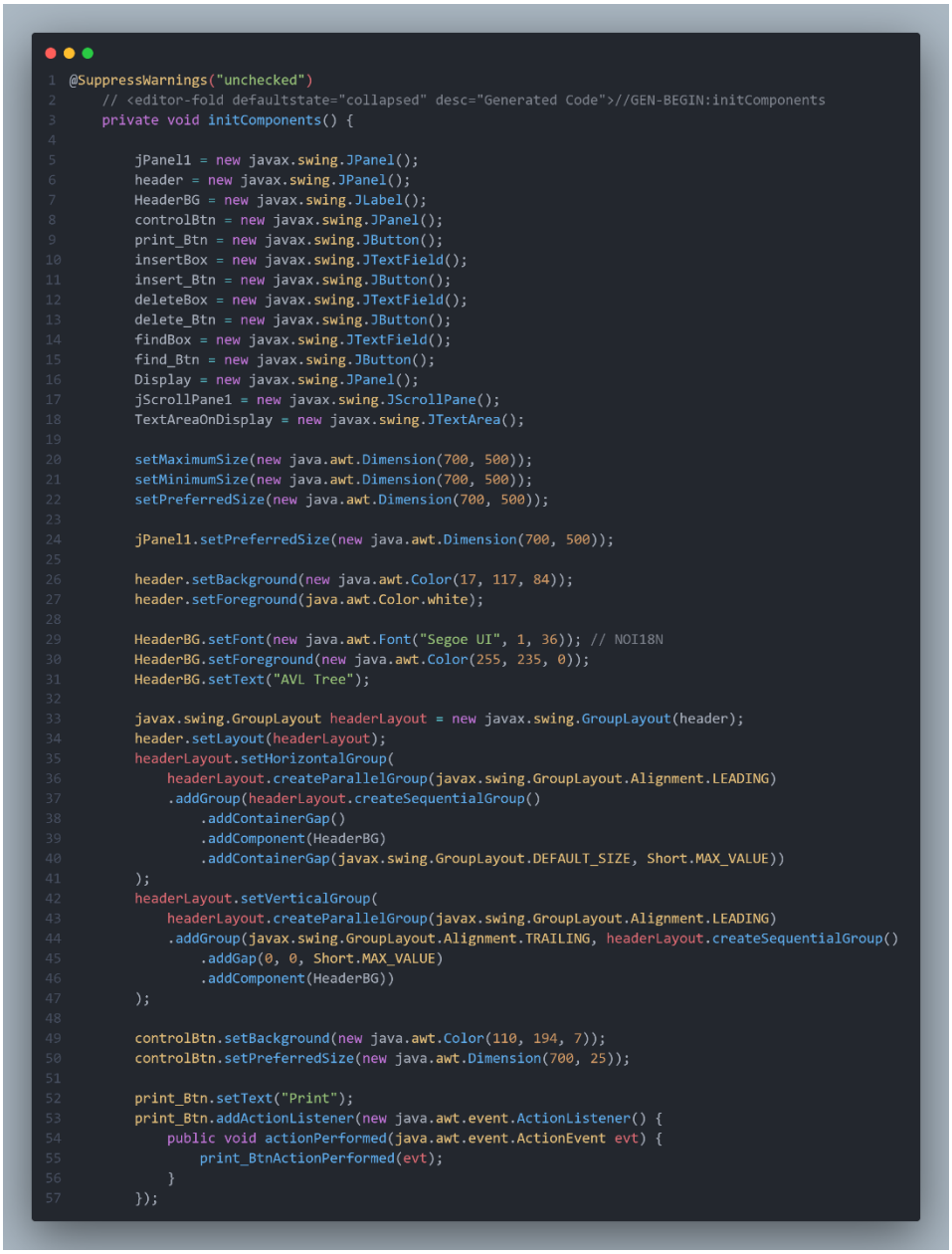
ตั้งค่า displayPanel ให้มีพื้นหลังสีขาว และตั้งค่า layout ของ Display เป็น BorderLayout

เพิ่ม displayPanel ในตำแหน่ง CENTER ของ layout

เรียกใช้ addKeyListeners() เพื่อเพิ่มตัวดักฟังการกดแป้นพิมพ์

1.4.4 ฟังก์ชัน addKeyListeners()

ในฟังก์ชันนี้ มีการเพิ่มตัวดักฟัง (KeyListener) ให้กับกล่องข้อความต่างๆ (เช่น insertBox, deleteBox, findBox) เพื่อฟังการกดแป้นพิมพ์ โดยเฉพาะการกดปุ่ม Enter (VK_ENTER)
ถ้าผู้ใช้กดปุ่ม Enter ขณะที่อยู่ในกล่องข้อความนั้นๆ จะเรียกใช้ฟังก์ชันที่เกี่ยวข้อง เช่น insert_BtnActionPerformed(), delete_BtnActionPerformed(), หรือ find_BtnActionPerformed()



```
@SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        header = new javax.swing.JPanel();
        HeaderBG = new javax.swing.JLabel();
        controlBtn = new javax.swing.JPanel();
        print_Btn = new javax.swing.JButton();
        insertBox = new javax.swing.JTextField();
        insert_Btn = new javax.swing.JButton();
        deleteBox = new javax.swing.JTextField();
        delete_Btn = new javax.swing.JButton();
        findBox = new javax.swing.JTextField();
        find_Btn = new javax.swing.JButton();
        Display = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        TextAreaOnDisplay = new javax.swing.JTextArea();

        setMaximumSize(new java.awt.Dimension(700, 500));
        setMinimumSize(new java.awt.Dimension(700, 500));
        setPreferredSize(new java.awt.Dimension(700, 500));

        jPanel1.setPreferredSize(new java.awt.Dimension(700, 500));

        header.setBackground(new java.awt.Color(17, 117, 84));
        header.setForeground(java.awt.Color.white);

        HeaderBG.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N
        HeaderBG.setForeground(new java.awt.Color(255, 235, 0));
        HeaderBG.setText("AVL Tree");

        javax.swing.GroupLayout headerLayout = new javax.swing.GroupLayout(header);
        header.setLayout(headerLayout);
        headerLayout.setHorizontalGroup(
            headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(headerLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(HeaderBG)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        headerLayout.setVerticalGroup(
            headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, headerLayout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(HeaderBG))
        );

        controlBtn.setBackground(new java.awt.Color(110, 194, 7));
        controlBtn.setPreferredSize(new java.awt.Dimension(700, 25));

        print_Btn.setText("Print");
        print_Btn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                print_BtnActionPerformed(evt);
            }
        });
```

