**Question 1( chapter 15)**

Write a GUI program to perform addition, subtraction, multiplication, division, finds maximum, fins minimum , and average of 2 double numbers.

Your stage should consist of:

label and input text for number 1

label and input text for number 2

label and output text for answer

buttons for addition, subtraction, multiplication, division, finds maximum, fins minimum , and average

*Code*

/\*

\* Question 1( chapter 15)

Write a GUI program to perform addition, subtraction, multiplication, division, finds maximum, fins minimum , and average of 2 double numbers.

Your stage should consist of:

label and input text for number 1

label and input text for number 2

label and output text for answer

buttons for addition, subtraction, multiplication, division, finds maximum, fins minimum , and average

\*

\*

\*/

package application;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

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.layout.GridPane;

import javafx.stage.Stage;

public class CalculationFx extends Application{

private TextField txtNum1;

private TextField txtNum2;

private Button btAdd;

private Button btSub;

private Button btDiv;

private Button btMul;

private Button btnMax;

private Button btnMin;

private Button btnAvg;

private TextField txtRes;

@Override

public void start(Stage primaryStage) throws Exception {

GridPane pane = new GridPane();

pane.setAlignment(Pos.CENTER);

pane.setPadding(new Insets(11, 11, 11, 11));

pane.setHgap(10);

pane.setVgap(10);

//Labels

Label lblNum1 = new Label ("Input Number 1");

Label lblNum2 = new Label ("Input Number 2");

Label lblRes = new Label ("Answer");

//Text

txtNum1 = new TextField();

txtNum2 = new TextField();

txtRes = new TextField();

//Button

btAdd = new Button("+");

btSub = new Button("-");

btDiv = new Button("/");

btMul = new Button("\*");

btnMax = new Button ("Max");

btnMin = new Button ("Min");

btnAvg = new Button ("Avg");

// Place nodes in the pane

pane.add(lblNum1, 0, 0 );

pane.add(txtNum1, 1, 0 );

pane.add(lblNum2, 0, 1);

pane.add(txtNum2, 1, 1);

pane.add(lblRes, 0, 2);

pane.add(txtRes, 1, 2);

pane.add(btAdd, 0, 3);

pane.add(btSub, 1, 3);

pane.add(btMul, 2, 3);

pane.add(btDiv, 0, 4);

pane.add(btnMax, 1, 4);

pane.add(btnMin, 2, 4);

pane.add(btnAvg, 1, 5);

//event handler codes

btAdd.setOnAction(new EventHandler <ActionEvent> () {

@Override

public void handle(ActionEvent e) {

// TODO Auto-generated method stub

BtnHandle(e);

}

});

btSub.setOnAction(new EventHandler <ActionEvent> () {

@Override

public void handle(ActionEvent e) {

// TODO Auto-generated method stub

BtnHandle(e);

}

});

btMul.setOnAction(new EventHandler <ActionEvent> () {

@Override

public void handle(ActionEvent e) {

// TODO Auto-generated method stub

BtnHandle(e);

}

});

//lambda Expression (sane above code)

btDiv.setOnAction (e -> BtnHandle (e));

btnMax.setOnAction(e -> BtnHandle (e));

btnMin.setOnAction(e -> BtnHandle (e));

btnAvg.setOnAction(e -> BtnHandle (e));

//set Scene

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

primaryStage.setTitle("Cal");

primaryStage.setScene(scene);

primaryStage.show();

}

private void BtnHandle(ActionEvent e) {

double num1 = Double.parseDouble(txtNum1.getText());

double num2 = Double.parseDouble(txtNum2.getText());

double res = 0.0;

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

res = num1+ num2;

}

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

res = num1 - num2;

}

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

res = num1 / num2;

}

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

res = num1 \* num2;

}

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

res = Math.max(num1, num2);

}

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

res = Math.min(num1, num2);

}

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

res = (num1 + num2) /2.0;

}

txtRes.setText(String.valueOf(res));

