Ayush Jagga

1910110106

package GradedLab3;  
  
public class Value {  
 int N;  
 int height = 0;  
 Value parent = null, left = null, right = null;  
  
 public Value(int n, int h, Value P) {  
 N = n;  
 height = h;  
 parent = P;  
 }  
}

package GradedLab3;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.geom.Ellipse2D;  
import java.awt.geom.Line2D;  
  
public class Drawtree extends JComponent{  
  
 final Color bg = Color.*white*;  
 final Color fg = Color.*black*;  
 final Color red = Color.*red*;  
 final Color white = Color.*white*;  
 final BasicStroke stroke = new BasicStroke(2.0f);  
 final BasicStroke wideStroke = new BasicStroke(8.0f);  
  
 Color dGreen = new Color(20, 125, 10);  
  
 Dimension totalSize;  
 int height, width;  
 Value r = null;  
 Value n1 = null;  
  
 public void init(Value N, int x,Color cl) {  
 setBackground(bg);  
 setForeground(cl);  
 r = N;  
 width = x;  
 }  
  
 Graphics2D g2;  
  
 public void paint(Graphics g) {  
 g2 = (Graphics2D) g;  
 g2.setRenderingHint(RenderingHints.*KEY\_ANTIALIASING*, RenderingHints.*VALUE\_ANTIALIAS\_ON*);  
 inorder(r, 0, width, 80);  
 }  
  
 public void draw(int x1, int x2, int y, String n, int d) {  
  
 g2.setStroke(stroke);  
  
 g2.setPaint(Color.*black*);  
 int x = (x1 + x2) / 2;  
 if (d == 1)  
 g2.draw(new Line2D.Double(x2, y - 30, x + 15, y));  
 else if (d == 2)  
 g2.draw(new Line2D.Double(x + 15, y, x1 + 30, y - 30));  
 if(n1!=null)  
 {  
 if(n1.N == Integer.*parseInt*(n))  
 { g2.setPaint(Color.*orange*); }  
  
 else  
 g2.setPaint(dGreen);  
 }  
 else  
 g2.setPaint(dGreen);  
  
  
 Shape circle = new Ellipse2D.Double((x1 + x2) / 2, y, 30, 30);  
 g2.draw(circle);  
 g2.fill(circle);  
  
 g2.setPaint(Color.*white*);  
  
 g2.drawString(n, x + 10, y + 18);  
 }  
  
 int x1 = 500, y1 = 30;  
  
 void inorder(Value r, int x1, int x2, int y) {  
 if (r == null)  
 return;  
  
  
 inorder(r.left, x1, (x1 + x2) / 2, y + 40);  
 if (r.parent == null)  
 draw(x1, x2, y, r.N + "", 0);  
 else {  
 if (r.parent.N < r.N)  
 draw(x1, x2, y, r.N + "", 2);  
 else  
 draw(x1, x2, y, r.N + "", 1);  
 }  
 inorder(r.right, (x1 + x2) / 2, x2, y + 40);  
 }  
  
 void find(Value r)  
 {  
 n1 = r;  
 }  
}

package GradedLab3;  
  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.Label;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.\*;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.stage.Stage;  
import javafx.scene.layout.HBox;  
  
import javax.swing.\*;  
  
import java.awt.\*;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
import java.util.ArrayList;  
public class finalmain extends Application {  
 static AVLTree *AVL* = new AVLTree();  
 int x = -1;  
 int y = -1;  
 int z = -1;  
 int search=0;  
 Value n;  
 ArrayList<Integer> inOrder = new ArrayList<>();  
 ArrayList<Integer> preOrder = new ArrayList<>();  
 ArrayList<Integer> postOrder = new ArrayList<>();  
 Label inOrderLabel = new Label();  
 Label preOrderLabel = new Label();  
 Label postOrderLabel = new Label();  
 Pane workingpane = new Pane();  
 BorderPane Finalpane=new BorderPane();  
  
 public void start(Stage primaryStage) throws Exception {  
 Button top = new Button("AVL Trees");  
 top.setPrefSize(400, 100);  
 top.setStyle("-fx-background-color: #006400; -fx-font-size: 30px; -fx-text-fill: #FFD700; -fx-alignment: center;");  
 top.setMaxWidth(99999D);  
  
 HBox center = new HBox();  
 center.setStyle("-fx-background-color: #C0D9AF;");  
 center.setMaxWidth(99999D);  
 center.setPadding(new Insets(5, 5, 5, 5));  
 //center.setSpacing(15);  
  
 HBox hh=new HBox(50);  
 Label la3=new Label("Height of AVL search tree is ");  
 la3.setText(la3.getText()+Integer.*toString*(*AVL*.maxDepth(*AVL*.root)));  
 la3.setStyle(" -fx-text-fill: #FFD700;");  
 Label la4=new Label("No of nodes in tree are ");  
 la4.setText(la4.getText()+Integer.*toString*(*AVL*.countnodes(*AVL*.root)));  
 la4.setStyle(" -fx-text-fill: #FFD700;");  
 hh.getChildren().addAll(la3,la4);  
 hh.setPrefSize(400, 30);  
 hh.setStyle("-fx-background-color: #006400; -fx-font-size: 15px; -fx-alignment: center;");  
 hh.setMaxWidth(99999D);  
 Finalpane.setBottom(hh);  
 *AVL*.count=0;  
  
 VBox centreleft=new VBox(50);  
 Label order=new Label("Order");  
 order.setStyle(" -fx-text-fill:#006400 ;-fx-font-size: 30px;");  
 inOrderLabel.setText("\n\n\nInorder:"+inOrder.toString()+" \n");  
 preOrderLabel.setText("Preorder:"+preOrder.toString()+ "\n");  
 postOrderLabel.setText("Postorder:"+postOrder.toString()+ "\n");  
 centreleft.getChildren().addAll(order,inOrderLabel,preOrderLabel,postOrderLabel);  
 centreleft.setPrefSize(200, 200);  
 centreleft.setStyle("-fx-background-color:#FFD700 ; -fx-font-size: 15px; -fx-alignment: center;");  
 centreleft.setMaxWidth(99999D);  
 centreleft.setMaxHeight(9999D);  
 Finalpane.setLeft(centreleft);  
  
 Label list=new Label("Execution List");  
 list.setStyle(" -fx-text-fill:#006400 ;-fx-font-size: 30px;");  
 VBox centerone=new VBox();  
 centerone.setAlignment(Pos.*TOP\_CENTER*);  
 centerone.setPrefSize(800,500);  
 FlowPane centerbox=new FlowPane();  
 centerbox.setHgap(30);  
 centerbox.setVgap(30);  
 centerbox.setPadding(new Insets(30,30,30,30));  
 centerone.getChildren().addAll(list,centerbox);  
 workingpane.getChildren().addAll(centerone);  
  
  
  
  
  
 TextField insert = new TextField();  
 Button l1 = new Button("Insert");  
 l1.setOnAction(e ->  
 {  
 x = Integer.*parseInt*(insert.getText());  
 *AVL*.insert(x, *AVL*.root);  
 insert.clear();  
 printer();  
 addOrderToPane();  
  
 HBox h=new HBox(50);  
 Label la1=new Label("Height of AVL search tree is ");  
 la1.setText(la1.getText()+Integer.*toString*(*AVL*.maxDepth(*AVL*.root)));  
 la1.setStyle(" -fx-text-fill: #FFD700;");  
 Label la2=new Label("No of nodes in tree are ");  
 la2.setText(la2.getText()+Integer.*toString*(*AVL*.countnodes(*AVL*.root)));  
 la2.setStyle(" -fx-text-fill: #FFD700;");  
 h.getChildren().addAll(la1,la2);  
  
 h.setPrefSize(400, 30);  
 h.setStyle("-fx-background-color: #006400; -fx-font-size: 15px; -fx-alignment: center;");  
 h.setMaxWidth(99999D);  
 Finalpane.setBottom(h);  
 *AVL*.count=0;  
  
 Label exe= new Label(Integer.*toString*(x)+" has been successfully added to the tree->");  
 exe.setFont(Font.*font*("Times New Roman", FontWeight.*BOLD*, 12));  
 centerbox.getChildren().addAll(exe);  
  
 });  
  
 TextField delete = new TextField();  
 Button l2 = new Button("Delete");  
  
 l2.setOnAction(e ->  
 {  
 y = Integer.*parseInt*(delete.getText());  
 Value temp = *AVL*.search(y, *AVL*.root);  
 if (temp == null)  
 JOptionPane.*showMessageDialog*(null, "Not Found");  
 else {  
 *AVL*.remove(temp);  
 JOptionPane.*showMessageDialog*(null, "Removed");  
 }  
 delete.clear();  
 printer();  
 addOrderToPane();  
 HBox h=new HBox(50);  
 Label la1=new Label("Height of AVL search tree is ");  
 la1.setText(la1.getText()+Integer.*toString*(*AVL*.maxDepth(*AVL*.root)));  
 la1.setStyle(" -fx-text-fill: #FFD700;");  
 Label la2=new Label("No of nodes in tree are ");  
 la2.setText(la2.getText()+Integer.*toString*(*AVL*.countnodes(*AVL*.root)));  
 la2.setStyle(" -fx-text-fill: #FFD700;");  
 h.getChildren().addAll(la1,la2);  
  
 h.setPrefSize(400, 30);  
 h.setStyle("-fx-background-color: #006400; -fx-font-size: 15px; -fx-alignment: center;");  
 h.setMaxWidth(99999D);  
 Finalpane.setBottom(h);  
 *AVL*.count=0;  
  
  
 Label exe= new Label(Integer.*toString*(y)+" has been successfully dleted from the tree->");  
 exe.setFont(Font.*font*("Times New Roman", FontWeight.*BOLD*, 12));  
 centerbox.getChildren().addAll(exe);  
  
 });  
  
  
 TextField find = new TextField();  
 Button l3 = new Button("Find");  
 l3.setOnAction(e ->  
 {  
 z = Integer.*parseInt*(find.getText());  
 Value temp = *AVL*.search(z, *AVL*.root);  
 if (temp == null)  
 JOptionPane.*showMessageDialog*(null, "Not Found");  
 else {  
 JOptionPane.*showMessageDialog*(null, "Found ");  
 search=1;  
 n = temp;  
 }  
  
 printer();  
 find.clear();  
  
 Label exe= new Label(" User searched for the Element "+Integer.*toString*(z) +" in tree->");  
 exe.setFont(Font.*font*("Times New Roman", FontWeight.*BOLD*, 12));  
 centerbox.getChildren().addAll(exe);  
  
  
 });  
  
 Button l4 = new Button("Print");  
 l4.setOnAction(e ->  
 {  
 printer();  
 Label exe= new Label(" User printed the tree");  
 exe.setFont(Font.*font*("Times New Roman", FontWeight.*BOLD*, 12));  
 centerbox.getChildren().addAll(exe);  
  
  
 });  
  
 Button l5 = new Button("Clear");  
 l5.setOnAction(e ->  
 {  
 *AVL*.root=null;  
 printer();  
 HBox h=new HBox();  
 Label la1=new Label("Height of AVL search tree is ");  
 la1.setText(la1.getText()+Integer.*toString*(*AVL*.maxDepth(*AVL*.root)));  
 Label la2=new Label("No of nodes in tree are ");  
 la2.setText(la2.getText()+Integer.*toString*(*AVL*.countnodes(*AVL*.root)));  
 h.getChildren().addAll(la1,la2);  
 Finalpane.setBottom(h);  
 *AVL*.count=0;  
  
 Label exe= new Label("User has cleared the tree");  
 exe.setFont(Font.*font*("Times New Roman", FontWeight.*BOLD*, 12));  
 centerbox.getChildren().addAll(exe);  
 addOrderToPane();  
  
 });  
  
 center.getChildren().addAll(insert, l1, delete, l2, find, l3, l4, l5);  
 center.setSpacing(15);  
 center.setAlignment(Pos.*CENTER\_LEFT*);  
  
  
 VBox vbox = new VBox();  
 vbox.getChildren().addAll(top, center);  
  
  
 Finalpane.setTop(vbox);  
 Finalpane.setCenter(workingpane);  
  
  
 Scene sc = new Scene(Finalpane, 1000, 650);  
 sc.setOnMouseClicked(e -> {  
 System.*out*.println(e.getX() + "\n" + e.getY());  
 });  
 primaryStage.setTitle("jAVFX");  
 primaryStage.setScene(sc);  
 primaryStage.show();  
  
 }  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
  
 public void printer() {  
  
 JFrame f = new JFrame("AVL Tree panel");  
 f.addWindowListener(new WindowAdapter() {  
 public void windowClosing(WindowEvent e) {  
 }  
 });  
  
 JButton button = new JButton("Next!");  
 f.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 f.setContentPane(button);  
 button.addActionListener(ex -> {  
 f.dispose();  
 });  
  
 Drawtree applet = new Drawtree();  
 f.getContentPane().add("Center", applet);  
  
  
 Toolkit tk = Toolkit.*getDefaultToolkit*();  
 int xSize = ((int) tk.getScreenSize().getWidth() / 2);  
  
 if(search==1)  
 {  
 applet.find(n);  
  
 }  
  
 applet.init(*AVL*.root,xSize-50,applet.fg);  
  
 f.pack();  
 f.setSize(new Dimension(xSize, 500));  
 f.setVisible(true);  
 n=null;  
  