รูปที่ 1.4.5 – 1.4.9 การสร้างฟังก์ชั่น GUI

1.4.5 การสร้างคอมโพเนนต์ใหม่

สร้างตัวแปร Swing คอมโพเนนต์ต่างๆ เช่น JPanel, JLabel, JButton, JTextField, และ JTextAre เพื่อใช้ใน UI

1.4.6 กำหนดขนาดและการแสดงผล

กำหนดขนาดของคอมโพเนนต์ต่างๆ เช่น jPanel1 ให้มีขนาดคงที่ 700x500 พิกเซล

1.4.7 การกำหนดค่าของ header และ HeaderBG

ตั้งค่า HeaderBG ซึ่งเป็น JLabel เพื่อแสดงข้อความ "AVL Tree" โดยใช้ฟอนต์ "Segoe UI" ขนาด 36 และสีฟอนต์สีเหลือง

1.4.8 การจัดวางเลย์เอาท์ของ header

ใช้ GroupLayout เพื่อจัดการวางเลย์เอาท์ของ header โดยมีการกำหนดทั้งแนวตั้งและแนวนอน ซึ่งจะวางตำแหน่ง HeaderBG ให้อยู่ในตำแหน่งตรงกลาง

1.4.9 การกำหนดค่าปุ่ม print_Btn

ตั้งข้อความของปุ่ม print_Btn ให้เป็น "Print" และเพิ่ม ActionListener เพื่อตรวจจับการกดปุ่ม เมื่อผู้ใช้กดปุ่มจะเรียกใช้ฟังก์ชัน print_BtnActionPerformed()

```
insertBox.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                insertBoxMouseEntered(evt);
            }
        });
        insertBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                insertBoxActionPerformed(evt);
            }
        });

        insert_Btn.setText("Insert");
        insert_Btn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                insert_BtnActionPerformed(evt);
            }
        });

        deleteBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                deleteBoxActionPerformed(evt);
            }
        });

        delete_Btn.setText("Delete");
        delete_Btn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                delete_BtnActionPerformed(evt);
            }
        });

        findBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                findBoxActionPerformed(evt);
            }
        });

        find_Btn.setText("Find");
        find_Btn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                find_BtnActionPerformed(evt);
            }
        });
```

รูปที่ 1.4.10 – 1.4.16 การสร้างฟังก์ชั่น GUI รูปแบบที่ 3

1.4.10 การเพิ่ม MouseListener ให้กับ insertBox

เมื่อตรวจจับการกระทำใน insertBox (เช่น การกดปุ่ม Enter ขณะที่พิมพ์ข้อความในกล่อง) ฟังก์ชัน insertBoxActionPerformed(evt) จะถูกเรียกเพื่อดำเนินการตามที่กำหนด

1.4.11 การเพิ่ม ActionListener ให้กับ insertBox

เมื่อตรวจจับการกระทำใน insertBox (เช่น การกดปุ่ม Enter ขณะที่พิมพ์ข้อความในกล่อง) ฟังก์ชัน insertBoxActionPerformed(evt) จะถูกเรียกเพื่อดำเนินการตามที่กำหนด

1.4.12 การตั้งข้อความและเพิ่ม ActionListener ให้กับปุ่ม insert_Btn

ข้อความบนปุ่ม insert_Btn ถูกตั้งให้เป็น "Insert" และเมื่อผู้ใช้คลิกปุ่มนี้ ฟังก์ชัน insert_BtnActionPerformed(evt) จะถูกเรียกเพื่อดำเนินการแทรกข้อมูล

1.4.13 การเพิ่ม ActionListener ให้กับ deleteBox

เมื่อมีการกระทำเกิดขึ้นใน deleteBox (เช่นการกด Enter) ฟังก์ชัน deleteBoxActionPerformed(evt) จะถูกเรียก

