JavaFx

MVC in JavaFx:

* Driver:
  + Extends Application
  + Creates an instance of Controller
  + Gets final scene from controller and displays it

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

**Controller controller = new Controller();**

**Scene scene = new Scene(controller.myView.getPane(),WIDTH,HEIGHT);**

**primaryStage.setTitle("Title");**

**primaryStage.setScene(scene);**

**primaryStage.show();**

**}**

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

**launch(args);**

**}**

* Controller:
  + Holds instances of Models and Views
  + Creates handlers for a View’s nodes and links them to functionalities from Models
  + Returns the final scene to the driver from the view
* Model:
  + Holds all functionalities
  + Has no contact with the view
  + Creates no GUI objects or nodes
* View:
  + Creates all GUI and UI objects
  + Instantiates the scene

Base Objects:

Note: scenes can only hold Panes, UI, and Groups

**getChildren().add(node);**

**getChildren().add(node);**

* Panes: all have:
  + StackPane: Stacked above eachother in the order they were added in the center
  + Flowpane: Row-by-row (default) or column-by-column adjusting base on item width/height to go to a new line

**Pos includes: CENTER, CENTER\_LEFT, CENTER\_RIGHT**

**BOTTOM\_CENTER, BOTTOM\_LEFT, BOTTOM\_RIGHT,**

**TOP\_CENTER, TOP\_LEFT, TOP\_RIGHT**

**flowpane.setOrientation(Orientation.VERTICAL);**

**flowpane.setHgap(10);**

**flowpane.setVgap(10);**

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

**stackPane.setPadding(new Insets(TOP,RIGHT,BOT,LEFT));**

* + BorderPane: Based on region

**borderPane.setLeft(node);**

**borderPane.setRight(node);**

**borderPane.setCenter(node);**

**borderPane.setTop(node);**

**borderPane.setBottom(node);**

* + GridPane: Places nodes in cells in a 2D grid

**gridPane.add(node, i, j, x-span, y-span);**

**gridPane.setHgap(10);**

**gridPane.setVgap(10);**

* + HBox: Single row
  + VBox: Single column

**node.setText("Hello");**

**node.getText();**

* UI: all have:
  + Label:

**Label lb = new Label("Hello");**

* + Button:

**Button but1 = new Button("label");**

* + TextField / TextArea:

**Text.setPrefColumnCount(10);**

**Text.setPrefRowCount(5);**

* + Checkbox:

**CheckBox cb = new CheckBox("Hello");**

**boolean b = cb.isSelected();**

* + RadioButton:

**RadioButton rb = new RadioButton("Hello");**

**ToggleGroup tg = new ToggleGroup();**

**rb.setToggleGroup(tg);**

* Binding: changes to B are instantly applied to A

**A.textProperty().bind(B.textProperty());**

**Text text = new Text(20, 20, "some text");**

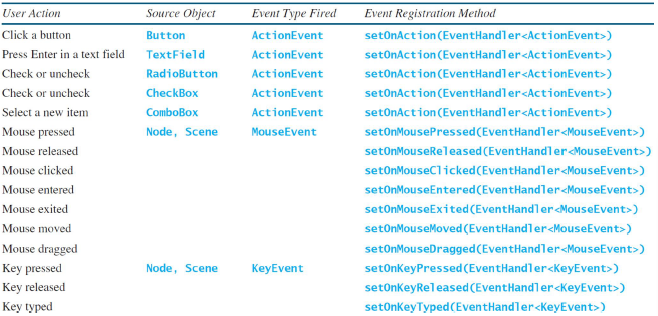
* Text
* Shapes:

**Line line = new Line(x1, y1, x2, y2);**

**Rectangle rect = new Rectangle(x, y, w, h);**

**Circle circle = new Circle(x, y, rad);**

* Event Handling:



Handling with a handler class:

**class HandlerClass implements EventHandler<ActionEvent> {**

**@Override**

**public void handle(ActionEvent e) {**

**if(e.getSource().equals(globalNode)){**

**//different event based on what e is**

**}**

**}**

**}**

**//in code:**

**globalNode.setOnAction(new HandlerClass() );**

Hanling instantly:

**btEnlarge.setOnAction( e -> {**

**// only works in Java 8**

**});**

**btEnlarge.setOnAction(**

**new EventHandler<ActionEvent>() {**

**@Override**

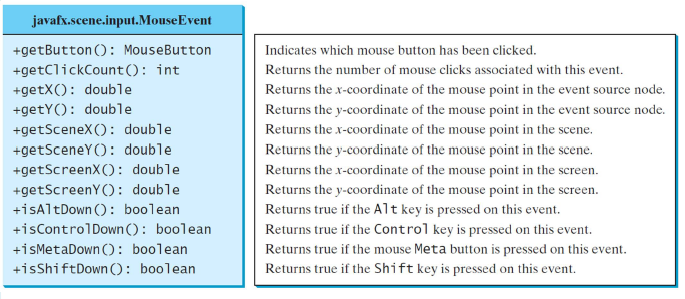
**public void handle(ActionEvent e) {**

**// Code for processing event e**

**}**

**}**

**});**



Mouse Events: