Project – CSD 207 OOP in Java

Prakhar Bhasin-1810110167
Kirtik Singh-1810110109

For this project assignment, you will solve a problem based on what you have learnt in this course.

Instructions

- Write Name and SNU ID of both the group members in the header of this document.
- Assignment submitted after the due date will not be evaluated and a score of zero will be awarded.
- Upload a word version of this document.
- Properly document/comment your code, followed by snapshots of output as desired.

Due Date and Time: **10 pm, November 24, 2019.**

Submitting this Assignment

You will submit (upload) this assignment in Blackboard. Email/paper submissions will not be accepted. All students must upload their project individually.

- Name this document as Project_CSD207-2019_John_Bill.doc in case the first names of group members are John and Bill respectively.

Grading Criteria

This assignment has 20 points (with weightage of 10% in your overall 100 points). Points will be awarded as follows:

(a) Functionality – **14 points**
(b) Look and Feel of node creation, deletion and searching implementations – **06 points**

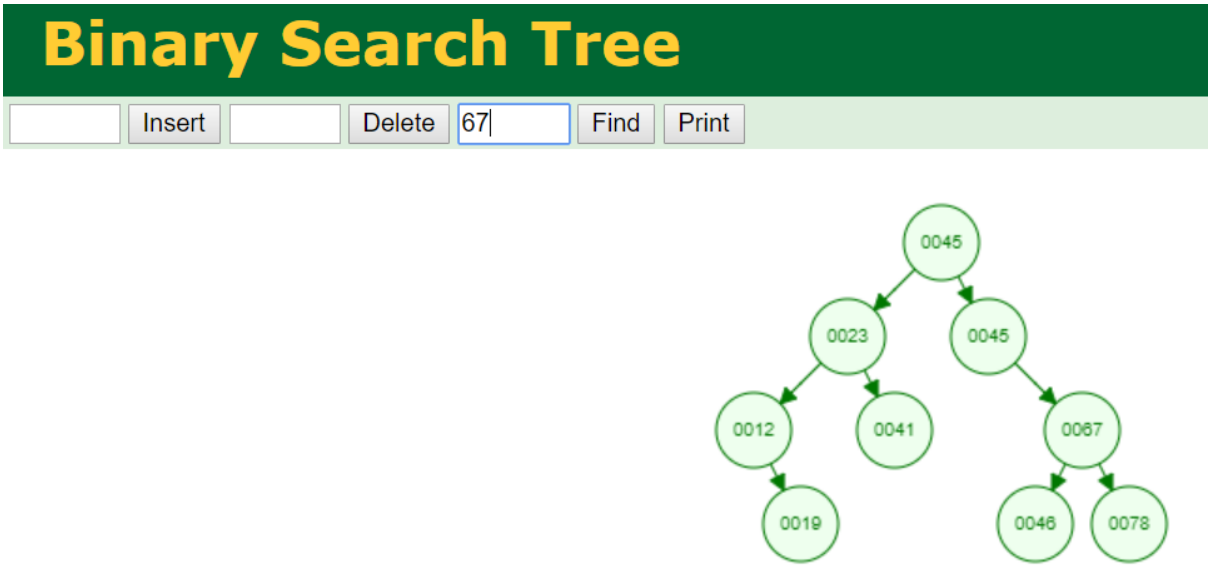
Project Problem

Write a java program to create a **binary search tree** (as shown in the figure) that will make use of several Swing components, event handling, graphics and Java Collections Framework to implement. GUI must contain buttons to perform following operations:

a. Insert - to insert a node (element) into the tree
b. Delete - to delete a node from the tree
c. Find- to search an element in the tree
d. Print – to print the sorted list of elements

Program should keep updating the following details at the bottom of the Frame:
a. height of the tree
b. number of vertices

