JavaFX2: Key Events and Slider

In our last lecture we covered Mouse Event and its handling. Today we are will cover Key Events. JavaFX programs - GUI applications, web applications and graphical applications, all such applications allow interaction with the application (nodes). On all such occasion, an event is said to have been occurred.

In addition to this note, more reading materials and example are there on power point slide 7, starting slide #137.

Following Key Events we will look into Sliders in JavaFX.

# Key Events:

A Key Event in JavaFX is an input event that indicates that a key stroke occurred on a Node. It is represented by the class named KeyEvent. This event includes actions like:

1. Key pressed
2. Key released
3. Key typed.

Depending on the type of the event it is passed to onKeyPressed, onKeyReleased or onKeyTyped function. The ‘Key typed’ is higher-level and it’s generated whenever a Unicode character is entered e.g ‘a’ and/or ‘A’.

‘KeyPressed’ is fired whenever any key press occurs. ‘KeyTyped’ is fired when a key is pressed that can be converted into a unicode character. If the shift key is down, for example, pressing "a" will tell ‘KeyTyped’ that you typed a capital A, and ‘KeyPressed’ will just get the "a" key, without capital or lowercase designations.

Other types of events handled by JavaFX: Mouse event represented by MouseEvent, Drag event represented by DragEvent and Window event represented by WindowEvent.

Go to this page: [KeyEvent (JavaFX8)](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/input/KeyEvent.html) for details on the constructor and the methods supported.

## Example1

Write a JavaFX application with a button. When it’s pressed by the ‘Enter’ key, a message is printed.

**import** javafx.application.Application;

**import** javafx.event.EventHandler;

**import** javafx.scene.Scene;

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

**import** javafx.scene.input.KeyCode;

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

**import** javafx.scene.layout.BorderPane;

**import** javafx.stage.Stage;

**public** **class** KeyExample **extends** Application {

@Override

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

BorderPane pane = **new** BorderPane();

Button button = **new** Button("Press Me!");

pane.setCenter(button);

Scene scene = **new** Scene(pane, 200, 200);

primaryStage.setScene(scene);

primaryStage.show();

button.setOnKeyPressed(**new** EventHandler<KeyEvent>() {

@Override

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

**if** (event.getCode() == KeyCode.***ENTER***) {

System.***out***.println("Enter Pressed");

}

}

});

}

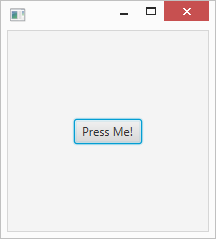
**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

Run the program and you will get the following output. Hit the ‘Enter’ key and the string “Enter Pressed” will be displayed on the console.



## Description

The getCode() method gets the key code associated with the key. It’s either the ‘key pressed’ or ‘key released’ event.

# Slider:

Slider is a Control in JavaFX which is used to display a continuous or discrete range of valid numeric choices and allows the user to interact with the control. It can be placed as a vertical or horizontal bar with a knob that the user can slide to indicate the desired value. It can also have tick marks and labels to indicate the intervals along the bar.

The three fundamental variables of the slider are min, max, and value. The value should always be a number within the range defined by min and max. min should always be less than to max. min defaults to 0, whereas max defaults to 100.

Create a slider:

Slider (double min, double max, double value)

Constructs a Slider control with the specified slider min, max and current value values.

import javafx.scene.control.Slider;

Slider slider = new Slider(0, 100, 5);

The min value here is 0, max is 100 and the slider defaults to 5.

Slider with customized tick marks and tick mark labels which spans from 0 to 1 and defaults to 0.5

import javafx.scene.control.Slider;

Slider slider = new Slider(0, 1, 0.5);

slider.setShowTickMarks(true);

slider.setShowTickLabels(true);

slider.setMajorTickUnit(0.25f);

slider.setBlockIncrement(0.1f);

For the methods in Slider go to [Slider (JavaFX8)](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Slider.html)

## Example1

Write a JavaFX program to implement the Slider Class

**import** javafx.application.Application;

**import** javafx.scene.Group;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Slider;

**import** javafx.stage.Stage;

**public** **class** SliderBasic **extends** Application {

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

// creating group

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

Scene scene = **new** Scene(root, 300, 200);

// set Scene to the stage

stage.setScene(scene);

// set title for the frame

stage.setTitle("Slider Sample");

// create slider

Slider slider = **new** Slider();

// add slider to the frame

root.getChildren().add(slider);

stage.show();

}

// Main Method

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

{

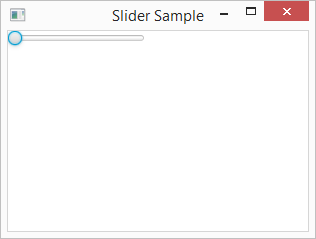
// launch the application

*launch*(args);

}

}

Run the program and you get the following output. Notice the default position of *min*, *max* and *default*.



