**Question 1: (Chapter 19)**

**Write a generic method that returns the maximum element in a two-dimensional array**

**public static <E extends Comparable<E>> E max (E[][] list)**

**Write a generic method that returns the minimum element in a two-dimensional array**

**public static <E extends Comparable<E>> E min (E[][] list)**

*Code*

import java.util.Scanner;

public class TwoDArray {

public static void main(String[] args) {

Scanner input = new Scanner (System.in);

System.out.println("Select the following option");

System.out.println("Option 1 --> test the code with Pre-Configured Double 2D Array");

System.out.println("Option 2 --> test the code with your Own 2D Array integers ");

int opt = input.nextInt();

switch (opt) {

case 1 : PrefconfArray();

break;

case 2 : UsercreatArray();

break;

}

input.close();

}

private static void UsercreatArray() {

Scanner input = new Scanner (System.in);

System.out.println("Enter the number of rows and columns of the array");

int row = input.nextInt();

int col = input.nextInt();

Integer [] [] value = new Integer [row][col];

System.out.println("Enter the array in Integers");

for (int i = 0 ; i<value.length; i++) {

for (int j = 0 ;j < value[i].length; j++) {

value [i][j] = input.nextInt();

}

}

int v = max(value);

int s = min(value);

System.out.println("The Maxium Value is " + v);

System.out.println("The Minimum Value is " + s);

input.close();

}

private static void PrefconfArray() {

Double [] [] value = { {23.5, 35.5, 2.2, 10.9},

{4.5, 3.0, 45.5, 3.5},

{35.0,44.3, 5.5, 9.6}

};

System.out.println ("Pre-configured Double 2-D Arrays as given below:");

for (int i = 0 ; i<value.length; i++) {

for (int j = 0 ;j < value[i].length; j++) {

System.out.print(value [i][j] + " ");

}

System.out.print("\n");

}

double v = max(value);

double s = min(value);

System.out.println ("\n");

System.out.println("The Maxium Value is " + v);

System.out.println("The Minimum Value is " + s);

String [] [] str = { {"grapes" ,"oranges", "apples" ,"banana"},

{"ice-cream", "pudding","jelly", "pie"}

};

System.out.println ("\n");

System.out.println ("Pre-configured String 2-D Arrays as given below:");

for (int i = 0 ; i<str.length; i++) {

for (int j = 0 ;j < str[i].length; j++) {

System.out.print(str [i][j] + " ");

}

System.out.print("\n");

}

String sMax = max(str);

String sMin = min(str);

System.out.println ("\n");

System.out.println("Max word from the array is " + sMax);

System.out.println("Min word from the array is " + sMin);

}

public static <E extends Comparable<E>> E max (E[][] list) {

E maxV = list [0] [0];

for (int i = 0; i <list.length; i++) {

for (int j = 0; j<list[i].length; j++ ) {

if (list[i][j].compareTo(maxV) > 0) {

maxV = list [i] [j];

}

}

}

return maxV;

}

public static <E extends Comparable<E>> E min (E[][] list) {

E minV = list [0] [0];

for (int i = 0; i <list.length; i++) {

for (int j = 0; j<list[i].length; j++ ) {

if (list[i][j].compareTo(minV) < 0) {

minV = list [i] [j];