Project – CSD 207 OOP in Java
Prakhar Bhasin-1810110167
Kirtik Singh-1810110109




```
package bst;

import java.util.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import static javax.swing.JFrame.EXIT_ON_CLOSE;

/**
 *
 * @author HP
 */
class node {

   int value;
   node left;
   node right;

   node(int v) {
      value = v;
   }

   node getLeft() {
      return left;
   }

   node getRight() {
      return right;
   }
```

```
    int getValue() {
        return value;
    }

    node setLeft(int v) {
        left = new node(v);
        return left;
    }

    node setRight(int v) {
        right = new node(v);
        return right;
    }
}

class BSTree {

    node root = null;
    String in = "";
    static int verticeCount = 0;

    node getRoot() {
        return root;
    }

    public void addNode(int v) {
        root = addToNode(root, v);
    }

    public node addToNode(node root, int v) {
        if (root == null) {
            root = new node(v);
            verticeCount++;
            return root;
        }
        if (v < root.value)
            root.left = addToNode(root.left, v);
        else if (v > root.value)
            root.right = addToNode(root.right, v);

        return root;
    }

    public node deleteNode(int v, node root) {
        if (root == null)
            return root;
        else if (v < root.getValue())
```

```
        root.left = deleteNode(v, root.left);
    else if (v > root.getValue())
        root.right = deleteNode(v, root.right);
    else {
        // Case- no child
        if (root.left == null && root.right == null) {
            root = null;
            verticeCount--;
        } // Case- one child
        //no left child
        else if (root.left == null) {
            node temp = root;
            root = root.right;
            temp = null;
            verticeCount--;
        } //no right child
        else if (root.right == null) {
            node temp = root;
            root = root.left;
            temp = null;
            verticeCount--;
        } // Case 3: 2 children
        else {
            node temp = inOrderSuccesor(root.right);
            root.value = temp.value;
            root.right = deleteNode(temp.getValue(), root.right);
        }
    }
    return root;
}

static node inOrderSuccesor(node root) {
    while (root.left != null)
        root = root.left;
    return root;
}

void inOrder() {
    inOrder(root);
}

void inOrder(node n) {
    if (n == null)
        return;
    inOrder(n.left);
    in = in + " " + n.value;
    inOrder(n.right);
```

```
    }

    int height(node n) {
        if (n == null)
            return 0;
        else {
            /* compute the depth of each subtree */
            int lHeight = height(n.left);
            int rHeight = height(n.right);

            /* use the larger one */
            if (lHeight > rHeight)
                return (lHeight + 1);
            else
                return (rHeight + 1);
        }
    }
    static boolean search(node root, int v) {
        while (root != null) {
            if (v > root.value)
                root = root.right;
            else if (v < root.value)
                root = root.left;
            else
                return true;
        }
        return false;
    }
}

public class BST {
    /**
     * @param args the command line arguments
     */
    int s = 35;
    JPanel paint;

    BST() {
        JFrame f = new JFrame();
        JPanel p = new JPanel();
        p.setBackground(new Color(236, 165, 76));
        paint = new JPanel();
        paint.setBackground(new Color(232, 235, 214));
        JPanel p1 = new JPanel();
        JPanel p2 = new JPanel();
        JTextField tf1 = new JTextField("", 3);
        JTextField tf2 = new JTextField("", 3);
```

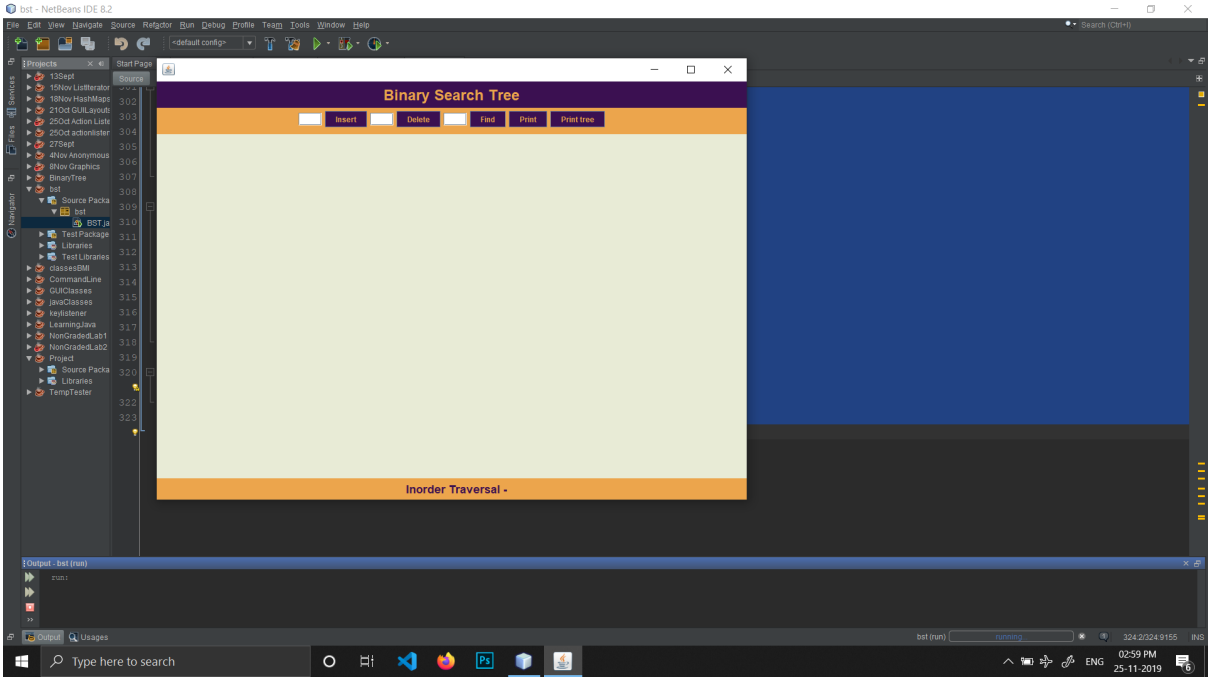
```
JTextField tf3 = new JTextField("", 3);
JButton b4 = new JButton("Find");
JButton b1 = new JButton("Insert");
JButton b2 = new JButton("Delete");
JButton b3 = new JButton("Print");
JButton printbutton = new JButton("Print tree");

JLabel l = new JLabel("Binary Search Tree");
JLabel l1 = new JLabel("");
JLabel l2 = new JLabel("");
JLabel l3 = new JLabel("");
Color c1 = new Color(59, 16, 81);

l.setForeground(new Color(236, 165, 76));
l.setBackground(c1);
Font myfont = new Font("Monospace", Font.BOLD, 24);
Font myfont1 = new Font("Sans Serif", Font.BOLD, 18);
l.setFont(myfont);
p1.setBackground(c1);
p1.add(l, BorderLayout.NORTH);

l1.setForeground(c1);
l2.setForeground(c1);
b1.setForeground(new Color(236, 165, 76));
b1.setBackground(c1);
b2.setForeground(new Color(236, 165, 76));
b2.setBackground(c1);
b3.setForeground(new Color(236, 165, 76));
b3.setBackground(c1);
b4.setForeground(new Color(236, 165, 76));
b4.setBackground(c1);
printbutton.setForeground(new Color(236, 165, 76));
printbutton.setBackground(c1);

JPanel panel = new JPanel();
panel.add(p1);
panel.add(p);
panel.setLayout(new GridLayout(2, 1));



p.add(tf1);
p.add(b1);
p.add(tf2);
p.add(b2);
p.add(tf3);
p.add(b4);
```

```
        p.add(b3);
        p.add(printbutton);
        p.setLayout(new FlowLayout(FlowLayout.CENTER));
        p2.setBackground(new Color(236, 165, 76));
        p2.add(l1);
        p2.add(l2);
        p2.add(l3);
        p2.setLayout(new FlowLayout(FlowLayout.CENTER));
//      f.add(panel);
        f.add(panel, BorderLayout.NORTH);
        f.add(paint, BorderLayout.CENTER);
        f.add(p2, BorderLayout.SOUTH);
        l1.setFont(myfont1);
        l2.setFont(myfont1);
        l3.setFont(myfont1);
        l3.setForeground(c1);


        f.setVisible(true);
        f.setSize(700, 700);
        //f.setLayout(null);
        f.setDefaultCloseOperation(EXIT_ON_CLOSE);
        BSTree obj = new BSTree();
        b1.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
              int x = Integer.parseInt(tf1.getText());
              obj.addNode(x);
              tf1.setText("");
              l1.setText("Height - " + obj.height(obj.root));
              l2.setText(" No. of Vertices - " + obj.verticeCount);

           }
        });
        b3.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
           obj.in="";
           obj.inOrder();
           l3.setText("Inorder Traversal - " +obj.in);
        }
        });
        b2.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
              int x = Integer.parseInt(tf2.getText());
              obj.deleteNode(x, obj.getRoot());
              tf2.setText("");
              l1.setText("Height - " + obj.height(obj.root));
              l2.setText(" No. of Vertices - " + obj.verticeCount);
```

```
            paint.repaint();
        }
    });

    printbutton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
            drawConnection(obj.getRoot(), 350 + 10, 10, 500 + 10, 10, 175);
            drawTheTree(obj.getRoot(), 350, 0, 175);
        }
    });
    b4.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            int x = Integer.parseInt(tf3.getText());
            boolean b = obj.search(obj.getRoot(), x);
            if(b){
                JOptionPane.showMessageDialog(null, x+ " is found");
            }
            else{
                JOptionPane.showMessageDialog(null, x+ " is not found");
        }
        }});

}

void drawTheNode(int x, int y, int data) {
    Graphics g = paint.getGraphics();
    Color c1 = new Color(59, 16, 81);
    g.setColor(c1);
    g.fillRect(x, y, s, s);
    g.setColor(new Color(236, 165, 76));
    g.drawString(Integer.toString(data), x + s / 4 + 4, y + s / 2 + 4);
}

void drawTheTree(node root, int x, int y, int X) {
    if (root == null)
        return;
    drawTheNode(x, y, root.getValue());
    drawTheTree(root.getLeft(), x - X / 2, y + 75, X / 2);
    drawTheTree(root.getRight(), x + X / 2, y + 75, X / 2);
}

void drawConnection(node root, int x, int y, int dx, int dy, int X) {
    if (root == null)
        return;
    Graphics g = paint.getGraphics();
    Color c1 = new Color(59, 16, 81);
```

```
        g.setColor(c1);
        g.drawLine(x, y, dx, dy);
        drawConnection(root.getLeft(), x - X / 2, y + 75, x, y, X / 2);
        drawConnection(root.getRight(), x + X / 2, y + 75, x, y, X / 2);
    }

    public static void main(String[] args) {
        new BST();
    }

}
```