1.4.14 การตั้งข้อความและเพิ่ม ActionListener ให้กับปุ่ม delete_Btn

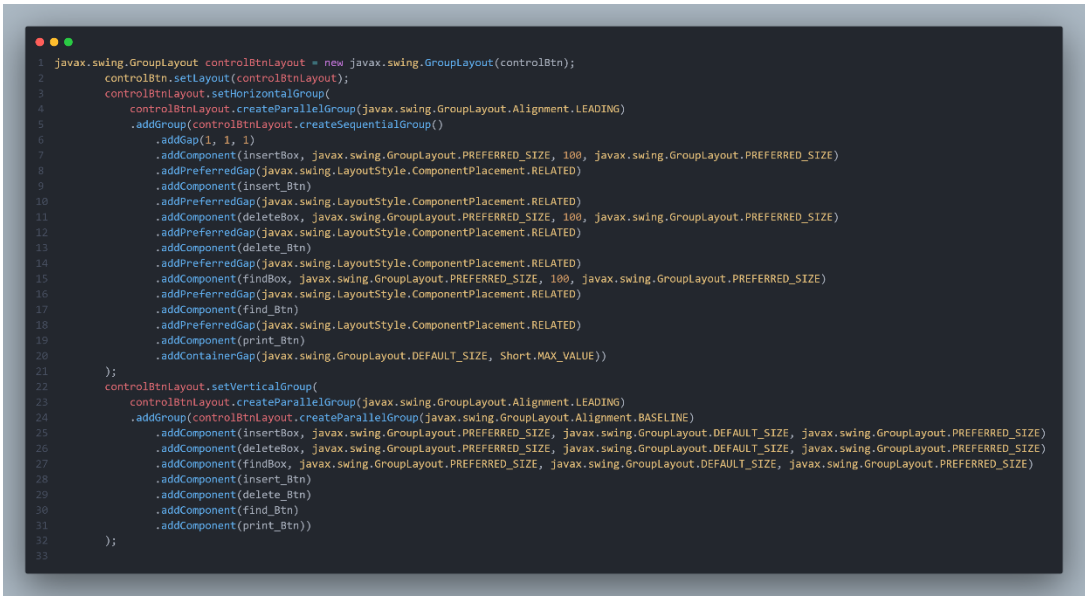
ข้อความบนปุ่ม delete_Btn ถูกตั้งเป็น "Delete" และเมื่อคลิกปุ่มนี้ ฟังก์ชัน delete_BtnActionPerformed(evt) จะถูกเรียกเพื่อดำเนินการลบข้อมูล

1.4.15 การเพิ่ม ActionListener ให้กับ findBox

เมื่อมีการกระทำใน findBox ฟังก์ชัน findBoxActionPerformed(evt) จะถูกเรียกใช้งาน

1.4.16 การตั้งข้อความและเพิ่ม ActionListener ให้กับปุ่ม find_Btn

ข้อความของปุ่ม find_Btn ถูกตั้งเป็น "Find" และเมื่อคลิกปุ่มนี้ ฟังก์ชัน find_BtnActionPerformed(evt) จะถูกเรียกเพื่อดำเนินการค้นหาข้อมูล



```
javax.swing.GroupLayout controlBtnLayout = new javax.swing.GroupLayout(controlBtn);
        controlBtn.setLayout(controlBtnLayout);
        controlBtnLayout.setHorizontalGroup(
            controlBtnLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(controlBtnLayout.createSequentialGroup()
                .addGap(1, 1, 1)
                .addComponent(insertBox, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(insert_Btn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(deleteBox, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(delete_Btn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(findBox, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(find_Btn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(print_Btn)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        controlBtnLayout.setVerticalGroup(
            controlBtnLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(controlBtnLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(insertBox, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(deleteBox, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(findBox, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(insert_Btn)
                .addComponent(delete_Btn)
                .addComponent(find_Btn)
                .addComponent(print_Btn))
        );

```

รูปที่ 1.4.17 – 1.4.19 การสร้างฟังก์ชั่น GUI รูปแบบที่ 4

1.4.17 สร้าง GroupLayout สำหรับ controlBtn

สร้างออบเจ็กต์ GroupLayout ชื่อ controlBtnLayout และกำหนดให้ใช้ Layout นี้กับ controlBtn

1.4.18 กำหนด Group สำหรับการจัดวางแนวนอน (setHorizontalGroup)

ในแนวนอน (HorizontalGroup):

มีการกำหนดให้วางคอมโพเนนต์ในลักษณะ SequentialGroup คือเรียงกันเป็นแถว (จากซ้ายไปขวา) คอมโพเนนต์จะถูกเพิ่มเข้ามาเรียงกัน ได้แก่ insertBox, insert_Btn, deleteBox, delete_Btn, findBox, find_Btn, และ print_Btn

addGap(1, 1, 1) และ addPreferredGap ใช้เพื่อเพิ่มช่องว่างระหว่างคอมโพเนนต์แต่ละตัว

1.4.19 กำหนด Group สำหรับการจัดวางแนวตั้ง (setVerticalGroup)

ในแนวตั้ง (VerticalGroup):

ใช้ ParallelGroup เพื่อจัดวางคอมโพเนนต์ในแนวขนาน (เรียงในแนวตั้งเท่ากัน) คอมโพเนนต์ทั้งหมด ได้แก่ insertBox, deleteBox, findBox, insert_Btn, delete_Btn, find_Btn, และ print_Btn จะถูกวางเรียงในแนวตั้งอย่างสมดุล โดยทั้งหมดจะมีขนาดเท่ากัน (PREFERRED_SIZE)

```
Display.setBackground(new java.awt.Color(255, 255, 255));
        Display.setPreferredSize(new java.awt.Dimension(700, 500));

        TextAreaOnDisplay.setColumns(20);
        TextAreaOnDisplay.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
        TextAreaOnDisplay.setRows(5);
        jScrollPane1.setViewportView(TextAreaOnDisplay);

        javax.swing.GroupLayout DisplayLayout = new javax.swing.GroupLayout(Display);
        Display.setLayout(DisplayLayout);
        DisplayLayout.setHorizontalGroup(
            DisplayLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(DisplayLayout.createSequentialGroup()
                .addGap(734, 734, 734)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
        );
        DisplayLayout.setVerticalGroup(
            DisplayLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 427, Short.MAX_VALUE)
        );
```

รูปที่ 1.4.20 – 1.4.22 การสร้างฟังก์ชั่น GUI รูปแบบที่ 5

1.4.20 การตั้งค่าแผงแสดงผล (Display Panel)