}

}

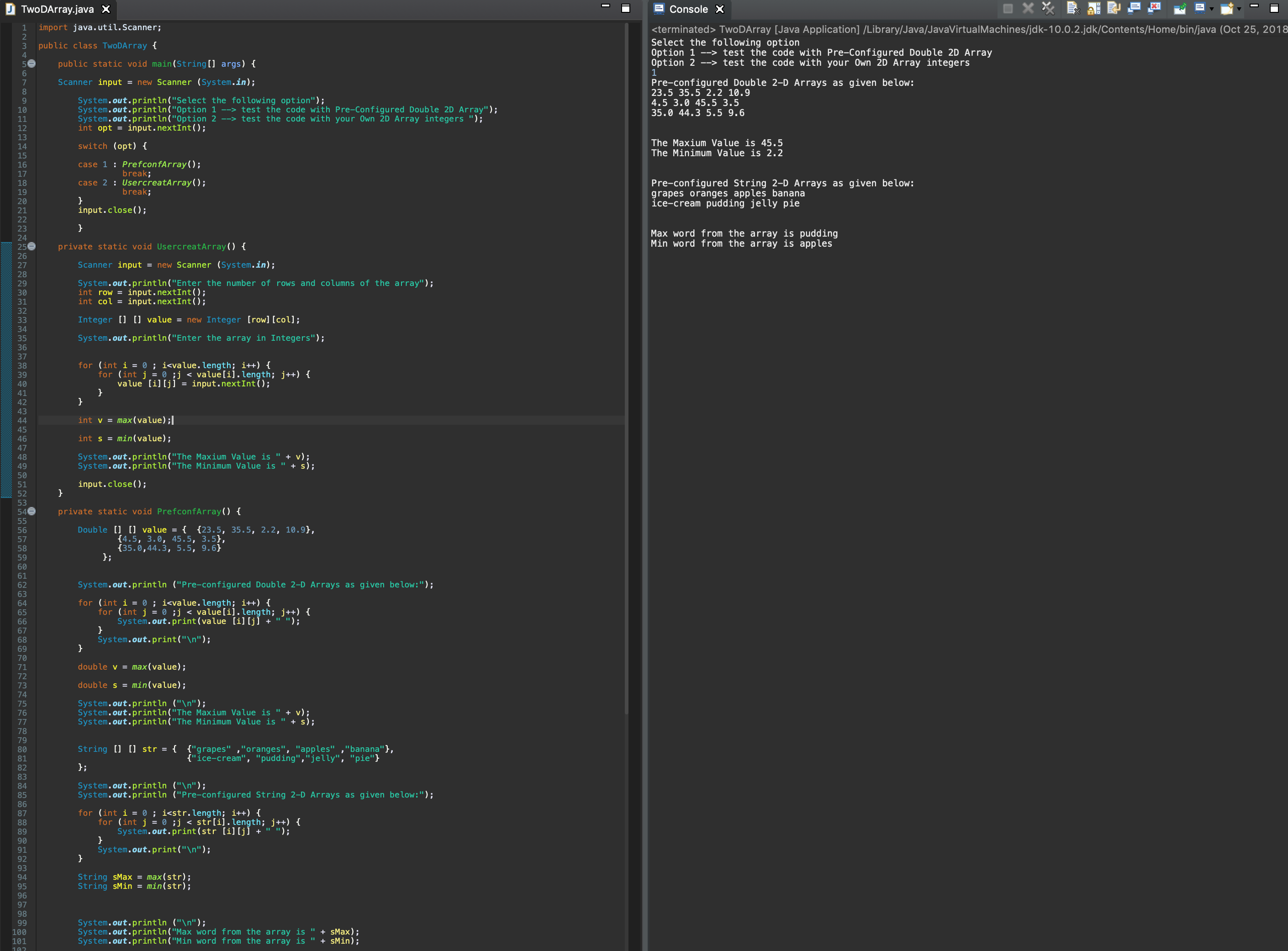
}

return minV;

}

}

*Screenshots*

**

**Question 2: (Chapter 20)**

**Write a program that lets the user enter last names from a graphical user interface and displays them in a text area. Use a linked list to store the last names. Do not store duplicate last names. Add the following buttons: Sort, Shuffle, and Reverse to sort, shuffle and reverse the list.**

*Code*

package application;

import javafx.scene.control.Button;

import javafx.scene.control.TextArea;

import java.util.Collections;

import java.util.LinkedList;

import java.util.ListIterator;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.stage.Stage;

public class SortLastNames extends Application {

private TextField tfLastName;

private TextArea taLastName;

private LinkedList<String> lkName ;

private Button btnSort;

private Button btnShuffle;

private Button btnReverse;

@Override

public void start(Stage primaryStage) throws Exception {

// TODO Auto-generated method stub

GridPane pane = new GridPane();

pane.setAlignment(Pos.CENTER);

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

pane.setHgap(10);

pane.setVgap(10);

//Linked List

lkName = new LinkedList<>();

//button Field

btnSort = new Button();

btnShuffle = new Button ();

btnReverse = new Button();

btnSort.setText("Sort");

btnShuffle.setText("Shuffle");

btnReverse.setText("Reverse");

//text field

tfLastName =new TextField();

//text area

taLastName = new TextArea();

taLastName.setDisable(true);

taLastName.setWrapText(true);

//sizes

tfLastName.setMinSize(100, 20);

taLastName.setMinSize(200, 300);

//place node

pane.add(tfLastName, 0, 0);

pane.add(taLastName, 1, 0);

pane.add(btnSort,0,1);

pane.add(btnShuffle, 0, 2);

pane.add(btnReverse, 0, 3);

//set on Action

tfLastName.setOnAction(e -> AddName());

btnSort.setOnAction(e -> SortHandle(e));

btnShuffle.setOnAction(e -> SortHandle(e));

btnReverse.setOnAction(e -> SortHandle(e));

Scene scene = new Scene (pane,500,500);

primaryStage.setTitle("Sort Last Name");

primaryStage.setScene(scene);

primaryStage.show();

}

private void SortHandle(ActionEvent e) {

if (e.getSource() == btnSort) {

Collections.sort(lkName);