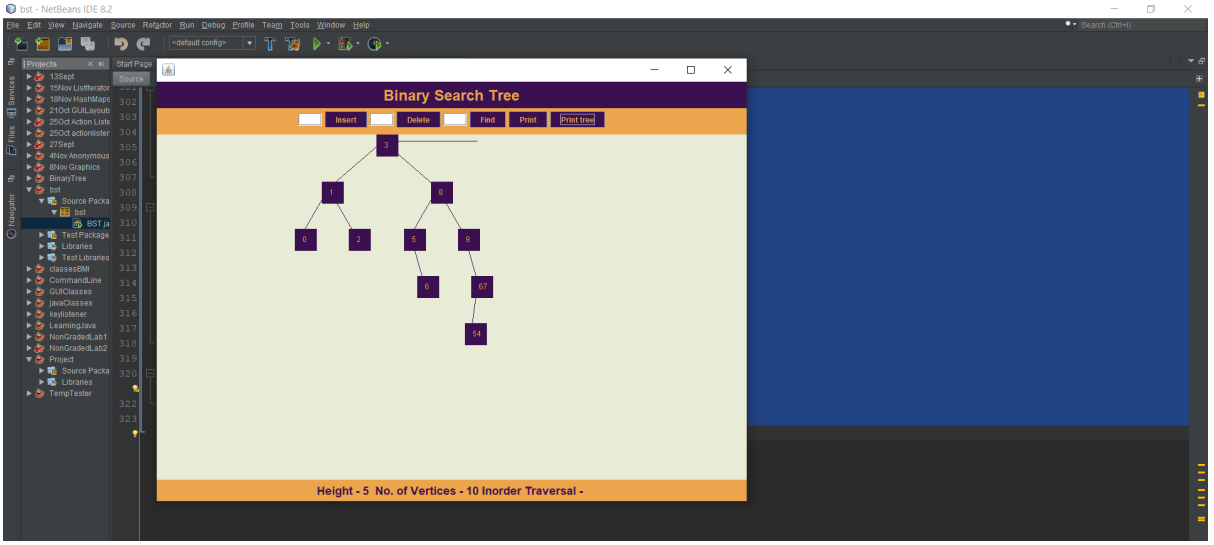
RUNNING CODE



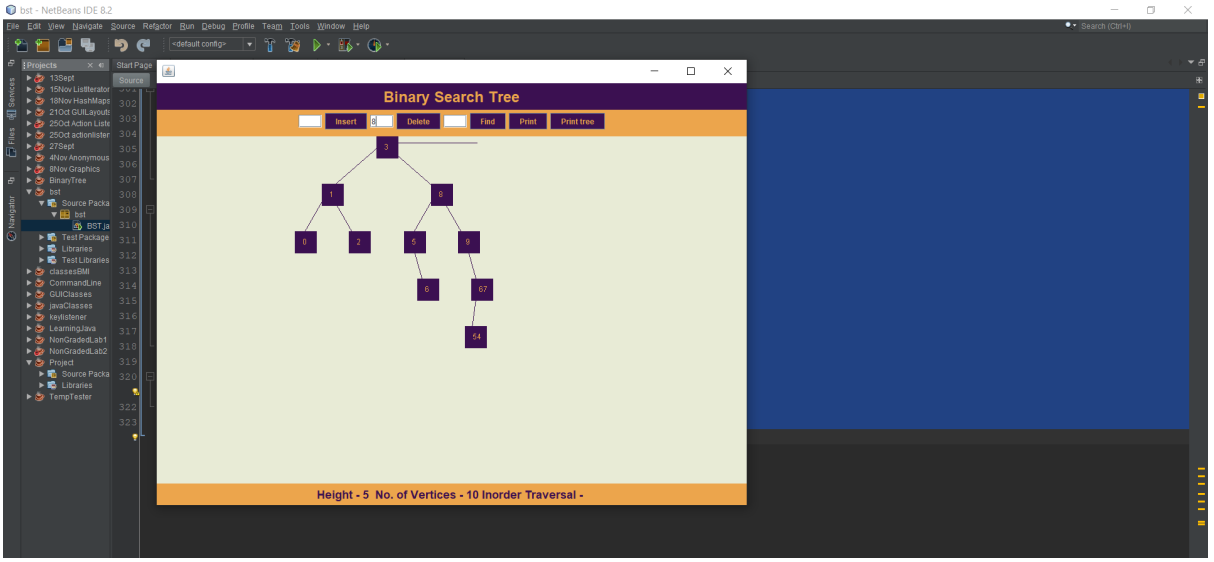
INSERTION IN BST

Project – CSD 207 OOP in Java

Prakhar Bhasin-1810110167
Kirtik Singh-1810110109

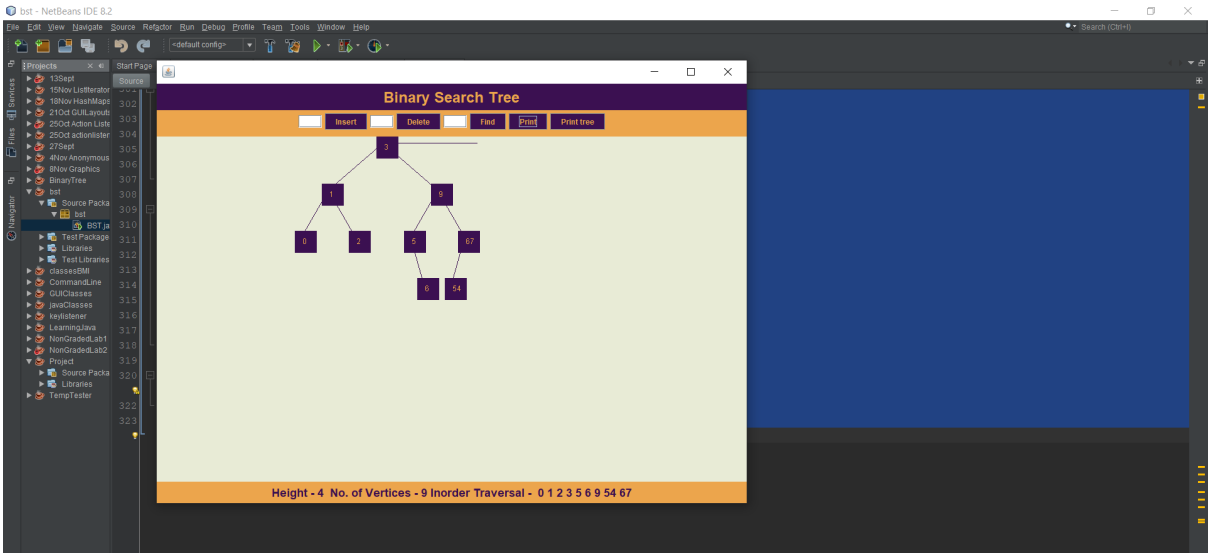


DELETION IN BST



Project – CSD 207 OOP in Java

Prakhar Bhasin-1810110167
Kirtik Singh-1810110109



Project – CSD 207 OOP in Java

Prakhar Bhasin-1810110167
Kirtik Singh-1810110109

SEARCH IN BST