**GRADED LAB 3**

**JAVAFX PROJECT**

Max Heap

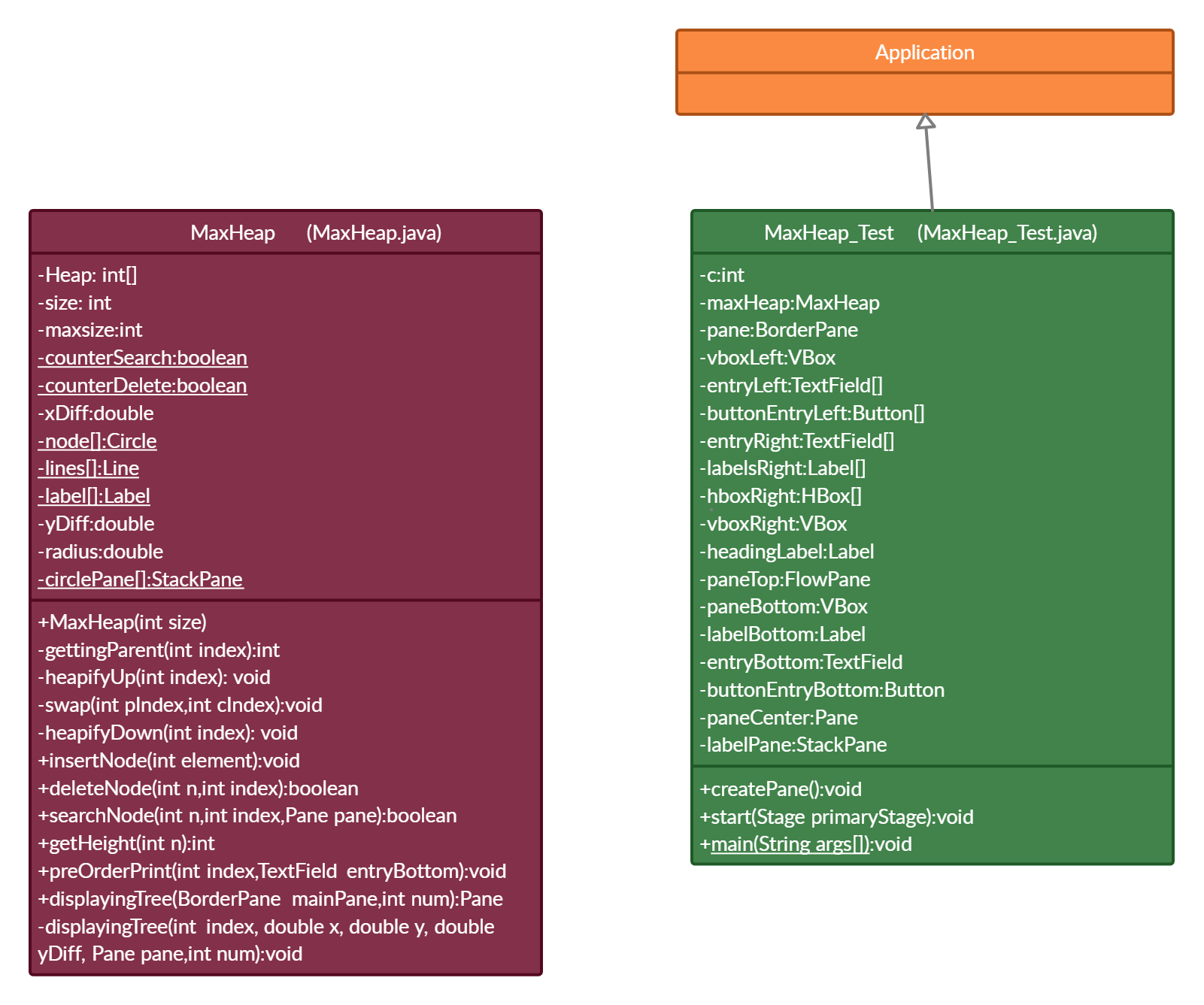
**GROUP MEMBERS:**

1.Aditya Srivastava:1910110034

2.Jayati Sharma:1910110181

We have made a project incorporating insertion, deletion, finding an element and displaying elements in pre order in a max heap using a javafx interface.

UML diagram for the program:



\*\*There are two files, one named MaxHeap.java and another Max\_Heap.java.

**File: MaxHeap.java**

**Source code:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package GL3;

import javafx.geometry.Pos;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.Pane;

import javafx.scene.layout.StackPane;

import javafx.scene.paint.Color;

import javafx.scene.shape.Circle;

import javafx.scene.shape.Line;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.scene.text.Text;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author user

\*/

public class MaxHeap {

private int[] Heap; //Heap array of int type

private int size; //numbering strating from 1

private int maxsize;//maximum size of the array

private static boolean counterSearch = false,counterDelete = false; //counter for searchNode and deleteNode function

private double xDiff = 30;// difference between gettingParent and child nodes horizontally

private double yDiff = 300;//difference between gettingParent and child nodes vertically

private double radius = 25;//radius of the circle for node

private static Circle node[]; //contains all the circle nodes of tree

private static Line lines[]; //contains all the lines of the tree

private static Label label[];

private static StackPane circlePane[]; //contains the circle and label of the tree

public MaxHeap(int size) { //initialization of all variables necessary for the maxHeap

this.maxsize = size;

this.size = 0;

Heap = new int[this.maxsize + 1];

for(int i=0;i<size;i++){

Heap[i]=-1;

}

Heap[0] = Integer.MAX\_VALUE;

node = new Circle[size];

lines = new Line[size];

label = new Label[size];

circlePane = new StackPane[size];

}

private int gettingParent(int index) { //return gettingParent of the current node

return index / 2;

}

private void swap(int pIndex, int cIndex)

{

int temp = Heap[pIndex];

Heap[pIndex] = Heap[cIndex];

Heap[cIndex] = temp;

}

private void heapifyDown(int index) //to be used for deletion

{

if (index > size)

return;

if (Heap[index] < Heap[(2 \* index)] ||

Heap[index] < Heap[(2 \* index) + 1]) {

if (Heap[(2 \* index)] > Heap[(2 \* index) + 1]) {

swap(index, (2 \* index));

heapifyDown(2 \* index);

}

else {

swap(index, (2 \* index) + 1);

heapifyDown((2 \* index) + 1);

}

}

}

private void heapifyUp(int index) { //heap sort

int temp = Heap[index];

while(index>0 && temp > Heap[gettingParent(index)]){//checks if child is greater than the gettingParent

Heap[index] = Heap[gettingParent(index)];//if so swaps the element

index = gettingParent(index);

}

Heap[index] = temp;

}

public void insertNode(int element) throws IllegalArgumentException,ArithmeticException{

for(int i=0;i<=size;i++)//checks that only unique element is inserted

{

if(Heap[i]==element){

throw new IllegalArgumentException();//if not unique then throws error to notify the user

}

}

Heap[++size] = element;//if unique insert the element in array

int current = size;

heapifyUp(current);//heap sort to balance the heap

}

public boolean deleteNode(int n,int index){

if (index > size) { //checks if index is greater than the current size then return

return false;

}

if(Heap[index]==n){ //if the node is found

int lastElement = Heap[size]; //gets last element

Heap[index]=lastElement; //replace it with the current node

Heap[size]=-1; //delete the last node of the tree

size--;

heapifyDown(index); //heap sort to balance the heap

return true; //returns true indicating everything was done successfully

}

// if not able to find the element then check in the left and right child

if(counterDelete!=true){

counterDelete = deleteNode(n,(2 \* index));

}

if(counterDelete!=true){

counterDelete = deleteNode(n,(2 \* index)+1);

}

if(index==1){

if(counterDelete==true){//initializing counterDelete to be false at the end of stack of recursion

counterDelete=false;

return true;

}

}

return counterDelete;

}

public boolean searchNode(int n,int index,Pane pane){

if (index > size) { //checks if index is greater than size

return false;

}

if(index == 1){

for(int i=1;i<=size;i++){ //set border of all nodes as black

node[i].setStroke(Color.BLACK);

node[i].setFill(Color.WHITE);

}

}

if(Heap[index]==n){ //if element found

node[index].setStroke(Color.LAWNGREEN);//change its border to lawngreen

node[index].setFill(Color.LAWNGREEN);

label[index].setFont(Font.font("Verdana",FontWeight.BOLD,14));

return true;

}

else{

node[index].setStroke(Color.CRIMSON);

node[index].setFill(Color.CRIMSON);

label[index].setFont(Font.font("Verdana",FontWeight.BOLD,14));

}

// if not found searches left and right child of that node

if(counterSearch!=true){

counterSearch = searchNode(n,(2 \* index),pane);

}

if(counterSearch!=true){

counterSearch= searchNode(n,(2 \* index)+1,pane);

}

if(index==1){

if(counterSearch==true){//initializes the counter variable before exiting the function completely

counterSearch=false;

return true;

}

}

return counterSearch;

}

public int getHeight(int n){

int log = (int) (Math.log(n)/Math.log(2)); //if nodes are present returns height

return ((int)Math.floor(log));

}

public void preOrderPrint(int index,TextField entryBottom){

if(size==0){//to empty the textField if no nodes are present

entryBottom.clear();

}

if (index > size) { // if greater than the size

return;

}

if(index==1) {// empty the TextField before printing the preOrderPrint

entryBottom.setText("");

}

entryBottom.setText(entryBottom.getText()+" "+Heap[index]);//prints element

preOrderPrint((2 \* index),entryBottom);//check for left child of the node

preOrderPrint((2 \* index)+1,entryBottom);//check for right child of the node

}

public Pane displayingTree(BorderPane mainPane,int num) {

Pane pane = new Pane();//creates new pane for displaying the tree

if(Heap[1]!=-1){

displayingTree(1, mainPane.getWidth() / 2.8, mainPane.getHeight() / 4, yDiff ,pane,num);//if the heap is not empty then execute the code for creating the tree

}

return pane;

}

private void displayingTree(int index, double x, double y, double yDiff, Pane pane,int num) {

if(index>size)

return;

if (Heap[(2\*index)] != -1) { //checks if left node exists

lines[2\*index] = new Line(x - yDiff, y + xDiff, x, y);//create lines for the left node

pane.getChildren().add(lines[2\*index]); // display the line

displayingTree(2\*index, x - yDiff, y + xDiff, yDiff / 1.8,pane,num); // create the left subtree recursively

}

if (Heap[(2\*index)+1] != -1) {

lines[(2\*index)+1] = new Line(x + yDiff, y + xDiff, x, y); //create lines for the left node

pane.getChildren().add(lines[(2\*index)+1]); //display the line

displayingTree((2\*index)+1, x + yDiff, y + xDiff, yDiff / 1.8,pane,num); // create the right subtree recursively

}

node[index] = new Circle(x, y, radius); //create the node

node[index].setFill(Color.WHITE);

node[index].setStrokeWidth(3);

label[index] = new Label(Heap[index] + ""); //add text on the node

circlePane[index] = new StackPane(node[index],label[index]); // add circle and text in stackPane

if(num==Heap[index]){

node[index].setStroke(Color.LAWNGREEN); //change border for the current node being inserted

node[index].setFill(Color.LAWNGREEN);

label[index].setFont(Font.font("Verdana",FontWeight.BOLD,14));

}

else{

node[index].setStroke(Color.BLACK);

node[index].setFill(Color.WHITE);

}

pane.getChildren().addAll(circlePane[index]); //display the node along with text

circlePane[index].setTranslateX(node[index].getCenterX()-node[index].getRadius()); //set the x-coordinate of stackPane

circlePane[index].setTranslateY(node[index].getCenterY()-node[index].getRadius()); //set the y-coordinate of stackPane

}

}

**File: MaxHeap\_Test.java**

**Source code:**

package GL3;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.InputStream;

import java.util.Timer;

import java.util.TimerTask;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.application.Application;

import static javafx.application.Application.launch;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.layout.Background;

import javafx.scene.layout.BackgroundImage;

import javafx.scene.layout.BackgroundPosition;

import javafx.scene.layout.BackgroundRepeat;

import javafx.scene.layout.BackgroundSize;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.FlowPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.Text;

import javafx.stage.Popup;

import javafx.stage.Stage;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author user

\*/

public class MaxHeap\_Test extends Application{

private int c=0; //contains the nodes in the Heap

private MaxHeap maxHeap = new MaxHeap(200); //create maxHeap

private BorderPane pane = new BorderPane(); //create pane for the frontend

private VBox vboxLeft = new VBox(10); //creates left panel of the frontend

private TextField[] entryLeft = new TextField[3]; //array for the textField to be used on the left panel

private Button[] buttonEntryLeft = new Button[3]; //array for the buttons to be used on the left panel

private TextField[] entryRight = new TextField[2]; //array for the textField to be used on the right panel

private Label[] labelsRight = new Label[3]; //array for the labels to be used on the right panel

private HBox[] hboxRight = new HBox[2]; //array for the pane to be used on the right panel

private VBox vboxRight = new VBox(10); //creates right panel of the front end

private Label headingLabel = new Label("MAX HEAP"); //creates heading

private FlowPane paneTop = new FlowPane(); //creates top of the frontend

private VBox paneBottom = new VBox(10); //creates bottom of the frontend

private Label labelBottom = new Label("Pre-Order:");//creates label for the bottom in frontend

private TextField entryBottom = new TextField(); //creates textField to show the preOrderPrint in the frontend

private Button buttonEntryBottom = new Button("PRINT PRE-ORDER"); //creates button to print the preOrderPrint in the frontend

private Pane paneCenter = new Pane(); //create a new pane for the center of the frontend

private StackPane labelPane = new StackPane(); //create pane for labels

public void createPane() throws FileNotFoundException{ //setting up the frontend

//code for the tags used

InputStream stream1 = new FileInputStream("C:\\Users\\Hp\\Desktop\\SNU\\SNU SECOND YEAR\\SEM-3\\Principles of Programming Language\\JAVAFX PROJECT\\Project\_CSD203-2020\_Aditya\_Jayati\\2.png");

Image im1 = new Image(stream1);

ImageView imageView1 = new ImageView(im1);

InputStream stream2 = new FileInputStream("C:\\Users\\Hp\\Desktop\\SNU\\SNU SECOND YEAR\\SEM-3\\Principles of Programming Language\\JAVAFX PROJECT\\Project\_CSD203-2020\_Aditya\_Jayati\\1.png");

Image im2 = new Image(stream2);

ImageView imageView2 = new ImageView(im2);

//setting up the left panel in frontend

for(int i=0;i<3;i++){

entryLeft[i]=new TextField();

}

for(int i=0;i<3;i++){

buttonEntryLeft[i] = new Button();

}

vboxLeft.setPadding(new Insets(20,20,20,20));

entryLeft[0].setAlignment(Pos.CENTER\_RIGHT);

entryLeft[0].setPromptText("Insert:(Only Positive No.)");

entryLeft[1].setAlignment(Pos.CENTER\_RIGHT);

entryLeft[1].setPromptText("Delete:(Only Positive No.)");

entryLeft[2].setAlignment(Pos.CENTER\_RIGHT);

entryLeft[2].setPromptText("Find:(Only Positive No.)");

buttonEntryLeft[0].setText("INSERT");

buttonEntryLeft[0].setFont(new Font("Imprint MT Shadow",14));

buttonEntryLeft[1].setText("DELETE");

buttonEntryLeft[1].setFont(new Font("Imprint MT Shadow",14));

buttonEntryLeft[2].setText("FIND");

buttonEntryLeft[2].setFont(new Font("Imprint MT Shadow",14));

vboxLeft.getChildren().addAll(entryLeft[0],buttonEntryLeft[0],entryLeft[1],buttonEntryLeft[1],entryLeft[2],buttonEntryLeft[2]);

vboxLeft.setAlignment(Pos.CENTER);

vboxLeft.setStyle("-fx-background-color: #EDF285;");

//setting up the right panel in frontend

for(int i=0;i<3;i++){

labelsRight[i] = new Label();

}

for(int i=0;i<2;i++){

entryRight[i] = new TextField();

}

labelsRight[0].setText("CURRENT STATUS");

labelsRight[0].setFont(new Font("Imprint MT Shadow",28));

labelsRight[1].setText("Height:");

labelsRight[1].setFont(new Font("Imprint MT Shadow",20));

labelsRight[2].setText("Nodes: ");

labelsRight[2].setFont(new Font("Imprint MT Shadow",20));

entryRight[0].setAlignment(Pos.CENTER\_RIGHT);

entryRight[0].setEditable(false);

entryRight[0].setPromptText("Height of Tree");

entryRight[1].setAlignment(Pos.CENTER\_RIGHT);

entryRight[1].setEditable(false);

entryRight[1].setPromptText("Vertices of Tree");

hboxRight[0]=new HBox(10);

hboxRight[1]=new HBox(10);

hboxRight[0].getChildren().addAll(labelsRight[1],entryRight[0]);

hboxRight[1].getChildren().addAll(labelsRight[2],entryRight[1]);

vboxRight.setPadding(new Insets(0,20,20,20));

vboxRight.setAlignment(Pos.CENTER);

vboxRight.getChildren().addAll(labelsRight[0],hboxRight[0],hboxRight[1]);

vboxRight.setStyle("-fx-background-color: #EDF285;");

//setting up the headline in frontend

headingLabel.setFont(new Font("Segoe UI Black",60));

headingLabel.setAlignment(Pos.CENTER);

headingLabel.setPadding(new Insets(30,30,30,30));

paneTop.getChildren().add(headingLabel);

paneTop.setStyle("-fx-background-color: #206A5D;");

headingLabel.setTextFill(Color.WHITE);

paneTop.setAlignment(Pos.CENTER);

//setting up the bottom bar in frontend

labelBottom.setFont(new Font("Imprint MT Shadow",20));

entryBottom.setAlignment(Pos.CENTER\_RIGHT);

entryBottom.setPromptText("Pre-Order of Max Heap");

buttonEntryBottom.setFont(new Font("Imprint MT Shadow",20));

paneBottom.setPadding(new Insets(30,30,30,30));

entryBottom.setMaxWidth(500);

paneBottom.getChildren().addAll(buttonEntryBottom,entryBottom);

paneBottom.setAlignment(Pos.CENTER);

paneBottom.setStyle("-fx-background-color: #206A5D;");

//setting up the mainPane

pane.setLeft(vboxLeft);

pane.setRight(vboxRight);

pane.setTop(paneTop);

pane.setBottom(paneBottom);

//code to be executed at the click of the buttons

buttonEntryLeft[0].setOnAction(e->{ //code for insert

try{

int num = Integer.parseInt(entryLeft[0].getText()); //number to be added

if(num<0)

throw new ArithmeticException();

maxHeap.insertNode(num);

paneCenter.getChildren().clear();

paneCenter = maxHeap.displayingTree(pane,num);

pane.setCenter(paneCenter);

labelPane.getChildren().clear();

Text popupText = new Text(num+" added successfully!");

popupText.setFont(new Font("Imprint MT Shadow",20));

labelPane.getChildren().add(imageView1);

labelPane.setMaxHeight(50);

labelPane.setMaxWidth(250);

labelPane.getChildren().add(popupText);

paneCenter.getChildren().add(labelPane);

labelPane.setAlignment(Pos.CENTER);

popupText.setVisible(true);

labelPane.setVisible(true);

labelPane.setTranslateY(100);

Timer timer = new Timer();

timer.schedule(new TimerTask(){

public void run(){

popupText.setVisible(false);

labelPane.setVisible(false);

}

},3000);

entryLeft[0].setText("");

entryRight[1].setText(Integer.toString(++c));

entryRight[0].setText(Integer.toString(maxHeap.getHeight(c)));

}

catch(ArithmeticException qw){

entryLeft[0].setText("");

JOptionPane.showMessageDialog(null,"Enter Only Positive Numbers!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}

catch(NumberFormatException qw){

entryLeft[0].setText("");

JOptionPane.showMessageDialog(null,"Enter an Appropriate Element to be Inserted!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}

catch(IllegalArgumentException re){

entryLeft[0].setText("");

JOptionPane.showMessageDialog(null,"Element Already Exists!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ea){

entryLeft[0].setText("");

JOptionPane.showMessageDialog(null,"Error in inserting the element!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}

});

buttonEntryLeft[1].setOnAction(e->{ //code for delete

try{

String numText = entryLeft[1].getText();

int num = Integer.parseInt(numText);

if(num<0)

throw new ArithmeticException();

boolean a = maxHeap.deleteNode(num,1);

Text popupText;

labelPane.getChildren().clear();

paneCenter.getChildren().clear();

paneCenter = maxHeap.displayingTree(pane,-1);//-1 because no nodes needs to be highlighted

pane.setCenter(paneCenter);

labelPane.setVisible(true);

if(a==true){

popupText = new Text(num+" deleted successfully!");

popupText.setFont(new Font("Imprint MT Shadow",20));

labelPane.getChildren().add(imageView1);

popupText.setFill(Color.BLACK);

c--;

if(c!=0){

entryRight[0].setText(Integer.toString(maxHeap.getHeight(c)));

entryRight[1].setText(Integer.toString(c));

}

else{

entryRight[0].clear();

entryRight[1].clear();

}

}

else

{

popupText = new Text(num+" not in the heap!");

popupText.setFill(Color.WHITE);

popupText.setFont(new Font("Imprint MT Shadow",20));

labelPane.getChildren().add(imageView2);

}

labelPane.setAlignment(Pos.CENTER);

labelPane.setMaxHeight(50);

labelPane.setMaxWidth(250);

labelPane.getChildren().add(popupText);

paneCenter.getChildren().add(labelPane);

labelPane.setTranslateY(100);

Timer timer = new Timer();

timer.schedule(new TimerTask(){

public void run(){

popupText.setVisible(false);

labelPane.setVisible(false);

}

},3000);

//throwing appropriate exceptions

entryLeft[1].setText("");

}catch(ArithmeticException qw){

entryLeft[1].setText("");

JOptionPane.showMessageDialog(null,"Enter Only Positive Numbers!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}catch(NumberFormatException qw){

entryLeft[1].setText("");

JOptionPane.showMessageDialog(null,"Enter an Appropriate Element for Deletion!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ea){

entryLeft[1].setText("");

JOptionPane.showMessageDialog(null,"Error in deletion of the element!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}

});

buttonEntryLeft[2].setOnAction(e->{ //code for find

try{

boolean h;

Text popupText;

labelPane.getChildren().clear();

paneCenter.getChildren().clear();

paneCenter = maxHeap.displayingTree(pane,-1);//-1 because no nodes needs to be highlighted

pane.setCenter(paneCenter);

labelPane.setVisible(true);

int num = Integer.parseInt(entryLeft[2].getText());

if(num<0)

throw new ArithmeticException();

h = maxHeap.searchNode(num,1,paneCenter);

if(h==true){

popupText = new Text(num+" found!");

popupText.setFont(new Font("Imprint MT Shadow",20));

popupText.setFill(Color.BLACK);

labelPane.getChildren().add(imageView1);

}

else{

popupText = new Text(num+" not in the heap!");

popupText.setFont(new Font("Imprint MT Shadow",20));

popupText.setFill(Color.WHITE);

labelPane.getChildren().add(imageView2);

}

labelPane.setAlignment(Pos.CENTER);

labelPane.setMaxHeight(50);

labelPane.setMaxWidth(250);

labelPane.getChildren().add(popupText);

paneCenter.getChildren().add(labelPane);

labelPane.setTranslateY(100);

Timer timer = new Timer();

timer.schedule(new TimerTask(){

public void run(){

popupText.setVisible(false);

labelPane.setVisible(false);

}

},3000);

entryLeft[2].setText("");

}catch(ArithmeticException qw){

entryLeft[2].setText("");

JOptionPane.showMessageDialog(null,"Enter Only Positive Numbers!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}catch(NumberFormatException qw){

entryLeft[2].setText("");

JOptionPane.showMessageDialog(null,"Enter an Appropriate Element to Find!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}catch(Exception ea){

entryLeft[2].setText("");

JOptionPane.showMessageDialog(null,"Error in finding the element!","ALERT!",JOptionPane.ERROR\_MESSAGE);

}

});

buttonEntryBottom.setOnAction(e->{ //code for printing preOrderPrint of the maxHeap

maxHeap.preOrderPrint(1,entryBottom);

});

}

//setting up scene

public void start(Stage primaryStage) throws FileNotFoundException{

createPane();

Scene scene = new Scene(pane);

primaryStage.setScene(scene);

primaryStage.setMaximized(true);

primaryStage.show();

}

public static void main(String args[]){

launch(args);

}

}

For Output screenshots click [here](Project_CSD203-2020_Aditya_Jayati_outputs%20screenshots.pdf).

**\*\*Note: Kindly change the picture paths before running the code in line 70,67 in MaxHeap\_Test.**

**\*\*\*\*\*\*\*\***