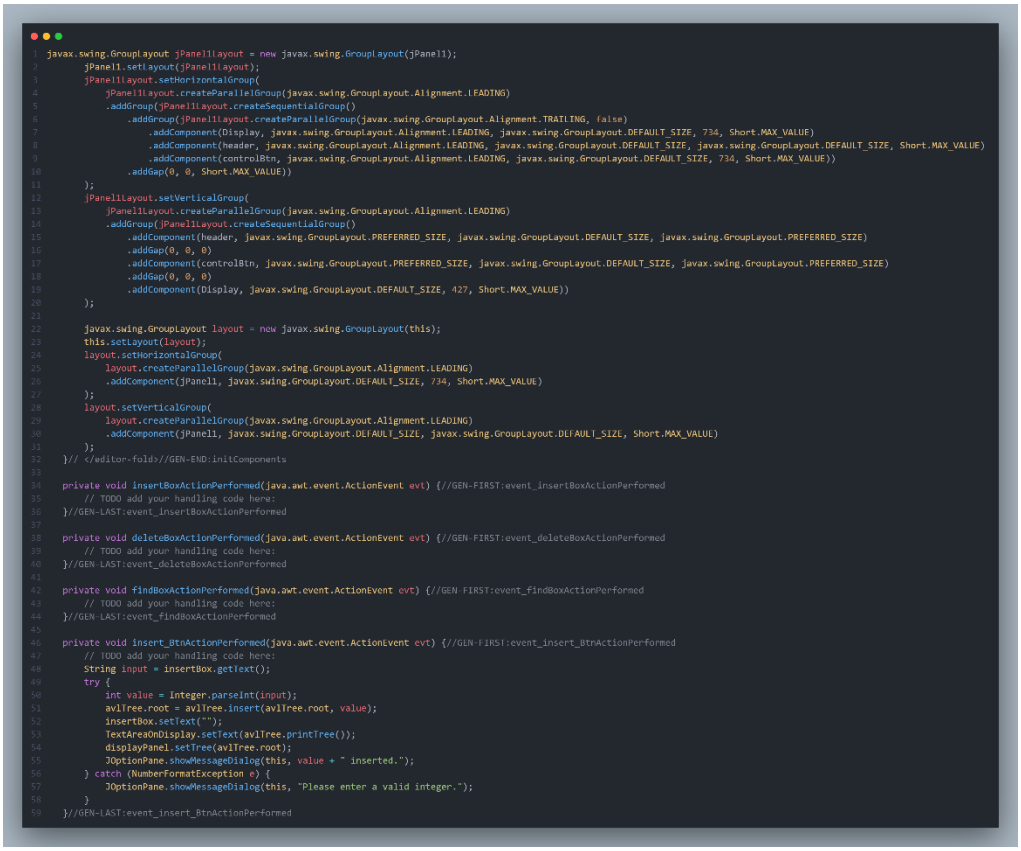
ตั้งค่าสีพื้นหลังของแผงแสดงผลให้เป็นสีขาว (255, 255, 255 คือตัวแทนของค่าสี RGB สำหรับสีขาว) ขนาดที่ต้องการของแผงแสดงผลถูกตั้งค่าให้กว้าง 700 พิกเซล และสูง 500 พิกเซล

1.4.21 การตั้งค่า TextArea

TextAreaOnDisplay ซึ่งเป็น TextArea ถูกตั้งค่าให้มีจำนวนคอลัมน์ 20 คอลัมน์ฟอนต์ที่ใช้คือ Segoe UI โดยมีสไตล์เป็น 1 (ซึ่งโดยทั่วไปหมายถึงตัวหนา) และขนาดฟอนต์เป็น 24จำนวนแถวถูกตั้งค่าเป็น 5 แถว TextArea นี้จะถูกใส่เข้าไปใน JScrollPane (jScrollPane1) เพื่อเพิ่มฟังก์ชันการเลื่อน (scroll)

1.4.22 การจัดการ Layout ของแผงแสดงผล

ใช้ GroupLayout ในการจัดการ layout ของแผงแสดงผล (Display)มีการกำหนดกลุ่มแนวนอน (Horizontal Group) และแนวตั้ง (Vertical Group) เพื่อควบคุมการวางตำแหน่งของ JScrollPane ที่ห่อหุ้ม TextArea ช่องว่างในกลุ่มแนวนอนถูกตั้งค่าเป็น 734 พิกเซล ส่วนขนาดของ JScrollPane ในแนวตั้งถูกกำหนดให้มีขนาด 427 พิกเซล



```
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                    .addComponent(Display, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 734, Short.MAX_VALUE)
                    .addComponent(header, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(controlBtn, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 734, Short.MAX_VALUE))
                .addGap(0, 0, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addComponent(header, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, 0)
                .addComponent(controlBtn, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, 0)
                .addComponent(Display, javax.swing.GroupLayout.DEFAULT_SIZE, 427, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, 734, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
    }// </editor-fold//GEN-END:initComponents

    private void insertBoxActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_insertBoxActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_insertBoxActionPerformed

    private void deleteBoxActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_deleteBoxActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_deleteBoxActionPerformed

    private void findBoxActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_findBoxActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_findBoxActionPerformed

    private void insert_BtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_insert_BtnActionPerformed
        // TODO add your handling code here:
        String input = insertBox.getText();
        try {
            int value = Integer.parseInt(input);
            avlTree.root = avlTree.insert(avlTree.root, value);
            insertBox.setText("");
            TextAreaOnDisplay.setText(avlTree.printTree());
            displayPanel.setTree(avlTree.root);
            JOptionPane.showMessageDialog(this, value + " inserted.");
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter a valid integer.");
        }
    }//GEN-LAST:event_insert_BtnActionPerformed
```

รูปที่ 1.4.23 – 1.4.25 การสร้างฟังก์ชั่น GUI รูปแบบที่ 6

1.4.23 การจัดการแผงด้วย GroupLayout

การตั้งค่า **GroupLayout** สำหรับแผง (jPanel1) โดยกำหนดรูปแบบในแกนแนวนอน (setHorizontalGroup) และแนวตั้ง (setVerticalGroup)

ในแนวนอนมีการจัดวางองค์ประกอบต่าง ๆ เช่น Display, header, และ controlBtn ให้มีขนาดเท่ากัน (734 พิกเซล)

ในแนวตั้งมีการจัดเรียงองค์ประกอบแบบเรียงต่อกัน (SequentialGroup) ซึ่งแสดง header ก่อน ตามด้วยปุ่มควบคุม (controlBtn) และ Display

1.4.24 การจัดการ Layout หลัก

โค้ดส่วนนี้ใช้ GroupLayout อีกครั้งในการตั้งค่าเลย์เอาต์สำหรับหน้าต่างหลัก (this) ในแนวนอนและแนวตั้งมีการเพิ่มแผง jPanel1 ทั้งหมดเข้าไป ซึ่งจะเป็นพื้นที่แสดงผลหลักของหน้าต่าง GUI นี้

1.4.25 การจัดการ Action Events ของปุ่มต่าง ๆ

ฟังก์ชันเหล่านี้เป็นการตอบสนองต่อการคลิกปุ่มต่าง ๆ ใน GUI

ฟังก์ชัน insertBoxActionPerformed, deleteBoxActionPerformed, และ findBoxActionPerformed ยังไม่มีโค้ดการทำงานจริง ๆ

ฟังก์ชัน insert_BtnActionPerformed ทำงานเมื่อปุ่ม Insert ถูกกด โดยจะทำการรับค่าจากผู้ใช้ แปลงค่าที่ใส่เข้ามาเป็นเลขจำนวนเต็ม และแทรกค่าเข้าไปในโครงสร้าง Tree AVL (avlTree) จากนั้นจะแสดงผลโครงสร้าง Tree ใน TextAreaOnDisplay และแจ้งเตือนเมื่อการแทรกสำเร็จ

```
private void delete_BtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_delete_BtnActionPerformed
        // TODO add your handling code here:
        String input = deleteBox.getText();
        try {
            int value = Integer.parseInt(input);
            avlTree.root = avlTree.deleteNode(avlTree.root, value);
            deleteBox.setText("");
            TextAreaOnDisplay.setText(avlTree.printTree());
            displayPanel.setTree(avlTree.root);
            JOptionPane.showMessageDialog(this, value + " deleted.");
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter a valid integer.");
        }
    }//GEN-LAST:event_delete_BtnActionPerformed

    private void find_BtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_find_BtnActionPerformed
        // TODO add your handling code here:
        String input = findBox.getText();
        try {
            int value = Integer.parseInt(input);
            boolean found = avlTree.find(avlTree.root, value);
            JOptionPane.showMessageDialog(this, found ? (value + " found in the tree.") : (value + " not found in the tree."));
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter a valid integer.");
        }
    }//GEN-LAST:event_find_BtnActionPerformed

    private void print_BtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_print_BtnActionPerformed
        // TODO add your handling code here:
        String preOrderTraversal = avlTree.preOrder(avlTree.root);
        String inOrderTraversal = avlTree.inOrder(avlTree.root);
        String postOrderTraversal = avlTree.postOrder(avlTree.root);

        // Print results to the terminal
        System.out.println("Pre-order: " + preOrderTraversal);
        System.out.println("In-order: " + inOrderTraversal);
        System.out.println("Post-order: " + postOrderTraversal);

        // Optionally display a message dialog
        JOptionPane.showMessageDialog(this, "Traversal output printed in the terminal.");
    }//GEN-LAST:event_print_BtnActionPerformed

    private void insertBoxMouseEntered(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_insertBoxMouseEntered
        // TODO add your handling code here:
    }//GEN-LAST:event_insertBoxMouseEntered
```

รูปที่ 1.4.26 – 1.4.29 การสร้างฟังก์ชั่น GUI รูปแบบที่ 7

1.4.26 ลบNodeออกจาก Tree (delete_BtnActionPerformed)

ฟังก์ชันนี้ทำงานเมื่อผู้ใช้กดปุ่ม Delete เพื่อลบโหนดออกจาก Tree AVL
รับค่าจากช่อง deleteBox ซึ่งเป็นตัวเลขที่ผู้ใช้ใส่เข้ามา และพยายามแปลงเป็นเลขจำนวนเต็ม
ใช้เมธอด deleteNode ของ avlTree เพื่อลบโหนดนั้นออกจาก Tree และอัปเดตการแสดงผล Treeใน TextAreaOnDisplay หากการแปลงตัวเลขล้มเหลว (เช่น ผู้ใช้ใส่ข้อมูลที่ไม่ใช่ตัวเลข) จะมีการแสดงกล่องข้อความเตือนให้ใส่เลขที่ถูกต้อง

1.4.27 ค้นหาโหนดใน Tree (find_BtnActionPerformed)

ฟังก์ชันนี้ทำงานเมื่อผู้ใช้กดปุ่ม Find เพื่อค้นหาโหนดใน Tree AVL
รับค่าจากช่อง findBox แปลงเป็นตัวเลข และใช้เมธอด find เพื่อค้นหาโหนดใน Tree
หากพบโหนดนั้น จะมีการแสดงข้อความว่าโหนดนั้นอยู่ใน Tree หากไม่พบจะมีข้อความบอกว่าไม่พบ
หากเกิดข้อผิดพลาดในการแปลงค่า จะมีการแจ้งเตือนเช่นเดียวกับฟังก์ชันลบ