## Example2

Write a JavaFX program to implement the Slider Class that uses TickMarks and TickLabels

**import** javafx.application.Application;

**import** javafx.scene.Group;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Slider;

**import** javafx.stage.Stage;

**public** **class** SliderTick **extends** Application {

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

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

// create a Scene

Scene scene = **new** Scene(root, 300, 200);

// add Scene to the frame

stage.setScene(scene);

// set title of the frame

stage.setTitle("Slider Sample");

// Creates a slider, min=0, max=1, default=0.5

Slider slider = **new** Slider(0, 1, 0.5);

// enable the marks

slider.setShowTickMarks(**true**);

// enable the Labels

slider.setShowTickLabels(**true**);

// set Major tick unit

slider.setMajorTickUnit(0.25f);

// sets the value of the property blockIncrement

slider.setBlockIncrement(0.1f);

root.getChildren().add(slider);

// display

stage.show();

}

// Main Method

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

{

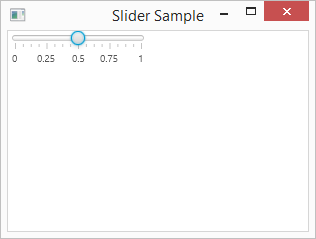
// Launch the application

*launch*(args);

}

}

Run the program and you will get the following output. Notice the tick marks on the slider.



### Description

The root.getChildren()gets the list of children of this Group object ‘root’.

root.getChildren().add(slider); adds the slider to the root node.

## Example3

Write a JavaFX program to create a slider and a circle which are linked by the Translate class and the values are displayed when slided along the x-axis.

**import** javafx.application.Application;

**import** javafx.beans.value.ChangeListener;

**import** javafx.beans.value.ObservableValue;

**import** javafx.geometry.Insets;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Slider;

**import** javafx.scene.layout.VBox;

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

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

**import** javafx.scene.transform.Translate;

**import** javafx.stage.Stage;

**public** **class** SliderExample **extends** Application {

@Override

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

//Drawing a Circle

Circle circle = **new** Circle(20);

circle.setFill(Color.***YELLOW***);

circle.setStroke(Color.***BLUEVIOLET***);

//Setting the slider, min=0, max=500, default=0

Slider slider = **new** Slider(0, 500, 0);

slider.setShowTickLabels(**true**);

slider.setShowTickMarks(**true**);

slider.setMajorTickUnit(100);

slider.setBlockIncrement(50);

//Setting the width of the slider

slider.setMaxWidth(300);

//Creating the translation transformation

Translate translate = **new** Translate();

//Linking the transformation to the slider

slider.valueProperty().addListener(**new** ChangeListener<Number>() {

**public** **void** changed(ObservableValue <?**extends** Number>observable, Number oldValue, Number newValue){

translate.setX((**double**) newValue);

translate.setY(50);

translate.setZ(100);

}

});

//Adding the transformation to the circle

circle.getTransforms().add(translate);

//VBox to arrange circle and the slider

VBox vbox = **new** VBox(85);

vbox.setPadding(**new** Insets(10));

vbox.getChildren().addAll(circle, slider);

vbox.setStyle("-fx-background-color: BEIGE");

//Preparing the scene

Scene scene = **new** Scene(vbox, 400, 250);

stage.setTitle("Slider Example");

stage.setScene(scene);

stage.show();

}

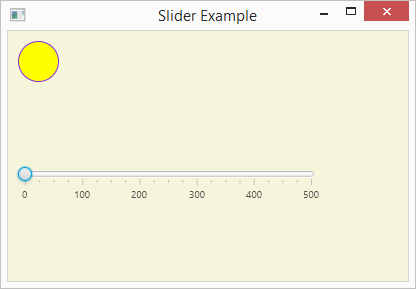
**public** **static** **void** main(String args[]){

*launch*(args);

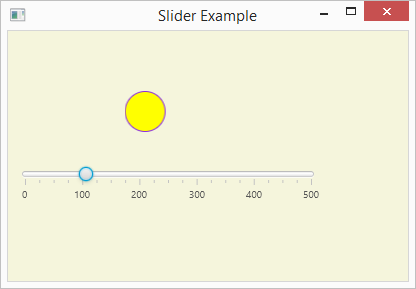
}

}

Run the program and you will get the following output:



Move the slider to the right:



### Description

In the code circle.getTransforms().add(translate); the circle object is calling getTransfors() method which uses the add() method.

A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate (tx, ty) to the original coordinate (X, Y) to get the new coordinate (X’, Y’).

For more details on the class Translate, go to: [Translate (JavaFX 8](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/transform/Translate.html)).