}

else if (e.getSource() == btnShuffle) {

Collections.shuffle(lkName);

}

else if (e.getSource() == btnReverse) {

Collections.reverse(lkName);

}

DisplayList();

}

private void AddName() {

String name = tfLastName.getText();

if (lkName.contains(name)) {

taLastName.setText("Error:Duplicate Value found, please try again");

tfLastName.setText("");

}

else {

lkName.add(name);

DisplayList();

}

}

private void DisplayList() {

ListIterator<String> listIterator = lkName.listIterator();

String set = "";

while(listIterator.hasNext()) {

set = set + listIterator.next() + " ";

}

taLastName.setText(set);

tfLastName.setText("");

System.out.println(lkName);

}

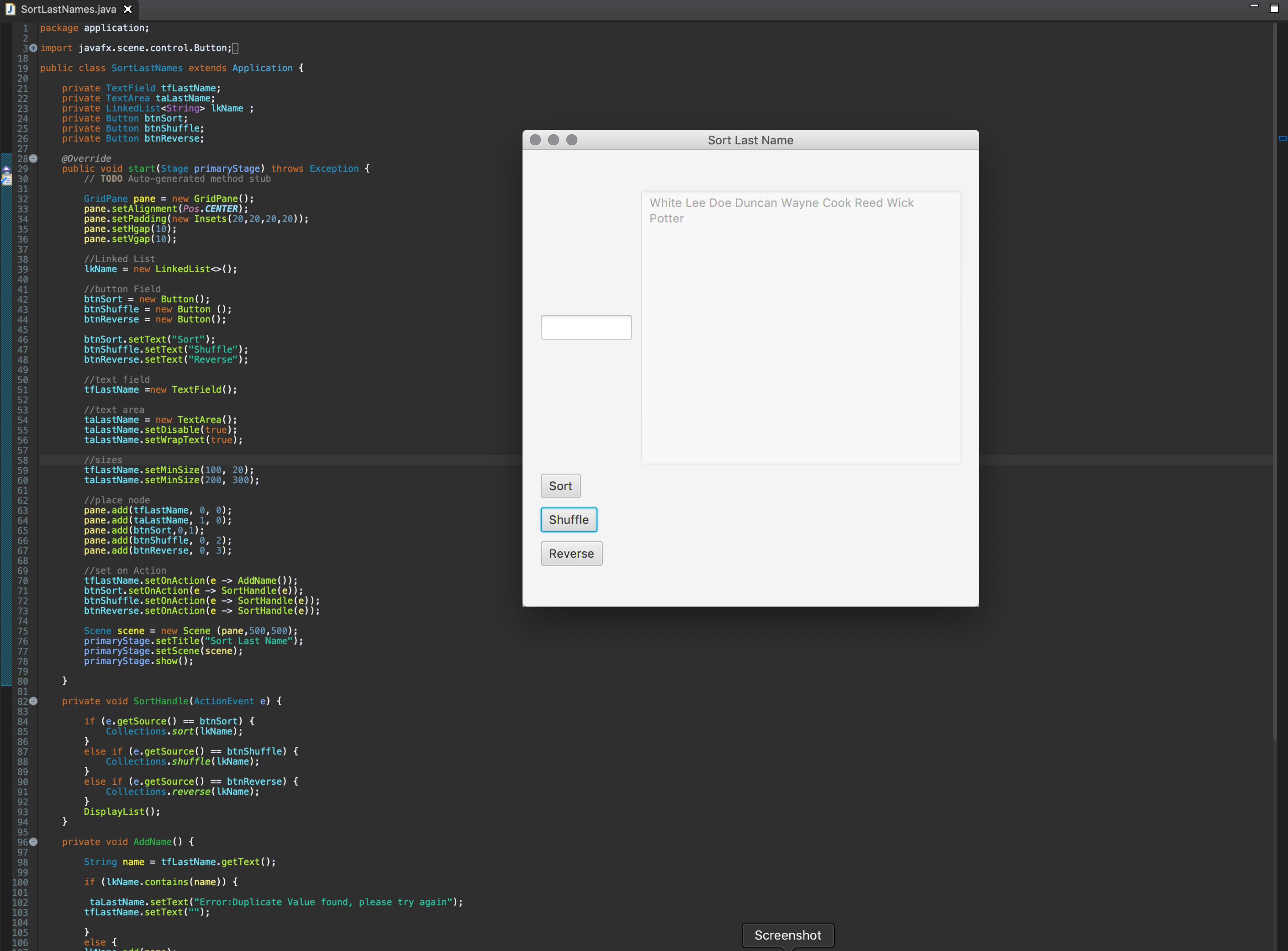
public static void main(String[] args) {

launch(args);

}

}

*Screenshots*

**