**Task 5: JavaFX**

* + To Create Different Layouts with JavaFX animation.
  + To Design a web application with JavaFX event handling.

5.1 Create simple calculator using JavaFx by using VBox Layout

Program:

**import** javafx.application.Application;

**import** javafx.beans.binding.Bindings;

**import** javafx.beans.property.\*;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**import** javafx.scene.control.\*;

**import** javafx.scene.input.KeyEvent;

**import** javafx.scene.layout.\*;

**import** javafx.stage.\*;

**import** java.util.\*;

// a simple JavaFX calculator.

**public** **class** Main **extends** Application {

**private** **static** **final** String[][] ***template*** = {

{"7", "8", "9", "/"},

{"4", "5", "6", "\*"},

{"1", "2", "3", "-"},

{"0", "c", "=", "+"}

};

**private** **final** Map<String, Button> accelerators = **new** HashMap<>();

**private** **final** DoubleProperty stackValue = **new** SimpleDoubleProperty();

**private** **final** DoubleProperty value = **new** SimpleDoubleProperty();

**private** **enum** Op {***NOOP***, ***ADD***, ***SUBTRACT***, ***MULTIPLY***, ***DIVIDE***}

**private** Op curOp = Op.***NOOP***;

**private** Op stackOp = Op.***NOOP***;

**public** **static** **void** main(String[] args) {

*launch*(args);

}

@Override

**public** **void** start(Stage stage) {

**final** TextField screen = createScreen();

**final** TilePane buttons = createButtons();

stage.setTitle("Calc");

stage.initStyle(StageStyle.***UTILITY***);

stage.setResizable(**false**);

stage.setScene(**new** Scene(createLayout(screen, buttons)));

stage.show();

}

**private** VBox createLayout(TextField screen, TilePane buttons) {

**final** VBox layout = **new** VBox(20);

layout.setAlignment(Pos.***CENTER***);

layout.setStyle("-fx-background-color: chocolate; -fx-padding: 20; -fx-font-size: 20;");

layout.getChildren().setAll(screen, buttons);

handleAccelerators(layout);

screen.prefWidthProperty().bind(buttons.widthProperty());

**return** layout;

}

**private** **void** handleAccelerators(VBox layout) {

layout.addEventFilter(KeyEvent.***KEY\_PRESSED***, keyEvent -> {

Button activated = accelerators.get(keyEvent.getText());

**if** (activated != **null**) {

activated.fire();

}

});

}

**private** TextField createScreen() {

**final** TextField screen = **new** TextField();

screen.setStyle("-fx-background-color: aquamarine;");

screen.setAlignment(Pos.***CENTER\_RIGHT***);

screen.setEditable(**false**);

screen.textProperty().bind(Bindings.*format*("%.0f", value));

**return** screen;

}

**private** TilePane createButtons() {

TilePane buttons = **new** TilePane();

buttons.setVgap(7);

buttons.setHgap(7);

buttons.setPrefColumns(***template***[0].length);

**for** (String[] r : ***template***) {

**for** (String s : r) {

buttons.getChildren().add(createButton(s));

}

}

**return** buttons;

}

**private** Button createButton(**final** String s) {

Button button = makeStandardButton(s);

**if** (s.matches("[0-9]")) {

makeNumericButton(s, button);

} **else** {

**final** ObjectProperty<Op> triggerOp = determineOperand(s);

**if** (triggerOp.get() != Op.***NOOP***) {

makeOperandButton(button, triggerOp);

} **else** **if** ("c".equals(s)) {

makeClearButton(button);

} **else** **if** ("=".equals(s)) {

makeEqualsButton(button);

}

}

**return** button;

}

**private** ObjectProperty<Op> determineOperand(String s) {

**final** ObjectProperty<Op> triggerOp = **new** SimpleObjectProperty<>(Op.***NOOP***);

**switch** (s) {

**case** "+" -> triggerOp.set(Op.***ADD***);

**case** "-" -> triggerOp.set(Op.***SUBTRACT***);

**case** "\*" -> triggerOp.set(Op.***MULTIPLY***);

**case** "/" -> triggerOp.set(Op.***DIVIDE***);

}

**return** triggerOp;

}

**private** **void** makeOperandButton(Button button, **final** ObjectProperty<Op> triggerOp) {

button.setStyle("-fx-base: lightgray;");

button.setOnAction(actionEvent -> curOp = triggerOp.get());

}

**private** Button makeStandardButton(String s) {

Button button = **new** Button(s);

button.setStyle("-fx-base: beige;");

accelerators.put(s, button);

button.setMaxSize(Double.***MAX\_VALUE***, Double.***MAX\_VALUE***);

**return** button;

}

**private** **void** makeNumericButton(**final** String s, Button button) {

button.setOnAction(actionEvent -> {

**if** (curOp == Op.***NOOP***) {

value.set(value.get() \* 10 + Integer.*parseInt*(s));

} **else** {

stackValue.set(value.get());

value.set(Integer.*parseInt*(s));

stackOp = curOp;

curOp = Op.***NOOP***;

}

});

}

**private** **void** makeClearButton(Button button) {

button.setStyle("-fx-base: mistyrose;");

button.setOnAction(actionEvent -> value.set(0));

}

**private** **void** makeEqualsButton(Button button) {

button.setStyle("-fx-base: ghostwhite;");