1.4.28 พิมพ์ลำดับการท่อง Tree (print_BtnActionPerformed)

ฟังก์ชันนี้ทำงานเมื่อกดปุ่ม Print เพื่อพิมพ์ลำดับการท่อง Tree (Traversal) ใน 3 รูปแบบ คือ Pre-order, In-order, และ Post-order
ใช้เมธอด preOrder, inOrder, และ postOrder ของ avlTree เพื่อท่องไปตามโหนดในลำดับต่าง ๆ
ผลลัพธ์จะแสดงออกทาง Terminal โดยใช้ System.out.println และมีการแจ้งข้อความว่าการท่อง Treeเสร็จสมบูรณ์

1.4.29 การตรวจจับการนำเมาส์ไปวางเหนือปุ่ม (insertBoxMouseEntered)

ฟังก์ชันนี้ทำงานเมื่อผู้ใช้นำเมาส์ไปวางเหนือช่อง insertBox ในฟังก์ชันนี้ยังไม่มีการทำงานจริง แต่สามารถใส่โค้ดเพื่อแสดงการทำงานที่ต้องการได้ เช่น เปลี่ยนสีของช่อง หรือแสดงข้อมูลเพิ่มเติม

```
// Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JPanel Display;
    private javax.swing.JLabel HeaderBG;
    private javax.swing.JTextArea TextAreaOnDisplay;
    private javax.swing.JPanel controlBtn;
    private javax.swing.JTextField deleteBox;
    private javax.swing.JButton delete_Btn;
    private javax.swing.JTextField findBox;
    private javax.swing.JButton find_Btn;
    private javax.swing.JPanel header;
    private javax.swing.JTextField insertBox;
    private javax.swing.JButton insert_Btn;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton print_Btn;
    // End of variables declaration//GEN-END:variables
}

```

รูปที่ 1.4.30 การสร้างฟังก์ชั่น GUI รูปแบบที่ 8

1.4.30 ประกาศตัวแปร GUI

การประกาศตัวแปรที่เกี่ยวข้องกับอินเทอร์เฟซผู้ใช้ (UI) โดยใช้ javax.swing สำหรับการสร้างหน้าต่างแอปพลิเคชันแบบกราฟิก (GUI) มีการประกาศตัวแปรที่ใช้ในการควบคุมองค์ประกอบต่าง ๆ เช่น ปุ่ม (JButton), ช่องข้อความ (JTextField), และแผงควบคุม (JPanel) ดังนี้:

Display: แผง (JPanel) ที่อาจใช้แสดงผลบางอย่าง

HeaderBG: ป้ายข้อความ (JLabel) ที่อาจใช้เป็นหัวเรื่อง

TextAreaOnDisplay: พื้นที่ข้อความ (JTextArea) สำหรับแสดงข้อมูล

controlBtn: แผงควบคุม (JPanel) สำหรับปุ่มต่าง ๆ

deleteBox, findBox, insertBox: ช่องข้อความ (JTextField) สำหรับการลบ, ค้นหา และแทรกข้อมูล

delete_Btn, find_Btn, insert_Btn: ปุ่ม (JButton) สำหรับการลบ, ค้นหา และแทรกข้อมูล

header: แผง (JPanel) ที่อาจใช้เป็นหัวเรื่อง

jPanel1, jScrollPane1: แผงและแถบเลื่อน

print_Btn: ปุ่มพิมพ์

1.4.31 การเปิดใช้งานตัวโปรแกรม

อ้างอิงจากหัวข้อ 1.2 ในตัวไฟล์จะมี Method Main ใช้ในการรันตัวโปรแกรมทั้งหมดใน package AVL Tree

main Method:

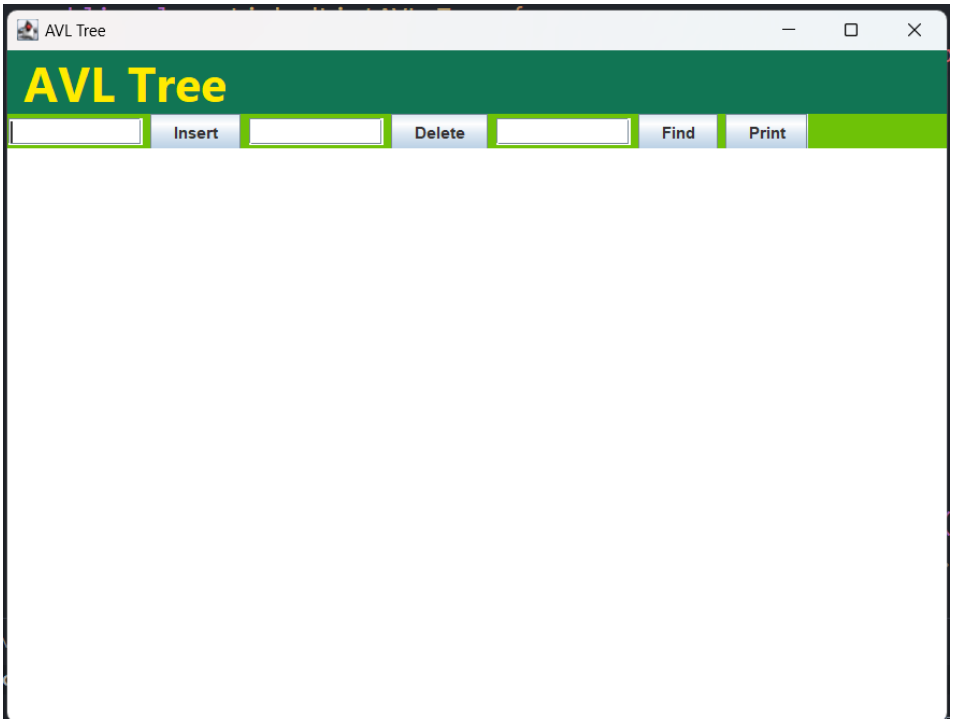
ใน main method โค้ดนี้ใช้ Java Swing เพื่อแสดง GUI โดยสร้างหน้าต่างที่มีชื่อว่า "AVL Tree" ซึ่งหน้าต่างนี้จะถูกสร้างและแสดงผลเมื่อโปรแกรมทำงาน

ผลลัพธ์ตัวโปรแกรม AVL Tree

2.1 ตัวโปรแกรม

AVL Tree คือโครงสร้างข้อมูลชนิดหนึ่งที่ใช้ในการจัดเก็บข้อมูลแบบเรียงลำดับ (sorted data) โดยมีคุณสมบัติพิเศษที่ทำให้การค้นหาข้อมูลทำได้อย่างรวดเร็วและมีประสิทธิภาพสูงเสมอ ไม่ว่าจะมีการเพิ่มหรือลบข้อมูลเข้าไปในโครงสร้างนี้ก็ตาม

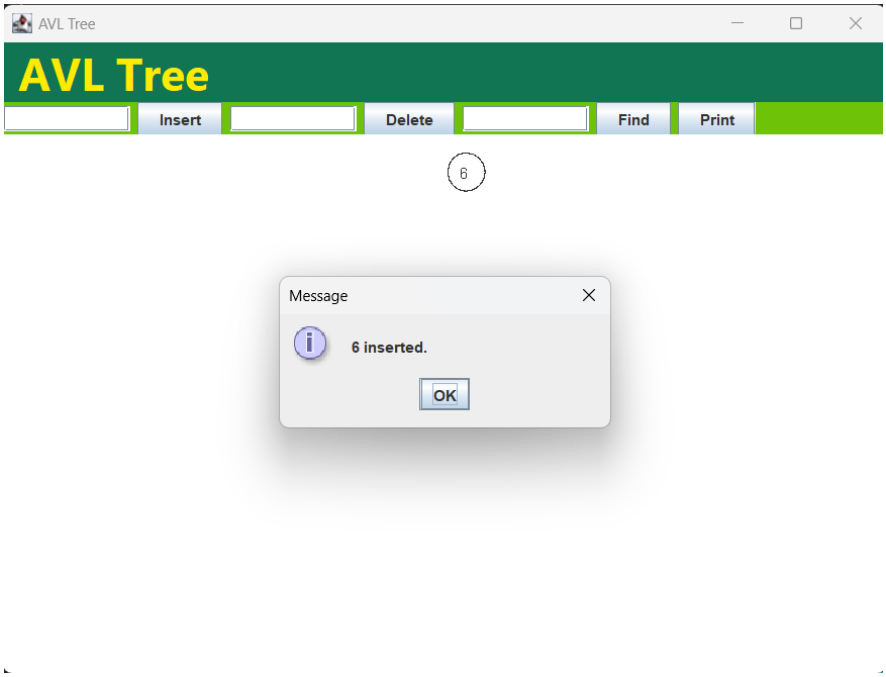
โปรแกรม AVL Tree ที่แสดงภาพมานั้น น่าจะเป็นโปรแกรมที่ถูกพัฒนาขึ้นมาเพื่อให้ผู้ใช้สามารถทดลองใช้งานและเรียนรู้เกี่ยวกับโครงสร้างข้อมูล AVL Tree ได้อย่างง่ายดาย โดยมีฟังก์ชันการทำงานหลักๆ ดังนี้



รูปที่ 2.1 ตัวโปรแกรม AVL Tree

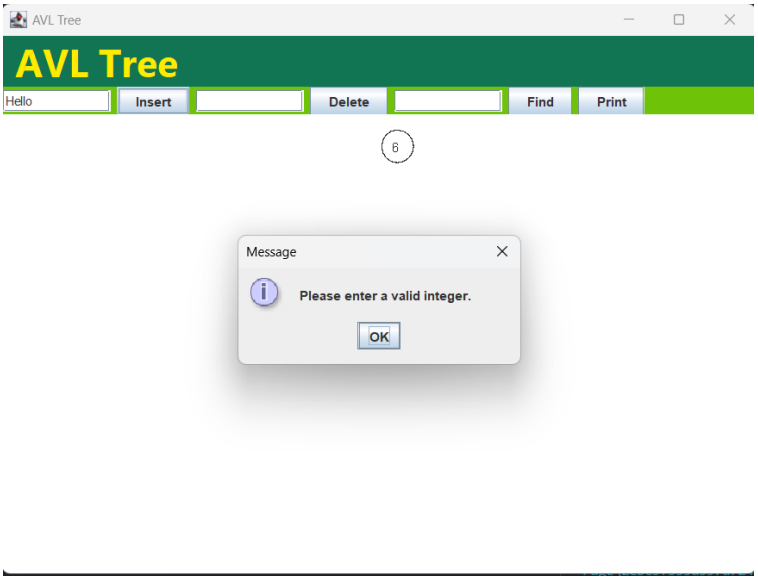
2.1.1 Insert

ใช้สำหรับเพิ่มข้อมูลเข้าไปใน AVL Tree โดยโปรแกรมจะทำการจัดเรียงข้อมูลให้อยู่ในตำแหน่งที่ถูกต้อง รักษาสมดุลของ Treeให้คงอยู่เสมอ และต้องใส่ข้อมูลที่เป็นตัวเลขจำนวนเต็มบวกเท่านั้น