 }  
  
 void printPostorder(Value node){  
 if (node == null)  
 return;  
  
 printPostorder(node.left);  
  
 printPostorder(node.right);  
  
 postOrder.add(node.N);  
 }  
  
 void printInorder(Value node) {  
 if (node == null)  
 return;  
  
 printInorder(node.left);  
 inOrder.add(node.N);  
 printInorder(node.right);  
 }  
  
 void printPreorder(Value node) {  
 if (node == null)  
 return;  
  
 preOrder.add(node.N);  
  
 printPreorder(node.left);  
  
 printPreorder(node.right);  
 }  
  
 public void addOrderToPane(){  
 VBox orderVBox= new VBox(50);  
 printPostorder(*AVL*.root);  
 printInorder(*AVL*.root);  
 printPreorder(*AVL*.root);  
  
 Label lab=new Label("Order");  
 lab.setStyle(" -fx-text-fill:#006400 ;-fx-font-size: 30px;");  
 inOrderLabel.setText("\n\n\nInorder:"+inOrder.toString()+" \n");  
 preOrderLabel.setText("Preorder:"+preOrder.toString()+ "\n");  
 postOrderLabel.setText("Postorder:"+postOrder.toString()+ "\n");  
  
 orderVBox.getChildren().addAll(lab,inOrderLabel, preOrderLabel,postOrderLabel);  
 orderVBox.setPrefSize(200, 200);  
 orderVBox.setStyle("-fx-background-color:#FFD700 ; -fx-font-size: 15px; -fx-alignment: center;");  
 orderVBox.setMaxWidth(99999D);  
 orderVBox.setMaxHeight(9999D);  
 Finalpane.setLeft(orderVBox);  
 preOrder.clear();  
 inOrder.clear();  
 postOrder.clear();  
  
  
  
 }  
}

package GradedLab3;  
  
  
public class AVLTree {  
 Value root = null;  
 int op;  
  
 void insert(int N, Value R) {  
 op=1;  
 if (root == null)  
 root = new Value(N, 1, null);  
 else {  
 if (N <= R.N) {  
 if (R.left == null) {  
 R.left = new Value(N, 1, R);  
 dobalance(R.left);  
 } else  
 insert(N, R.left);  
 } else {  
 if (R.right == null) {  
 R.right = new Value(N, 1, R);  
 dobalance(R.right);  
 } else  
 insert(N, R.right);  
 }  
 }  
  
 }  
  
 void remove(Value x) {  
 Value y, z = null;  
 op=0;  
 int ch=0;  
 if (x.left == null || x.right == null)  
 y = x;  
 else {  
 Value temp = x;  
 for (y = temp.parent; y != null && temp == y.right; y = y.parent) {  
 temp = y;  
 }  
 }  
 if((x.right==null || x.left==null) && x.parent==null )  
 { root=null;  
 x=null;  
 }  
  
  
 else if(y==x)  
 {  
 if (y.left != null)  
 z = y.left;  
 else  
 z = y.right;  
 if (z != null)  
 z.parent = y.parent;  
  
 if (y.parent == null) // z was is root  
 root = z;  
 else { // removing y  
 if (y == y.parent.left) // y was in left  
 { ch=1;  
 y.parent.left = z;}  
 else  
 {ch=0;  
 y.parent.right = z; // y was in right}  
 }  
 }  
  
 if(y.parent.parent==null && y.parent!=null)  
 {  
 if (ch==1)  
 {  
 if(y.parent.right!=null)  
 Lrotation(y.parent.right,y.parent);  
 else if(y.parent.left!=null)  
 { Lrotation(y.parent.left,y.parent);  
 //Rrotation();  
 }  
 }  
 else if (ch==0 && y.left!=null)  
 {  
  
 if(y.parent.left!=null){  
  
 Rrotation(y.parent.left,y.parent);  
 /\* System.out.println(y.parent.N+" "+y.parent.left.N+" "+y.parent.parent.N);  
 if(root.height>1)  
 {Lrotation(y.parent,y.parent.parent);  
 System.out.println(root.N);}\*/  
 }  
 else if(y.parent.right!=null)  
 {  
 Rrotation(y.parent.right,y.parent);  
  
 }  
 }  
  
 }  
 x.N = y.N;  
 y.height = 0;  
 }  
 else if(x!=root)  
 {  
 System.*out*.println("2");  
 y = x;  
 z = y.right;  
 y.left.parent = z;  
 z.left = y.left;  
 if (y.parent == null)  
 root = z;  
 else {  
 if (y == y.parent.left)  
 y.parent.left = z;  
 else  
 y.parent.right = z;  
 }  
 z.parent = y.parent;  
 x.N = y.N;  
 y.height = 0;  
  
