**JavaFX**

JavaFX is a software platform for creating desktop applications (GUI), as well as rich web applications (UI) that can run across a wide variety of devices.

**Stage:** Stage is window/interface where we can insert anything we want. Aise samjh Stage ek ghar h jisme jo krna h karo.

**Scene:** A scene in JavaFX is what you put inside a stage. It represents the content that you want to display within a particular window/stage.

**Node:** Choti choti chize jo present hota h ek scene me. For eg, button,text,img,etx

Note: In JavaFX, always class where main method is present extends the class **Application**.

Application class has 1 abstract method called **start** so we need to override it.

launch(args);

In the main method we call **launch** method which belongs to Application class which takes args and behind the scenes **start** member is called.

public void start(Stage stage) throws Exception { // start method definiton. Here Stage named stage is already passed to the method.

**Codes to peform basic actions on the stage**

package com.example.hellofx;  
// all neccessary imports  
public class HelloApplication extends Application {  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
 @Override   
 public void start(Stage stage) throws Exception { // start method definiton  
 // Stage st = new Stage(); // creating stage (There is no need to create a stage since it is already passed to the member).

// Creating root node  
 Group root = new Group();  
  
 // create a scene (We must pass a node to the scene, baaki color is optional)  
 Scene scene = new Scene(root, Color.*BLUE*); // Color.Black => set the bg of the scene to blue

// or  
// Scene scene = new Scene(root, 420,420, Color.BLUE); // passed node,set the size of the stage(420,420) and the bg color

// Setting the title (naam jo top left me display hota h).  
 stage.setTitle("Demo stage");  
  
 // Title ke left me icon ke liye (Make sure the image is present inside the src/main/resources in intelij else mention the complete path of the image.   
 Image icon = new Image("mikasa.png");  
 stage.getIcons().add(icon);  
  
 // setting height and width of the stage.  
 stage.setHeight(420);  
 stage.setWidth(420);  
  
 stage.setResizable(false); // this way we can't resize the stage.  
  
 // Setting where the stage will appear on own screen (Usually stage is set in the center of the center of the screen in fx)  
 stage.setX(50); // passing values according to our need.  
 stage.setY(50);  
  
 // stage to occupy the full screen  
 stage.setFullScreen(true);  
 stage.setFullScreenExitHint("Press q to escape full screen mode");// Full screen hone ke baad centre me ye dikhega (for 2-3 secs).  
 stage.setFullScreenExitKeyCombination(KeyCombination.*valueOf*("q")); // If tapped q in fullscreen mode full screen se exit.  
 stage.setScene(scene); // Here we are saying stage please whatever we have done in the scene.  
  
 // displaying the stage.  
 stage.show();  
 }  
}

**Displaying text and Lines**

// Inside start member

Group root = new Group();  
Scene scene = new Scene(root, 420,420, Color.*BLUE*);   
stage.setTitle("Demo stage");  
  
// Displaying text on the scene   
Text text = new Text();  
text.setText("Hey");  
text.setX(50); // co-ordinate shifting of the text  
text.setY(50);  
text.setFont(Font.*font*("Verdana", 50)); // set font and size (size is optional)  
text.setFill(Color.*LIMEGREEN*); // color of the text  
  
// Displaying line  
Line line = new Line(); // by default a horizontal line  
line.setStartX(200); // x axis me kaha start hoga  
line.setStartY(200);// y axis me kaha start hoga  
line.setEndX(500); // x axis me kaha end hoga  
line.setEndY(200); // y axis me kaha end hoga

// we can pass these 4 as paramter to the Constructor.

// Line line = new Line(200,200,500,200); // horizontal line //

line.setStrokeWidth(5);  
line.setStroke(Color.*RED*);  
line.setOpacity(0.5);  
line.setRotate(45);  
  
root.getChildren().add(text); // make sure to do this for every node  
root.getChildren().add(line);  
  
  
stage.setScene(scene);  
stage.show();

Refer: <https://www.youtube.com/watch?v=7nlU3_kEjTE&list=PLZPZq0r_RZOM-8vJA3NQFZB7JroDcMwev&index=5&t=459s> To display various shapes like rectangle triangle and circle

Also watch tutorial for a scene builder.

**Note: The class where all the logical work is done is called Controller class (naam khuch bhi de sakte h)**

**Switching between Scenes (below done with 2 scenes only)**

// Inside Controller class named SwitchSceneController

package com.example.hellofx;  
// all neccessary imports  
public class SwitchSceneController {  
 private Stage stage;  
 private Scene scene;  
 private Parent root;  
  
 public void switchToScene1(ActionEvent event) throws IOException { // Parameter ActionEvent says

// 2 ways to load the fxml file during event handling  
// 1.  
// root = FXMLLoader.load(getClass().getResource("Scene1.fxml")); // Here Scene1 is the name of the fxml file. Basicaly we are switcing to another using event handing  
// 2.   
 FXMLLoader loader = new FXMLLoader(getClass().getResource("Scene1.fxml"));  
 root = loader.load();

stage = (Stage)((Node)event.getSource()).getScene().getWindow();  
 scene = new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public void switchToScene2(ActionEvent event){  
 try {  
 root = FXMLLoader.*load*(getClass().getResource("Scene2.fxml")); // Here Scene2 is the name of the fxml file. Basicaly we are switcing to another using event handing  
 stage = (Stage) ((Node) event.getSource()).getScene().getWindow();  
 scene = new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
 } catch(Exception e){  
 System.*out*.println(e);  
 }  
 }  
  
}

Make sure in Scene Builder, Drag and Drop everything you need in the desired fxml file (here Scene1 and Scene2) and in “Add Action” make sure to write the name of the function (Here switchToScene1 or switchToScene2) that you want to activate after the click of the specified node.

Nahi samjha? Read Below

**For ex, upar Scene1.fxml me apn ne ek button drag and drop kiya and “Add Action” me switchToScene2 function mention kiya. Jis karan click krte hi button, switchToScene2 function invoke ho gaya and usne apna kaam kiya (Here Scene change kiya). Same with Scene2.fxml**

**Note: In Scene Builder Left center ke thoda niche controller likha hoga vaha click, Controller class dikhega vaha write the complete path of the Controller Class. For eg, Scene1.fxml me controller class me don’t write bss SwitchSceneController only, write packagename.packagename. ControllClassName (Here com.example.hellofx.SwitchSceneController)**

**Communcation Between Controllers (ek controller class se dusre controller class ko access krna)**

public class Controller1{  
 private Stage stage;  
 private Scene scene;  
 private Parent root;  
  
 public void login(ActionEvent event) {   
 try  
 {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("CommuncationBetControllersScene2.fxml")); Mentioned which fxml file to get  
 root = loader.load();  
  
 Controller2 scene2Controller = loader.getController(); // Object creation of other Controller class Creation // Now we can access all the members of the Controller2 class.

stage = (Stage) ((Node) event.getSource()).getScene().getWindow();  
 scene = new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
 } catch (Exception e){  
 System.*out*.println(e);  
 }  
 }  
}

**Eg,**

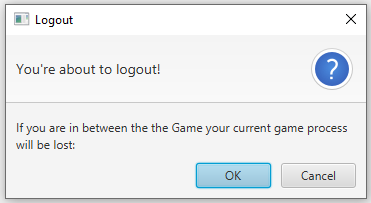
// First Controller Class

package com.example.hellofx;

// all neccessar imports   
public class CommunicationBetweenControllersController1 {  
 @FXML  
 TextField nameTextField; // object name must be same as fx:id of the TextField in the fxml file (make sure to write the object name in the fx:id of the TextField in the scene builder)  
  
 private Stage stage;  
 private Scene scene;  
 private Parent root;  
  
 public void login(ActionEvent event) { // A button labeled as login. Clicking which this function is invoked. This function changes the scene and also seeks help from another controller  
 try  
 {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("CommuncationBetControllersScene2.fxml")); // Mentioned which fxml file to get   
 root = loader.load();  
  
 String userName = nameTextField.getText();  
 CommunicationBetweenControllersController2 scene2Controller = loader.getController(); // Object creation of other Controller class Creation  
 scene2Controller.displayName(userName);  
  
 stage = (Stage) ((Node) event.getSource()).getScene().getWindow();  
 scene = new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
 } catch (Exception e){  
 System.*out*.println(e);  
 }  
 }  
}

// Second Controller Class

package com.example.hellofx;  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
  
public class CommunicationBetweenControllersController2 {  
 @FXML  
 Label nameLabel; // object name must be same as fx:id (make sure to write the object name in the fx:id of the Label in the scene builder)  
 public void displayName(String userName) {  
 nameLabel.setText("Hello " + userName);  
 }  
}

** Log Out Button / Action to perform if clicked red cross on rightmost top corner**

We can create a Alert screen while the users clicks at mentioned root and we can also create it if the user click the Close (X) button on the rightmost top corner. How it will look ->