}

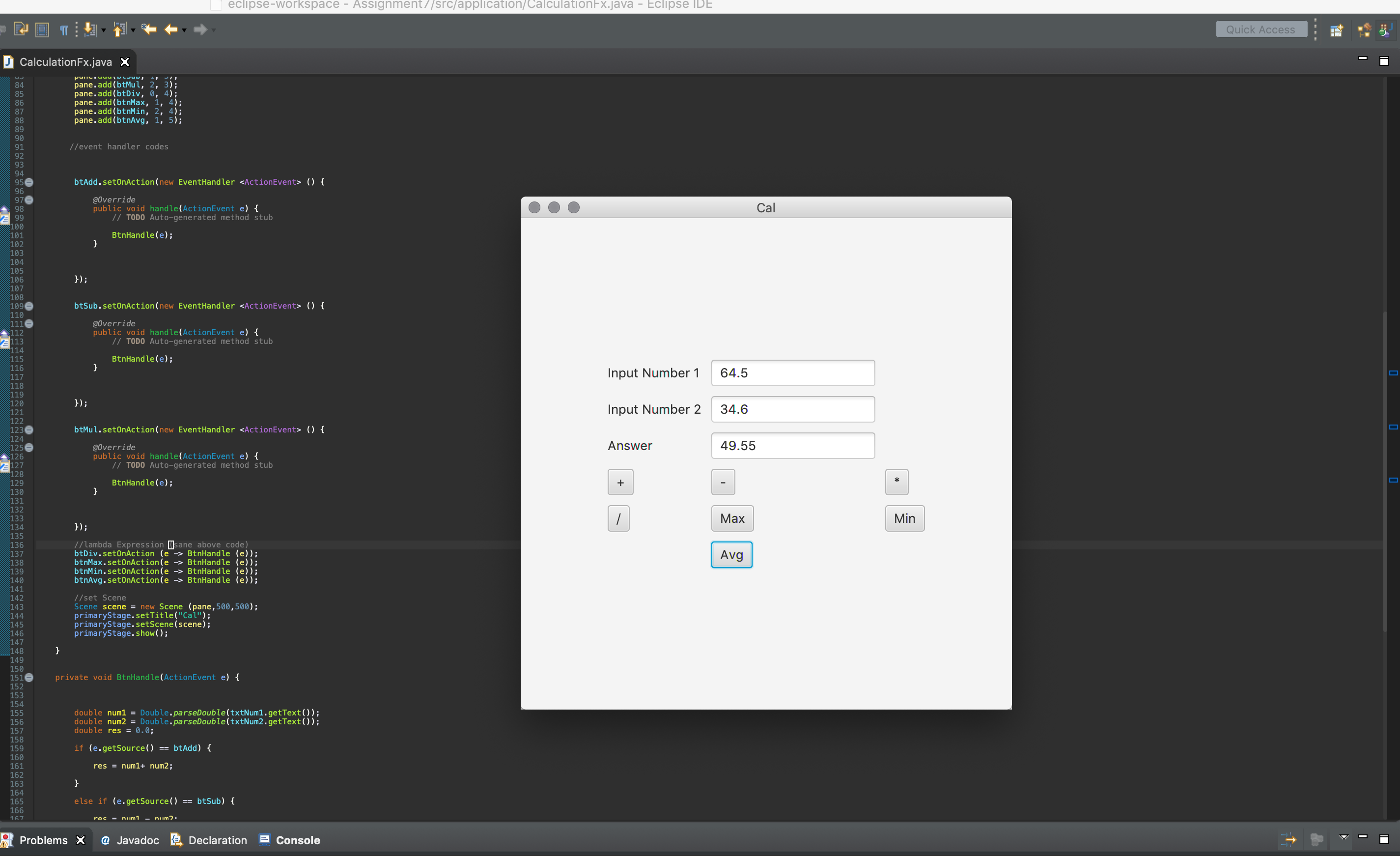
public static void main(String[] args) {

launch(args);

}

}

*Screenshots*



**Question 2( chapter 16)**

Write a GUI program that lets the user select one of the following empty shapes: circle, square, triangle or rectangle and  select one of the following colors: red, blue, green or yellow.

At the beginning, only the borders of the shapes are displayed. When the radio button of shape is selected and the radio button of the color is selected, that selected shape will be filled with the color selected. Only one figure and one color can be selected at a time.

*Code*

/\*

\* Write a GUI program that lets the user select one of the following empty shapes: circle, square,

\* triangle or rectangle and select one of the following colors: red, blue, green or yellow.

At the beginning, only the borders of the shapes are displayed. When the radio button of shape is

selected and the radio button of the color is selected, that selected shape will be filled with the

color selected. Only one figure and one color can be selected at a time.

\*

\*/

package application;

import javafx.application.Application;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.control.RadioButton;

import javafx.scene.control.Toggle;

import javafx.scene.control.ToggleGroup;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.shape.Circle;

import javafx.scene.shape.Polygon;

import javafx.scene.shape.Rectangle;

import javafx.stage.Stage;

public class ShapeControl extends Application{

private Circle shCir;

private Rectangle shSqu;

private Polygon shTri;

private Rectangle shRec;

private ToggleGroup shapeGrp;

private ToggleGroup colorGrp;

@Override

public void start(Stage primaryStage) throws Exception {

shapeGrp = new ToggleGroup();

colorGrp = new ToggleGroup();

BorderPane pane = new BorderPane ();

pane.setPadding(new Insets(10,10,10,10));

//draw Shapes

shCir = new Circle (30);

shSqu = new Rectangle(50,50,50,50);

shTri = new Polygon ();

shTri.getPoints().addAll(new Double [] {

100.0,50.0,

50.0,100.0,

150.0,100.0

});

shRec = new Rectangle(60,30,120,60);

defaultValue();

//shapes

RadioButton rbCir =new RadioButton ("Circle");

RadioButton rbSqu =new RadioButton ("Square");

RadioButton rbTri =new RadioButton ("Triangle");

RadioButton rbRec =new RadioButton ("Rectangle");

rbCir.setToggleGroup(shapeGrp);

rbSqu.setToggleGroup(shapeGrp);

rbTri.setToggleGroup(shapeGrp);

rbRec.setToggleGroup(shapeGrp);

//Color

RadioButton rbRed =new RadioButton ("Red");

RadioButton rbBlu =new RadioButton ("Blue");

RadioButton rbGre =new RadioButton ("Green");

RadioButton rbYel =new RadioButton ("Yellow");

rbRed.setToggleGroup(colorGrp);

rbBlu.setToggleGroup(colorGrp);

rbGre.setToggleGroup(colorGrp);

rbYel.setToggleGroup(colorGrp);

VBox vbox2 = new VBox ();

VBox vbox = new VBox ();

VBox vbox1 = new VBox ();

vbox.setSpacing(10);

vbox1.setSpacing(15);

vbox2.setSpacing(10);

vbox1.setAlignment(Pos.CENTER);

//Left Vbox

vbox.getChildren().add(new Label("Shapes"));

vbox.getChildren().add(rbCir);

vbox.getChildren().add(rbSqu);

vbox.getChildren().add(rbTri);

vbox.getChildren().add(rbRec);

//Right Vbox

vbox2.getChildren().add(new Label("Colors"));

vbox2.getChildren().add(rbRed);

vbox2.getChildren().add(rbBlu);

vbox2.getChildren().add(rbGre);

vbox2.getChildren().add(rbYel);

//Center Vbox

vbox1.getChildren().add(shCir);

vbox1.getChildren().add(shSqu);

vbox1.getChildren().add(shTri);

vbox1.getChildren().add(shRec);

pane.setLeft(vbox);

pane.setCenter(vbox1);

pane.setRight(vbox2);

//if first Color is selected then execute this

colorGrp.selectedToggleProperty().addListener(new ChangeListener<Toggle>() {

@Override

public void changed(ObservableValue<? extends Toggle> ov, Toggle old\_toggle, Toggle new\_toggle) {

if(colorGrp.getSelectedToggle() != null) {

RadioButton selected = (RadioButton)colorGrp.getSelectedToggle();

String color = selected.getText();

shapeChange(color);

}

}

});

//if first Shape is selected then execute this

shapeGrp.selectedToggleProperty().addListener(new ChangeListener<Toggle>() {

@Override

public void changed(ObservableValue<? extends Toggle> ov, Toggle old\_toggle, Toggle new\_toggle) {

if(shapeGrp.getSelectedToggle() != null) {

RadioButton selected = (RadioButton)shapeGrp.getSelectedToggle();

String shape = selected.getText();

colorChange(shape);

}

}

});

//set Scene

Scene scene = new Scene (pane,300,300);

primaryStage.setTitle("Shapes & Color ");

primaryStage.setScene(scene);

primaryStage.show();

}

private void defaultValue() {

shCir.setFill(Color.WHITE);

shSqu.setFill(Color.WHITE);

shTri.setFill(Color.WHITE);

shRec.setFill(Color.WHITE);

shCir.setStroke(Color.BLACK);

shSqu.setStroke(Color.BLACK);

shTri.setStroke(Color.BLACK);

shRec.setStroke(Color.BLACK);

}

public static void main(String[] args) {

launch(args);

}

private void shapeChange(String color) {

shapeGrp.selectedToggleProperty().addListener(new ChangeListener<Toggle>() {

@Override

public void changed(ObservableValue<? extends Toggle> ov, Toggle old\_toggle, Toggle new\_toggle) {

if(shapeGrp.getSelectedToggle() != null) {

RadioButton selected = (RadioButton)shapeGrp.getSelectedToggle();

String shape = selected.getText().toString();

outputChange(color,shape);

}

}

});

}

private void colorChange(String shape) {

colorGrp.selectedToggleProperty().addListener(new ChangeListener<Toggle>() {

@Override

public void changed(ObservableValue<? extends Toggle> ov, Toggle old\_toggle, Toggle new\_toggle) {

if(colorGrp.getSelectedToggle() != null) {

RadioButton selected = (RadioButton)colorGrp.getSelectedToggle();

