UNIT V

JAVAFX EVENT HANDLING, CONTROLS & COMPONENTS

JavaFx Events and controls: Event Basics - Handling Key and Mouse Events. controls: checkBox, ToggleButton - RadioButtons - ListView - ComboBox - choiceBox - Text controls - ScrollPane. layouts - FlowPane - HBOX and VBOX - Border Pane - StackPane - GridPane. Menus - Basics - Menu - Menubars - MenuItem.

---

1. JavaFx Events and controls:

Basics:

JavaFx is a set of packages that allow Java Programmers to create rich graphics and media applications such as 2D and 3D games, animations etc,

Advantages:

1. Hardware-accelerated graphics
2. High performance media engine.

Types of event.

1. Foreground Events
2. Background Events

→ JavaFx, event is an instance of the javaFx.event.Event class.

JavaFX provides several events.

* Drag Event
* Mouse Event
* Key Event.

Events are handled by implementing the EventHandler interface. which is in javafx.event.

interface EventHandler <T extends Event>

HANDLING KEY AND MOUSE EVENTS:

Key Event:

This is an input event that indicates the key stroke occured on a node. It is represented by the class named KeyEvent. This includes actions like key pressed, key released and key typed.

Mouse Event:

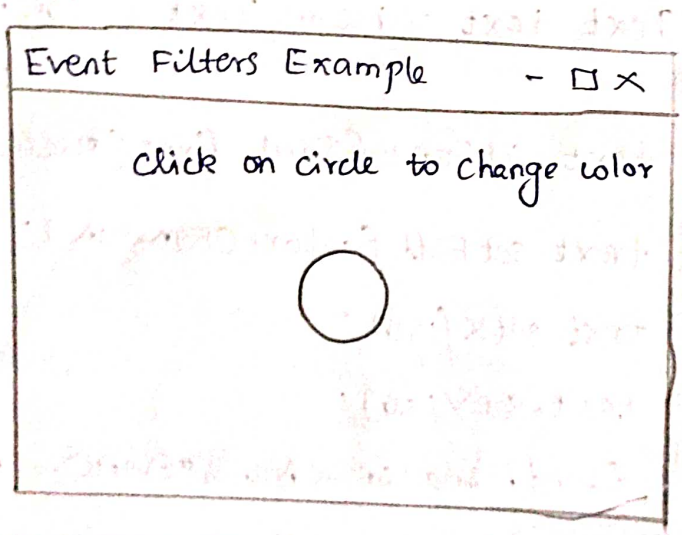
This is an input event that occurs when a mouse is clicked. It is represented by the class named MouseEvent. It includes actions like mouse clicked, mouse pressed, mouse released, mouse moved, etc.

xample:

```
import javaFx.application.Application,
import static javafx.application.Application.launch;
import javafx.event.EventHandler;
import javafx.Scene.*;
Public class Example extends Application
{
    public void start(Stage stage)
    {
        Circle circle = new Circle();
        Circle.setCenterX(300.0f);
        circle.setcenterY(135.0f);
        circle.setRadius(25.0f);
        circle.setFill(color.BROWN);
        circle.setStrokeWidth(20);
        Text text = new Text("click on circle to
                                change colour");
        text.setFont(Font font(null, FontWidth BOLD, 15));
        text.setFill(color.CRIMSON);
        text.setX(150);
        text.setY(150);
        EventHandler<MouseEvent> eventHandler = new
                    EventHandler<MouseEvent>()
        {
            public void handle(MouseEvent e)
            {
                System.o.pln("Hello World");
                Circle.setFill(color.DARKSLATEBLUE);
            }
        };
```

```
Circle.addEventFilter(MouseEvent.MOUSE_CLICKED, eventHan
Group root = new Group(circle, text);
Scene scene = new Scene(root, 600, 300);
Scene.setFill(color.LAVENDER);
Stage.setTitle("Event Filters Example");
Stage.setScene(scene);
Stage.show();
}
Public Static void main(String args[])
{
    launch(args);
}
}
```

Output:



2. CONTROLS : checkBox :

The checkBox Group component allows the user to make one and only one selection at a time.

Syntax:

CheckBox (String str, checkBoxGroup cbg, Boolean val);

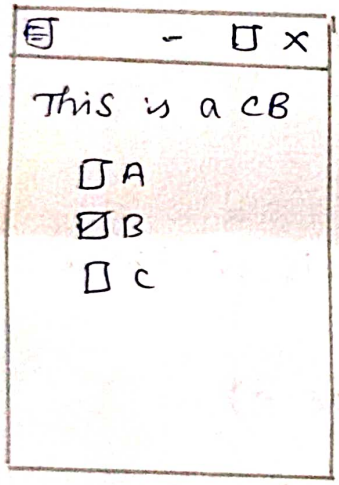
check Box can also called as Radio Buttons.

Example:

```
import java.awt.*;
import javafx.application.Application;
import javafx.scene.*;
import javafx.event.*;
import javafx.collections.*;
import javafx.stage.Stage;
public class checkbox_1 extends Application
{
    public void start(Stage s)
    s.setTitle("Creating CB");
    TilePane r = new TilePane();
    Label L = new Label("This is CB");
    String st[] = {"A", "B", "C"};
    r.getchildrec().add(l);
    for(int i=0; i < st.length; i++)
    {
        checkBox c = new checkBox(st[i]);
        r.getchildrec().add(c);
        c.setIndeterminate(true);
    }
```

```
        Scene sc = new Scene (r, 150, 200);
        s.setScene(sc);
        s.show();
    }
public static void main(String args[])
{
    launch(args);
}
}
```

Output:



JavaFx ToggleButton:

It is a button that can be selected or not selected. It is represented in the class javafx.scene.control.ToggleButton.

Example:

```
ToggleButton toggleButton1 = new ToggleButton ("Left");
```

Radio Button:

It is similar to ToggleButton but the difference is RadioButton cannot be unselected.

Example:

```
RadioButton rb = new RadioButton("Left");
```

LISTVIEW:

Listview control enables user to choose one or more options from a predefined list of choices.

Example:

```
Listview listview = new Listview();
```

Adding Items:

```
listview.getItems().add("Item 1");
```

COMBO BOX:

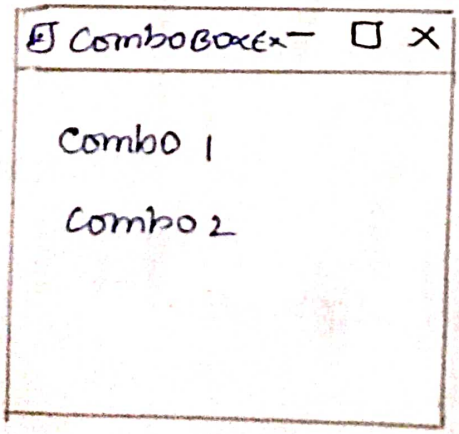
ComboBox control enables users to choose an option from a predefined list of choices. or type in another value if none of the predefined choices matches what the user want to select.

Example:

```
import javafx.application.Application;
import javafx.Scene.*;
import javafx.stage.Stage;
```

```
Public class comboBox1 extends Applica
{
    Public void start (stage primarystage) thro
                                        Exception
    {
        Primarystage.setTitle ("comboBox Ex-1");
        ComboBox comboBox = new comboBox();
        comboBox.getItems().add ("ch 1");
        comboBox.getItems().add ("ch 2");
        HBox hbox = new HBOX (comboBox);
        Scene scene = new Scene (hBox, 200, 120);
        primary staye.setscene (scene);
        primaryStage.show();
    }
    psvm()
    {
        Application.launch (arg);
    }
}
```

Output:



CHOICE BOX:

→ It enables user to choose an option from a predefined list of choices.

→ It is represented by the class

javafx.scene.control.choiceBox.

Creating ChoiceBox:

```
ChoiceBox choiceBox = new choiceBox();
```

Adding choices:

```
choiceBox.getItems().add("choice 1");
ChoiceBox.getItems().add("choice 2");
ChoiceBox.getItems().add("choice 3");
```

TEXT CONTROLS:

→ TextField controls enable users of a JavaFx application to enter text which can then be read by the application.

→ It is represented in the class

javafx.Scene.control.TextField

creating a TextField:

```
TextFiled textfield = new Textfield();
```

Adding:

The text field object must be added to the scene graph.

SCROLL PANE:

JavaFX Scroll pane control is a con
that has two scrollbars. The scroll bar ena
the user to scroll around the components
shown inside the scrollPane, so that different
parts of the components can be seen.

It is represented by the JavaFx class.

javafx.scene.control.scrollpane.

Create a Scroll Pane:

```
ScrollPane scrollpane = new ScrollPane();
```

---

3. LAYOUTS:

JavaFx provides several predefined layouts such as HBox, VBox, BorderPane, Flow Pane, Stack Pane and GridPane.

FLow Pane:

→ JavaFx FlowPane is a layout component which lays out the child componends either vertically or horizontally.

→ It is represented by the class

javafx.scene.layout.FlowPane

Creating a FlowPane:

```
FlowPane flowpane = new FlowPane();
```

Adding children:

```
Button button1=new Button ("Button NO 1");
Button button2 = new Button ("Button NO 2");
FlowPane flowpane = new FlowPane();
flowpane.getchildren().add(button1);
```

HBox and VBox:

HBox:

→ HBox component is a layout component which positions all its child nodes in a horizontal row.

→ It is represented by the class javafx.scene.layout.HBox.

Creating HBox:

```
HBox hbox = New HBox();
Button b1 = new Button ("Button No 1");
```

V Box:

It is a layout component which positions all its child nodes in a vertical column on top of each other.

→ It is represented by the class

javafx.scene.layout.VBOX.

creating a VBox:

```
VBox vbox = new VBox();
```

Border Pane:

→ BorderPane class lays its children in top, bottom, Center, left and right positions.

→ It also inherits Pane class.

Constructors:

* Border pane()
* Border pane (Node c)
* BorderPane (Node center, Node top, Node right, Node botton, Node left);

StackPane:

→ StackPane places all the nodes into a single stack where every new node gets placed on the top of the previous node.

→ It is represented in

javafx.scene.layout.Stackpane.

Constructors:

1. Stackpane()
2. Stackpane (Node ? children)

Grid pane:

Grid Pane is a layout component in a grid. The size of the cells in the grid depends on the components displayed in the GridPane.

Rules:

All cells in the same row will have same height and the cells in the same column will have the same width.

Creating a Grid Pane:

```
GridPane gridPane = new GridPane();
```

Adding Children:

```
gridPane.add(button1, 0, 0, 1, 1);
```

---

4. MENUS: BASICS:

Menu:

Menu Bar is the main component of any application. In JavaFx, javafx.scene.control, menu class provides all the method to deal with menus.

Menu Bar:

MenuBar Provides Javafx application with a Visual dropdown menu similar to that most desktop application.

Creating MenuBar Instance:

```
MenuBar menuBar = new MenuBar();
```

Menu Items:

Once you have created a Menu instance you must add one or more Menu Item instances to it.

Example:

```
Menu menu = new Menu("Menu 1");
MenuItem menuItem1 = new MenuItem("Item 1");
MenuItem menuItem2 = new MenuItem("Item 2")
menu.getItems().add(menuItems1);
menu.getItems().add(menuItems2);
MenuBar menuBar = new MenuBar();
menuBar.getMenus().add(menu);
```