If done using a node (Below is done using a node)

package com.example.hellofx;  
// All neccesaary imports  
public class LogOutOrExitController {  
 Stage stage;  
 public void logout(ActionEvent event){ // invoked when clicked the node  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*); // Creating Alert. (para says kis type ka alert chahiye. CONFIRMATION QuestionMark blue color me dega (upar dekh). Aur bhi alert type h).   
 alert.setTitle("Logout");  
 alert.setHeaderText("You're about to logout!");  
 alert.setContentText("If you are in between the the Game your current game process will be lost: ");  
  
 if(alert.showAndWait().get() == ButtonType.*OK*){ // agar OK tap kiya to  
 stage = (Stage)((Node)event.getSource()).getScene().getWindow();  
  
 System.*out*.println("You successfully logged out!");  
 stage.close();  
 }  
 }  
}

If clicked Close (X) button

// Inside the class/java file where the execution of the porgram starts package com.example.hellofx;  
// All neccesaary imports   
public class LogOutOrExit extends Application {  
 @Override  
 public void start(Stage stage) {  
 try {  
 Parent root = FXMLLoader.*load*(getClass().getResource("LogOutScene.fxml")); // loading the fxml file  
  
 stage.setOnCloseRequest(event -> {  
 event.consume(); // If tapped cancel instead of OK don't close the window. Ye nhi likha to cancel tap krne pe hi window close ho jayega.  
 logOut(stage); // Calling logOut method which will display prompt Alert Window  
 });

Scene scene = new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
 public void logOut(Stage stage){ // This method can be named anything   
  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*);  
 alert.setTitle("Logout");  
 alert.setHeaderText("You're about to logout!");  
 alert.setContentText("Do you want to save before exiting?");  
  
 if (alert.showAndWait().get() == ButtonType.*OK*){  
 System.*out*.println("You successfully logged out");  
 stage.close();  
 }  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
 }

**ImageView**

In Scene builder, In library section search ImageView draga and drop it the in the anchorpane. Click the ImageView, in properties on image section add the image.

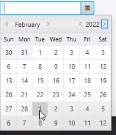
**Note: For Intelij all the Images must be present in the resources.** If the image is inside resources only write **“/img\_name.imageformat”** if within a folder inside resources write **“/folderName/img\_name.imgformat”.**.

In Code section of Scene builder set the fx:id (here I set it to imageView). Set the event handler (mention the name of the function here changeImage and changeImage1) according to desired action (here I chose MouseEvent). Set the controller class as usual and we are ready to go write code in controllor class

//Inside Controller Class

public class ImageViewController {  
  
 @FXML  
 private ImageView imageView; // same as fx:id of ImageView  
  
 Image image = new Image(getClass().getResourceAsStream("/image/mikasa.png"));  
 Image image1 = new Image(getClass().getResourceAsStream("/image/mikasa\_kun.jpg"));

// CHaning image on mouse enter  
 @FXML  
 void changeImage(MouseEvent event) {  
 imageView.setImage(image);  
 }  
  
 // Displayig the same image when we GUI was first displayed  
 @FXML  
 void changeImage1(MouseEvent event) {  
 imageView.setImage(image1);  
 }



**DatePicker**(letting user pick a date)

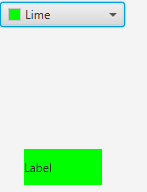
<https://www.youtube.com/watch?v=3Ht-JMQh2JI&list=PLZPZq0r_RZOM-8vJA3NQFZB7JroDcMwev&index=15>

**RadioButton**(Mcq type, can only choose one option)

<https://www.youtube.com/watch?v=9u5D1wNlW5M&list=PLZPZq0r_RZOM-8vJA3NQFZB7JroDcMwev&index=14>

**ColorPicker**

**In SceneBuilder:** In Library drag nd drop ColorPicker. Click dragged ColorPicker, in code set fx:id (for object creation), #onAction to method which we want to call (here changeColor).

****Below the code to change the Bg of label by the user using ColorPicker is mentioned mentioned. (GUI below)

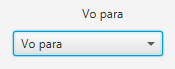
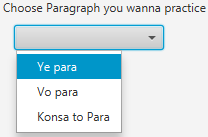
package com.example.hellofx;  
// All necessary imports   
public class ColorPickerController {  
  
 @FXML  
 private ColorPicker colorPicker; // obj name same as fx:id  
  
 @FXML  
 private Label label; // obj name same as fx:id   
  
 @FXML  
 void changeColor(ActionEvent event) { // functioned invoked when action performed  
 Color myColor = colorPicker.getValue();  
 label.setBackground(new Background(new BackgroundFill(myColor, CornerRadii.*EMPTY*, Insets.*EMPTY*)));  
 }  
}