String color = selected.getText().toString();

outputChange(color,shape);

}

}

});

}

private void outputChange(String color, String shape) {

Color choice = findColor(color);

switch (shape) {

case "Circle": defaultValue();

shCir.setFill(choice);

break;

case "Square": defaultValue();

shSqu.setFill(choice);

break;

case "Triangle": defaultValue();

shTri.setFill(choice);

break;

case "Rectangle":defaultValue();

shRec.setFill(choice);

break;

}

}

private Color findColor(String color) {

Color value = null;

switch (color) {

case "Red" : value = Color.RED; break;

case "Blue" : value = Color.BLUE; break;

case "Green": value = Color.GREEN;break;

case "Yellow":value=Color.YELLOW; break;

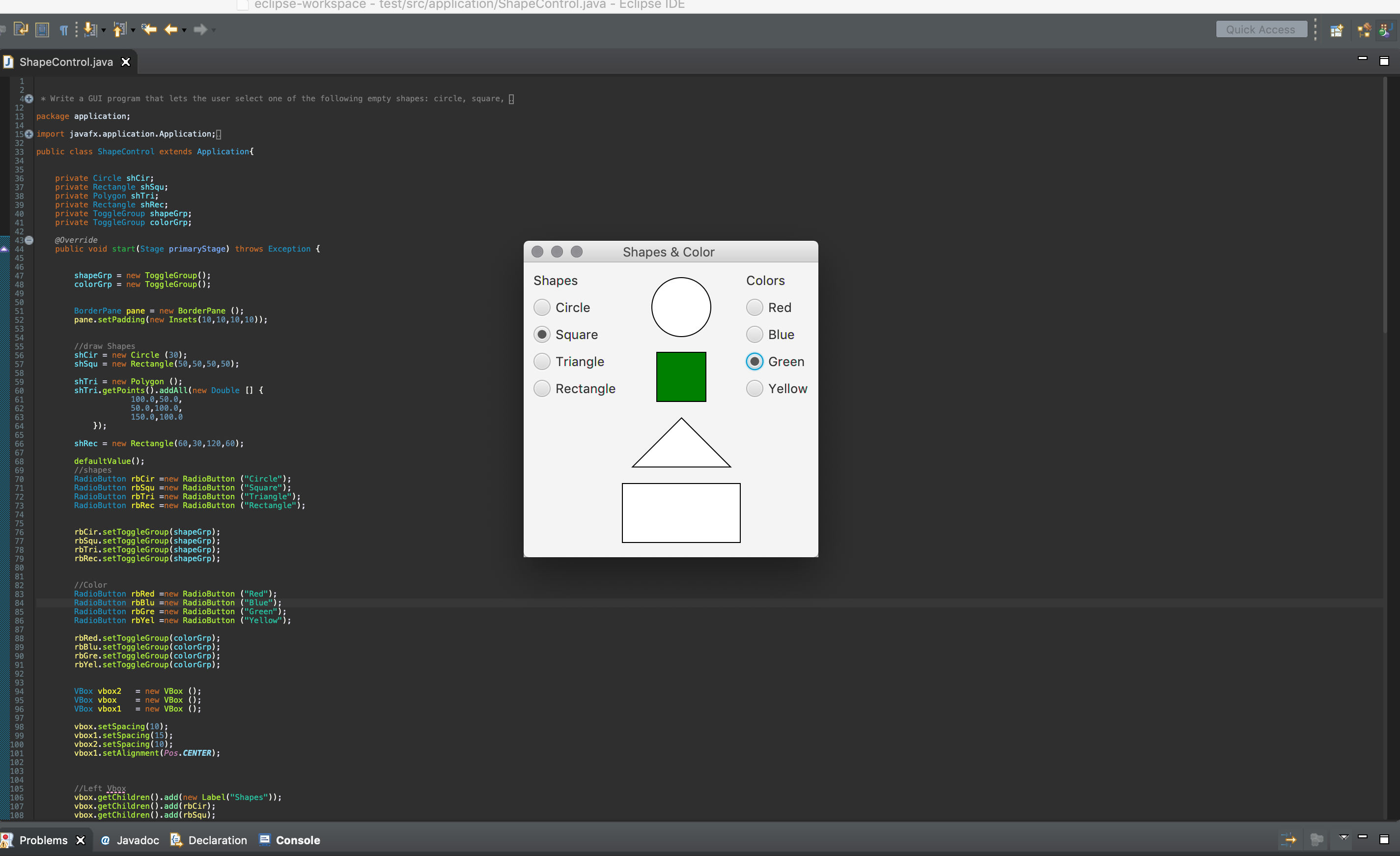
}

return value;

}

}

*Screenshot*

**