 }  
 else  
 {  
 System.*out*.println("3");  
 y = x;  
 z = y.right;  
 root = z;  
 Value temp;  
  
  
 for(temp=z; temp.left!=null; temp =temp.left)  
 {  
  
 }  
  
 y.left.parent = temp;  
 temp.left = y.left;  
 root.parent = null;  
  
 x.N = y.N;  
 y.height = 0;  
  
 temp=root;  
 if(temp.right!=null)  
 {  
 for(temp=root; temp.right.right!=null; temp = temp.right);  
 {  
  
 }}  
 z = temp;  
 System.*out*.println(z.N);  
 // dobalance(z);  
  
 }  
  
 y = z;  
 if(y!=null){  
 do {  
 op=0;  
 System.*out*.println(z.N);  
 if(y!=null)  
 y = dobalance(y);  
  
 } while (y != null);  
 }  
  
  
 }  
  
 Value dobalance(Value x) {  
 int h1 = 0, h2 = 0;  
 Value n1 = null, rent = null;  
 Value y = x, z;  
  
 System.*out*.println(root.height+ " "+op);  
  
 if(x==root && op==0)  
 {  
 System.*out*.println(root.height);  
 System.*out*.println(" rotation ");  
 if(root.left==null)  
 {  
 // l rotation  
 System.*out*.println("L rotation ");  
 Lrotation(y.right,y);  
 }  
 else if(root.right == null)  
 {  
 //r rotation  
 System.*out*.println("R rotation ");  
 if(y.right!=null)  
 Rrotation(y.right,y);  
 }  
 return null;  
 }  
  
 while (y.parent != null) {  
 if (y.parent.left == y)  
 n1 = y.parent.right;  
 else if (y.parent.right == y)  
 n1 = y.parent.left;  
 h1 = y.height;  
 if (n1 == null)  
 h2 = 0;  
 else  
 h2 = n1.height;  
 if (Math.*abs*(h2 - h1) > 1)  
 break;  
  
 y.parent.height = 1 + Math.*max*(h1, h2);  
 y = y.parent;  
 }  
  
 if(y.parent==null)  
 return null;  
  
  
 z = y.parent;  
 rent = z;  
 h1 = (z.left == null) ? 0 : z.left.height;  
 h2 = (z.right == null) ? 0 : z.right.height;  
 if (h1 < h2)  
 y = z.right;  
 else  
 y = z.left;  
  
 h1 = (y.left == null) ? 0 : y.left.height;  
 h2 = (y.right == null) ? 0 : y.right.height;  
 if (h1 < h2)  
 x = y.right;  
 else  
 x = y.left;  
  
 y.parent = z;  
 if(x!=null)  
 x.parent = y;  
  
 if (z.left == y) {  
 if (y.left == x)  
 Rrotation(y, z);  
 else {  
 Lrotation(x, y);  
 x.height++;  
 Rrotation(x, z);  
 }  
 } else {  
 if (y.right == x) {  
 Lrotation(y, z);  
 } else {  
 Rrotation(x, y);  
 x.height++;  
 Lrotation(x, z);  
 }  
 }  
  
 return rent;  
 }  
 void Rrotation(Value y, Value z) {  
 y.parent = z.parent;  
 if (y.parent == null)  
 root = y;  
 else if (y.parent.left == z)  
 y.parent.left = y;  
 else  
 y.parent.right = y;  
 z.left = y.right;  
 if (z.left != null)  
 z.left.parent = z;  
 y.right = z;  
 z.parent = y;  
 z.height--;  
 }  
 void Lrotation(Value y, Value z) {  
 y.parent = z.parent;  
 if (z.parent == null)  
 root = y;  
 else if (z.parent.left == z)  
 y.parent.left = y;  
 else  
 y.parent.right = y;  
 z.right = y.left;  
 if (z.right != null)  
 z.right.parent = z;  
 y.left = z;  
 z.parent = y;  
 z.height--;  
 }  
  
 Value search(int N, Value r) {  
 if (r == null)  
 return null;  
 if (N == r.N)  
 return r;  
 else if (N < r.N)  
 return search(N, r.left);  
 else  
 return search(N, r.right);  
 }  
  
 public int maxDepth(Value node)  
 {  
 if (node == null)  
 return 0;  
 else  
 {  
 int lDepth = maxDepth(node.left);  
 int rDepth = maxDepth(node.right);  
 if (lDepth > rDepth)  
 return (lDepth + 1);  
 else  
 return (rDepth + 1);  
 }  
 }  
 public int count = 0;  
 int countnodes(Value root)  
 {  
 if(root != null)  
 {  
 countnodes(root.left);  
 count++;  
 countnodes(root.right);  
 }  
 return count;  
 }  
  
}

OUTPUT