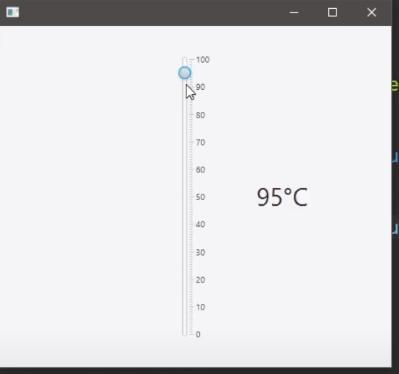
**ChoiceBox**(Dropdown menu)

**In SceneBuilder:** Library me drag and drop ChoiceBox, set it’s fx:id as you desire.

**Note: ChoiceBox do not have default dropdown values and #OnAction. We manually have to specify both. For this we implement Initializable Interface and we override initialize method and there we specify both (dropdown and #OnAction)**

package com.example.hellofx;  
 // All necessary imports  
 public class ChoiceBoxController implements Initializable {  
 @FXML  
 private Label label; // obj name same as fx:id   
 @FXML  
 private ChoiceBox<String> paraChoiceBox; // obj name same as fx:id  
 private String [] para = {"Ye para", "Vo para", "Konsa to Para"};  
  
 @Override  
 public void initialize(URL url, ResourceBundle resourceBundle) {  
 paraChoiceBox.getItems().addAll(para); // Dropdown added to the checkbox  
  
 // Setting OnAction for the desired object   
 paraChoiceBox.setOnAction(this::changeLabel);// :: method reference operator. Para says refer to this method for Action   
 }  
 private void changeLabel(ActionEvent event) {  
 String chosenPara = paraChoiceBox.getValue(); // storing selected dropdown option  
 label.setText(chosenPara);  
 }  
 }

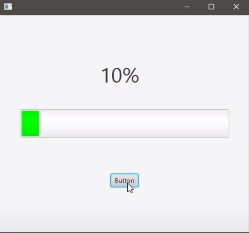
Output => => =>

**Slider**

package com.example.hellofx;

// All necessary imports  
public class SliderController implements Initializable{  
 @FXML  
 private Label myLabel; // label whose text will be decided depending on the slider value  
  
 @FXML  
 private Slider mySlider;  
  
 int myTemperature;  
  
 @Override  
 public void initialize(URL url, ResourceBundle resources) {

// We could done using code jaise Choice Box me kiya tha  
  
 myTemperature = (int) mySlider.getValue();  
 myLabel.setText(Integer.*toString*(myTemperature) + "°C");  
  
 mySlider.valueProperty().addListener(new ChangeListener<Number>() {  
  
 @Override  
 public void changed(ObservableValue<? extends Number> observable, Number oldNumber, Number newNumber) {  
  
 myTemperature = (int) mySlider.getValue();  
 myLabel.setText(Integer.*toString*(myTemperature) + "°C");  
  
 }  
 });  
 }  
}

**ProgessBar**

package com.example.hellofx;  
// All necessary imports  
public class ProgessBarController implements Initializable{  
  
 @FXML  
 private ProgressBar myProgressBar;  
  
 @FXML  
 private Button myButton;  
  
 @FXML  
 private Label myLabel;  
  
 //The BigDecimal class gives its user complete control over rounding behavior   
 BigDecimal progress = new BigDecimal(String.*format*("%.2f", 0.0));  
  
 @Override  
 public void initialize(URL arg0, ResourceBundle arg1) {  
 myProgressBar.setStyle("-fx-accent: #00FF00;"); // setting color of progress bar  
  
 }  
 public void increaseProgress() {  
  
 if(progress.doubleValue() < 1) {  
 progress = new BigDecimal(String.*format*("%.2f", progress.doubleValue() + 0.1));  
   
 myProgressBar.setProgress(progress.doubleValue());  
 myLabel.setText(Integer.*toString*((int)Math.*round*(progress.doubleValue() \* 100)) + "%");  
 }  
 }  
}

**Spinner**

Arrow up-down daba ke sequential order upar niche karna (for eg, 1-100 upar niche krna)

****[**https://www.youtube.com/watch?v=hSTEVJe4HSE&list=PLZPZq0r\_RZOM-8vJA3NQFZB7JroDcMwev&index=20**](https://www.youtube.com/watch?v=hSTEVJe4HSE&list=PLZPZq0r_RZOM-8vJA3NQFZB7JroDcMwev&index=20)