button.setOnAction(actionEvent -> {

**switch** (stackOp) {

**case** ***ADD*** -> value.set(stackValue.get() + value.get());

**case** ***SUBTRACT*** -> value.set(stackValue.get() - value.get());

**case** ***MULTIPLY*** -> value.set(stackValue.get() \* value.get());

**case** ***DIVIDE*** -> value.set(stackValue.get() / value.get());

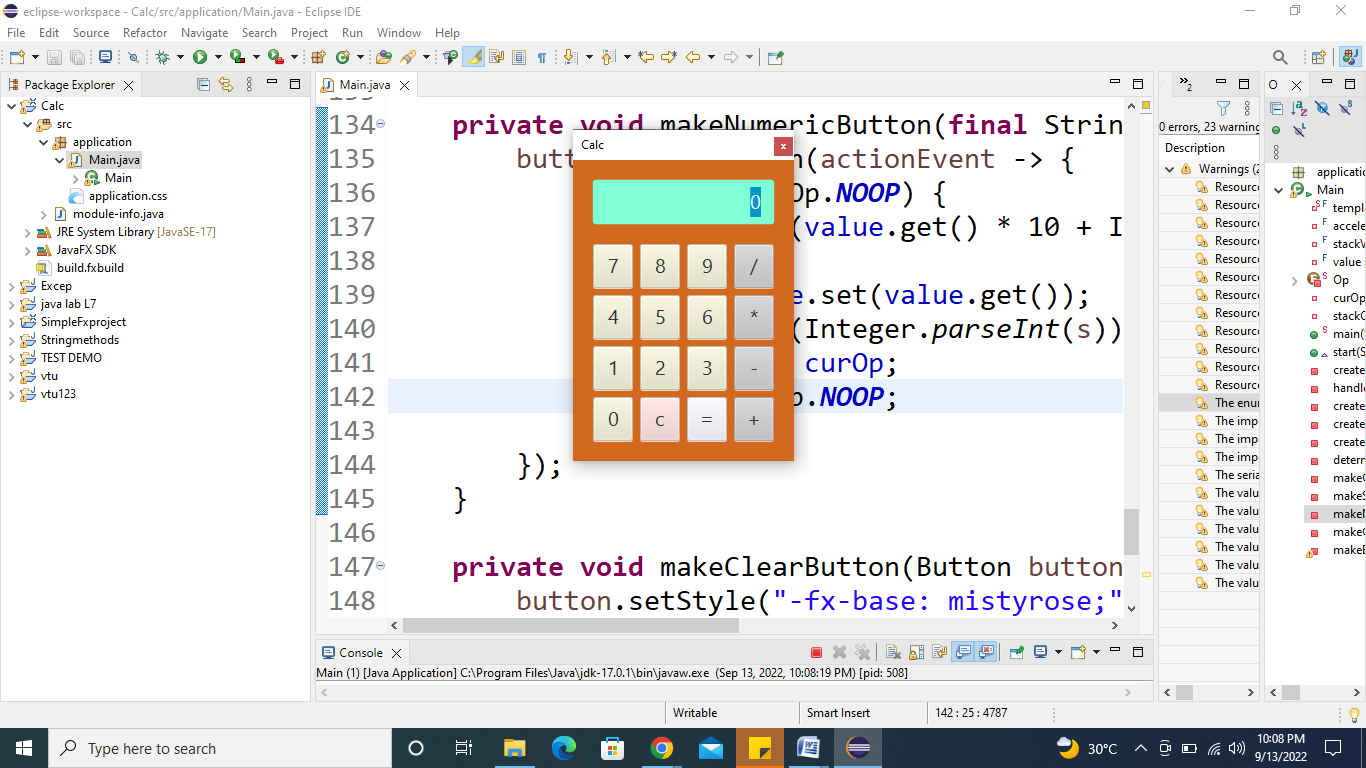
}

});

}

}

**Output:**



5.2 Create event handler with two different buttons. The event source is discriminated in the **handle()** method. The circle starts translating in the positive X direction when the **Play** button is clicked while It gets paused when the **Pause** button is clicked.

Program:

**import** javafx.animation.TranslateTransition;

**import** javafx.application.Application;

**import** javafx.event.EventHandler;

**import** javafx.scene.Group;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Button;

**import** javafx.scene.input.MouseEvent;

**import** javafx.scene.paint.Color;

**import** javafx.scene.shape.Circle;

**import** javafx.stage.Stage;

**import** javafx.util.Duration;

**public** **class** Main **extends** Application{

@Override

**public** **void** start(Stage primaryStage) **throws** Exception {

// **TODO** Auto-generated method stub

//Creating Circle and setting the color and stroke in the circle

Circle c = **new** Circle(100,100,50);

c.setFill(Color.***GREEN***);

c.setStroke(Color.***BLACK***);

//creating play button and setting coordinates for the button

Button btn = **new** Button("Play");

btn.setTranslateX(125);

btn.setTranslateY(200);

// creating pause button and setting coordinate for the pause button

Button btn1 = **new** Button("Pause");

btn1.setTranslateX(175);

btn1.setTranslateY(200);

//Instantiating TranslateTransition class to create the animation

TranslateTransition trans = **new** TranslateTransition();

//setting attributes for the TranslateTransition

trans.setAutoReverse(**true**);

trans.setByX(200);

trans.setCycleCount(100);

trans.setDuration(Duration.*millis*(500));

trans.setNode(c);

//Creating EventHandler

EventHandler<MouseEvent> handler = **new** EventHandler<MouseEvent>() {

@Override

**public** **void** handle(MouseEvent event) {

// **TODO** Auto-generated method stub

**if**(event.getSource()==btn)

{

trans.play(); //animation will be played when the play button is clicked

}

**if**(event.getSource()==btn1)

{

trans.pause(); //animation will be paused when the pause button is clicked

}

event.consume();

}

};

//Adding Handler for the play and pause button

btn.setOnMouseClicked(handler);

btn1.setOnMouseClicked(handler);

//Creating Group and scene

Group root = **new** Group();

root.getChildren().addAll(c,btn,btn1);

Scene scene = **new** Scene(root,420,300,Color.***WHEAT***);

primaryStage.setScene(scene);

primaryStage.setTitle("EventHandler example");

primaryStage.show();

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

Output:

