VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

JavaFX Events

Objectives

This session will give the knowledge about

- JavaFX Events
- JavaFX ActionEvent
- JavaFX KeyEvent
- JavaFX MouseEvent

JavaFX Event Handling

In JavaFX, An event is occurred whenever the user interacts with the application nodes. There are various sources by using which, the user can generate the event.

Types of Events:

- Foreground Events: mainly occurred due to the direct interaction of the user
- Background Events: mainly occurred to the operating system interrupts, failure, operation completion, etc.

Processing Events in JavaFX

- JavaFX provides the mechanism to capture the events, route the event to its target and letting the application handle the events.
- JavaFX provides the class javafx.event.Event which contains all the subclasses representing the types of Events that can be generated in JavaFX. Any event is an instance of the class Event or any of its subclasses.
- There are various events in JavaFX i.e. MouseEvent, KeyEvent, ScrollEvent, DragEvent, etc. We can also define our own event by inheriting the class javafx.event.Event.

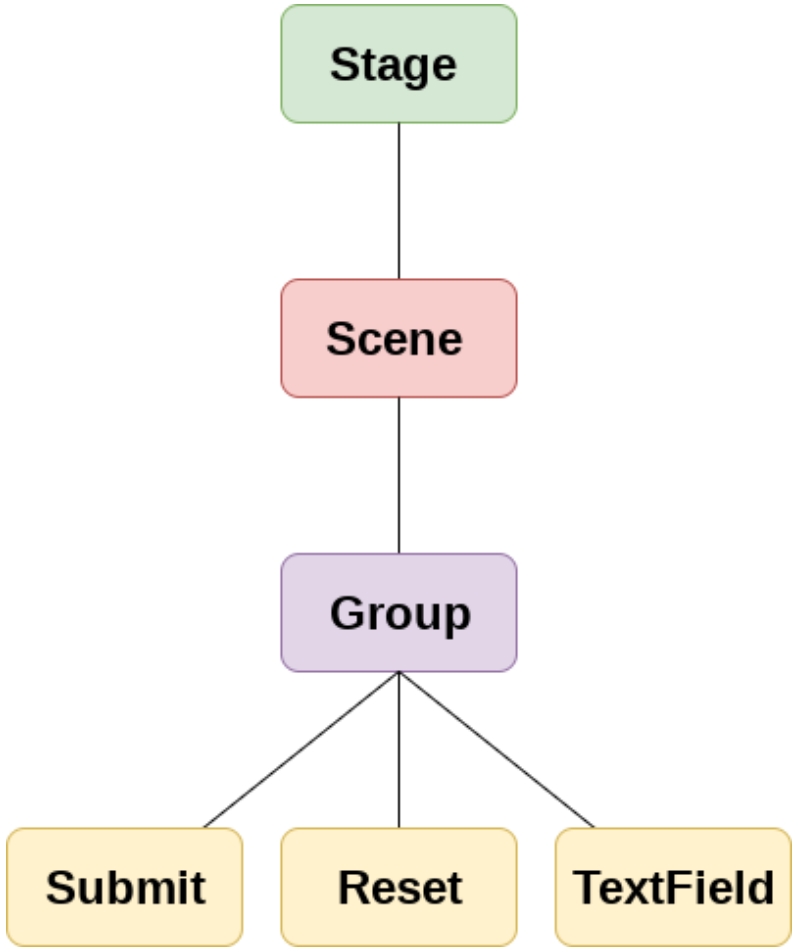
Properties of an Event

- **Event Type** — It is the type of the event that is being generated. It is basically the instance of EventType class. It is hierarchical. EventType: KeyEvent class contains KEY_PRESSED, KEY_RELEASED, and KEY_TYPED types.
- **Source** — It represents source of the event i.e. the origin which is responsible to generate the event.
- **Target** — It is the node on which the event is generated. It remains unchanged for the generated event. It is the instance of any of the class that implements the EventTarget interface.

Event Delivery Process

Step-1: Route Construction

- An Event Dispatch Chain is created in order to determine the default route of the event, whenever it is generated.
- The Event dispatch chain contains the path from the stage to the Node on which the event is generated.



Event Delivery Process

Step-2: Event Capturing Phase

- Once the Event Dispatch Chain is created, the event is dispatched from the source node of the event.
- All the nodes are traversed by the event from top to bottom. If the event filter is registered with any of these nodes, then it will be executed.

Event Delivery Process

Step-3: Event Bubbling

- Once the event is processed by the target node or by any of the registered filter, the event traverses all the nodes again from the bottom to the stage node. If any of these nodes are registered with the event filter, then it will get executed otherwise the process gets completed.

Step-4: Event Handlers and Filters

- Event Handlers and filters contains application logic to process an event. A node can be registered to more than one Event Filters. The interface javafx.event.EventHandler must be implemented by all the event handlers and filters.

JavaFX ActionEvent – implements ActionEvent

```
public class ButtonAction extends Application implements
EventHandler<ActionEvent> {
        Button btn;
        Label lb;
        @Override
        public void start(Stage primaryStage) {
                btn = new Button();
                btn.setText("Click");
                btn.setOnAction(this);

                lb = new Label();
```

JavaFX ActionEvent – implements ActionEvent

```
                //layout, scene, stage properties
        }
        @Override
        public void handle(ActionEvent e) {
                lb.setText("Welcome");
        }
  //main method
}
```

JavaFX ActionEvent – inline action definition

setOnAction

public final void setOnAction(EventHandler<ActionEvent> value)

Sets the value of the property onAction.

Property description:

The button's action, which is invoked whenever the button is fired. Thismay be due to the user clicking on the button with the mouse, or bya touch event, or by a key press, or if the developer programmaticallyinvokes the fire() method.

JavaFX ActionEvent – inline action definition

```
public class ActionEventDemo extends Application{
        Button btn;
        Label lb;
        @Override
        public void start(Stage primaryStage) {
                lb = new Label();
                btn = new Button();
                btn.setText("Click");
                btn.setOnAction(new EventHandler<ActionEvent>() {
```

JavaFX ActionEvent – inline action definition

```
                @Override
                public void handle(ActionEvent event) {
                    lb.setText("Welcome");
                }
            });
            //layout, scene, stage properties
        }
        //main method
}
```

JavaFX KeyEvent

setOnKeyPressed

public final void setOnKeyPressed(EventHandler<? super KeyEvent> value)

Sets the value of the property onKeyPressed.

Property description:

Defines a function to be called when this Node or its child Node has input focus and a key has been pressed. The functionis called only if the event hasn't been already consumed during itscapturing or bubbling phase.

JavaFX KeyEvent

setOnKeyReleased
public final void setOnKeyReleased(EventHandler<? super KeyEvent>
value)

Sets the value of the property onKeyReleased.

Property description:

Defines a function to be called when this Node or its child Node has input focus and a key has been released. The functionis called only if the event hasn't been already consumed during itscapturing or bubbling phase.

JavaFX KeyEvent

setOnKeyTyped

public final void setOnKeyTyped(EventHandler<? super KeyEvent> value)

Sets the value of the property onKeyTyped.

Property description:

Defines a function to be called when this Node or its child Node has input focus and a key has been typed. The functionis called only if the event hasn't been already consumed during itscapturing or bubbling phase.

JavaFX KeyEvent

```
//import required packages
public class KeyboardEvent extends Application {
        @Override
        public void start(Stage primaryStage) throws Exception {
                TextField tf1 = new TextField();
                Label l1 = new Label();
                Label l2 = new Label();
                Label l3 = new Label();
```

JavaFX KeyEvent

```
// Handling KeyEvent for textfield 1
tf1.setOnKeyPressed(new EventHandler<KeyEvent>() {
        public void handle(KeyEvent key) {
                l1.setText("Key Pressed :" + " " + key.getText());
                l1.setTextFill(Color.GREEN);
        } });
tf1.setOnKeyReleased(new EventHandler<KeyEvent>() {
        public void handle(KeyEvent key) {
                l2.setText("Key Released :" + " " + key.getText());
                l2.setTextFill(Color.RED);
        } });
```

JavaFX KeyEvent

```
tf1.setOnKeyTyped(new EventHandler<KeyEvent>() {
        public void handle(KeyEvent key) {
                // TODO Auto-generated method stub
                l3.setText("Key Typed :" + " " + tf1.getText());
                l3.setTextFill(Color.BLUE);
        }       });
// setting group and scene
VBox root = new VBox();
root.getChildren().addAll(tf1, l1, l2, l3);
Scene scene = new Scene(root, 500, 200, Color.WHEAT);
primaryStage.setScene(scene);
```

JavaFX KeyEvent

```
            primaryStage.setTitle("Handling KeyEvent");
            primaryStage.show();
        }

        public static void main(String[] args) {
            launch(args);
        }

}
```

JavaFX MouseEvent

setOnMouseDragEntered

public final void setOnMouseDragEntered(EventHandler<? super MouseDragEvent> value)

Sets the value of the property onMouseDragEntered.

Property description:

Defines a function to be called when a full press-drag-release gestureenters this Node.

Since:JavaFX 2.1

JavaFX MouseEvent

setOnMouseDragExited

public final void setOnMouseDragExited(EventHandler<? super MouseDragEvent> value)

Sets the value of the property onMouseDragExited.

Property description:

Defines a function to be called when a full press-drag-release gestureleaves this Node.

Since:JavaFX 2.1

JavaFX MouseEvent

setOnMouseDragOver

public final void setOnMouseDragOver(EventHandler<? super MouseDragEvent> value)

Sets the value of the property onMouseDragOver.

Property description:

Defines a function to be called when a full press-drag-release gestureprogresses within this Node.

Since:JavaFX 2.1

JavaFX MouseEvent

setOnMouseDragReleased

public final void setOnMouseDragReleased(EventHandler<? super MouseDragEvent> value)

Sets the value of the property onMouseDragReleased.

Property description:

Defines a function to be called when a full press-drag-release gestureends (by releasing mouse button) within this Node.

Since:JavaFX 2.1

JavaFX MouseEvent

setOnMouseEntered

public final void setOnMouseEntered(EventHandler<? super MouseEvent> value)

Sets the value of the property onMouseEntered.

Property description:

Defines a function to be called when the mouse enters this Node.

JavaFX MouseEvent

setOnMouseExited

public final void setOnMouseExited(EventHandler<? super MouseEvent> value)

Sets the value of the property onMouseExited.

Property description:

Defines a function to be called when the mouse exits this Node.

JavaFX MouseEvent

setOnMouseMoved

public final void setOnMouseMoved(EventHandler<? super MouseEvent> value)

Sets the value of the property onMouseMoved.

Property description:

Defines a function to be called when mouse cursor moves within this Node but no buttons have been pushed.

JavaFX MouseEvent

setOnMousePressed

public final void setOnMousePressed(EventHandler<? super MouseEvent> value)

Sets the value of the property onMousePressed.

Property description:

Defines a function to be called when a mouse buttonhas been pressed on this Node.

JavaFX MouseEvent

setOnMouseReleased

public final void setOnMouseReleased(EventHandler<? super MouseEvent> value)

Sets the value of the property onMouseReleased.

Property description:

Defines a function to be called when a mouse buttonhas been released on this Node.

JavaFX MouseEvent

```
//import required packages
public class MouseEventDemo extends Application {
        @Override
        public void start(Stage primaryStage) throws Exception {
                Button button = new Button("how is it");
                button.setStyle("-fx-font-size: 4em; ");
                button.setOnMouseEntered(new EventHandler<MouseEvent>() {
                        @Override
                        public void handle(MouseEvent event) {
                                button.setTextFill(Color.RED);
                        }       });
```

JavaFX MouseEvent

```
button.setOnMouseExited(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
                button.setTextFill(Color.GREEN);
        }       });

button.setOnMouseClicked(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
                button.setTextFill(Color.BLUE);
        }       });
```

JavaFX MouseEvent

```
        HBox root = new HBox(button);
        Scene scene = new Scene(root, 400, 100);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        Application.launch(args);
    }
}
```

JavaFX Getting Inputs

```
            HBox root = new HBox(button);
            Scene scene = new Scene(root, 400, 100);
            primaryStage.setScene(scene);
            primaryStage.show();
        }

        public static void main(String[] args) {
            Application.launch(args);
        }
}
```

Summary

We have discussed about

- JavaFX Events
- JavaFX ActionEvent
- JavaFX KeyEvent
- JavaFX MouseEvent