รูปที่ 2.1.1 การ Insert Node

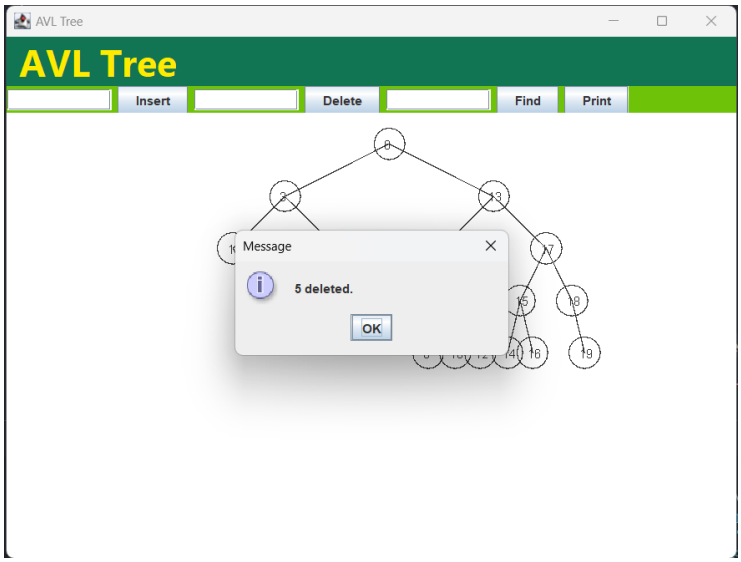
และไม่สามารถ Insert ข้อมูลที่นอกเหนือจากตัวเลขจำนวนเต็มบวกเท่านั้น



รูปที่ 2.1.1 ดักจับการรับค่าข้อมูล

2.1.2 Delete

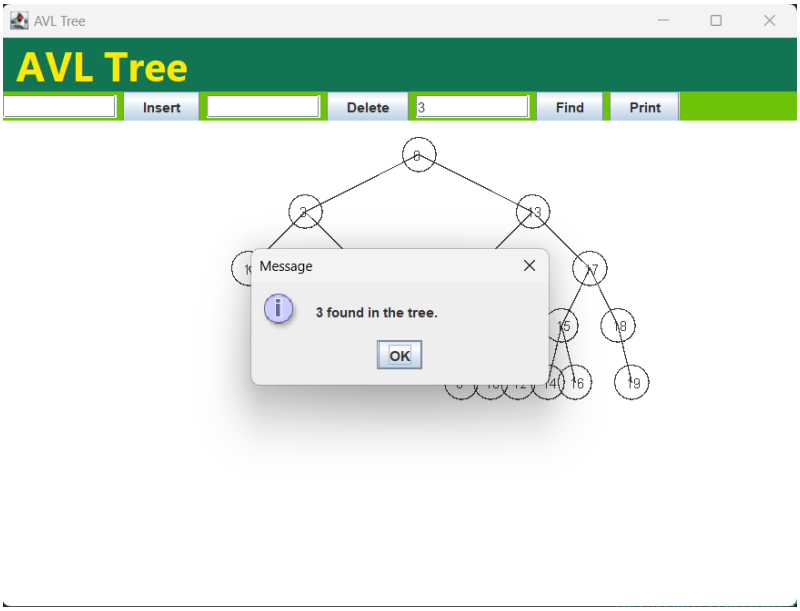
ใช้สำหรับลบข้อมูลออกจาก AVL Tree โดยโปรแกรมจะทำการปรับโครงสร้างของ Treeให้กลับมาสมดุลอีกครั้ง



รูปที่ 2.1.2 การลบข้อมูล

2.1.3 Find

ใช้สำหรับค้นหาข้อมูลที่ต้องการภายใน AVL Tree



รูปที่ 2.1.3 การหาข้อมูล

2.1.4 Print

ใช้สำหรับแสดงข้อมูลทั้งหมดที่มีอยู่ใน AVL Tree ออกมาในรูปแบบ Pre-order Inorder Post-order ในตัว Terminal

รูปที่ 2.1.4 การแสดงข้อมูลตามลำดับ

2.2 การทำงานเบื้องหลัง

2.2.1 Binary Search Tree (BST)

AVL Tree เป็นการต่อยอดมาจาก BST ซึ่งเป็นโครงสร้างข้อมูลที่เรียงลำดับข้อมูล โดยข้อมูลที่น้อยกว่ารากจะอยู่ทางซ้าย และข้อมูลที่มากกว่ารากจะอยู่ทางขวา

2.2.2 Balance: AVL Tree

มีคุณสมบัติพิเศษคือการรักษาความสมดุลของ Tree โดยความสูงของ subtree ซ้ายและขวาของทุกโหนดจะแตกต่างกันไม่เกิน 1 หน่วย ซึ่งทำให้การค้นหาข้อมูลมีประสิทธิภาพสูงเสมอ

2.2.3 Rotation

เมื่อมีการเพิ่มหรือลบข้อมูล ทำให้ AVL Tree สูญเสียความสมดุล โปรแกรมจะทำการปรับโครงสร้าง Tree โดยใช้เทคนิคการหมุน (rotation) เพื่อคืนความสมดุลให้กับ Tree

2.2.4 Order

Pre-order: เป็นการเยี่ยมชมโหนดในลำดับ ราก (Root) ซ้าย (Left) ขวา (Right)

In-order: เป็นการเยี่ยมชมโหนดในลำดับ ซ้าย (Left) ราก (Root) ขวา (Right) ซึ่งจะได้ลำดับข้อมูลที่เรียงจากน้อยไปมาก

Post-order: เป็นการเยี่ยมชมโหนดในลำดับ ซ้าย (Left) ขวา (Right) ราก (Root)

บรรณานุกรม

- https://www.cs.usfca.edu/~galles/visualization/AVLtree.html
- https://www.geeksforgeeks.org/introduction-to-avl-tree/
- https://gemini.google.com/
- https://chatgpt.com/