University of Nottingham
UK | CHINA | MALAYSIA

COMP2059
Developing Maintainable Software

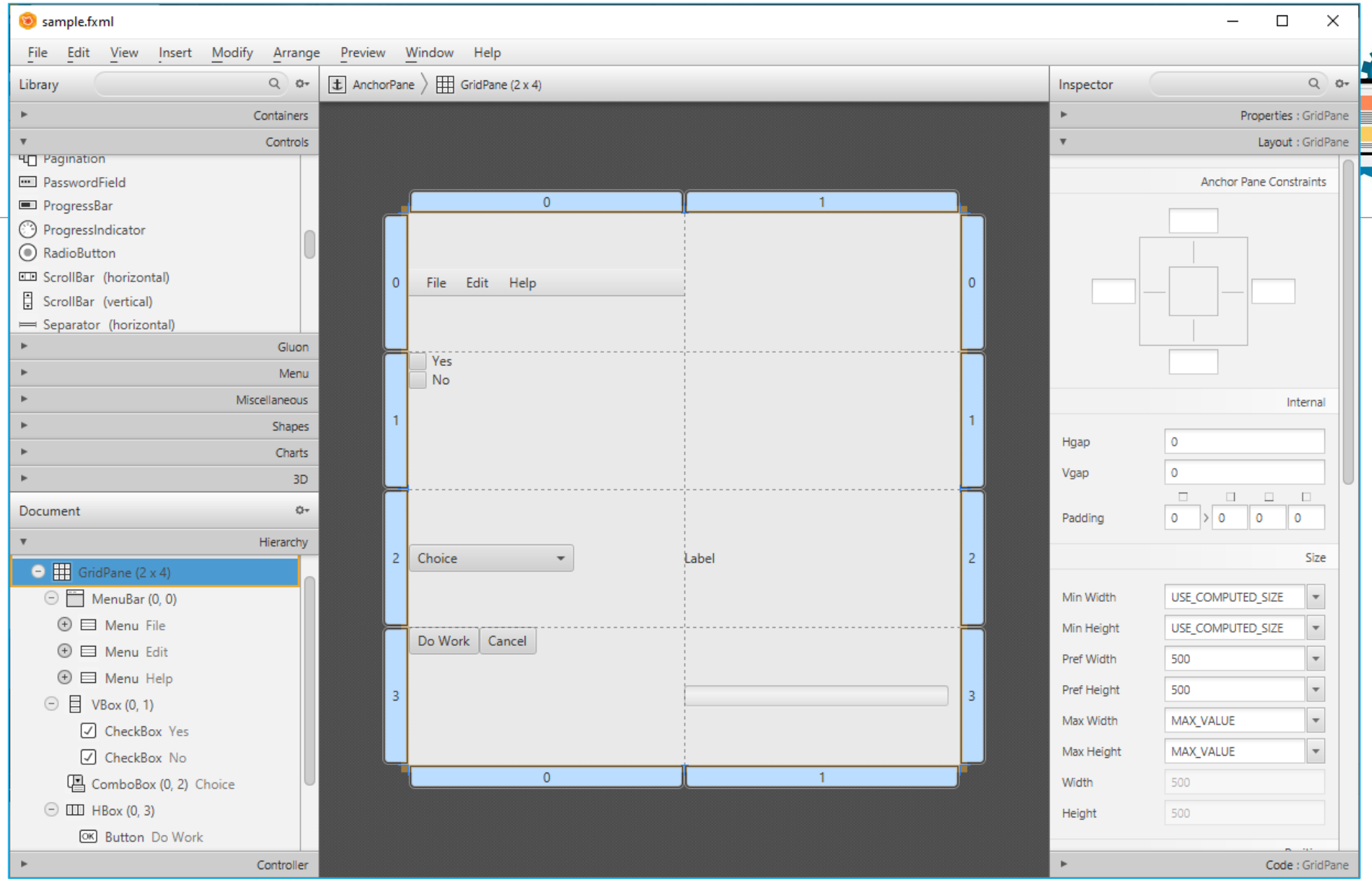
LECTURE 07 – MAINTAINABLE GUI DEVELOPMENT (2/2)

Boon Giin Lee (Bryan)

University of Nottingham
UK | CHINA | MALAYSIA

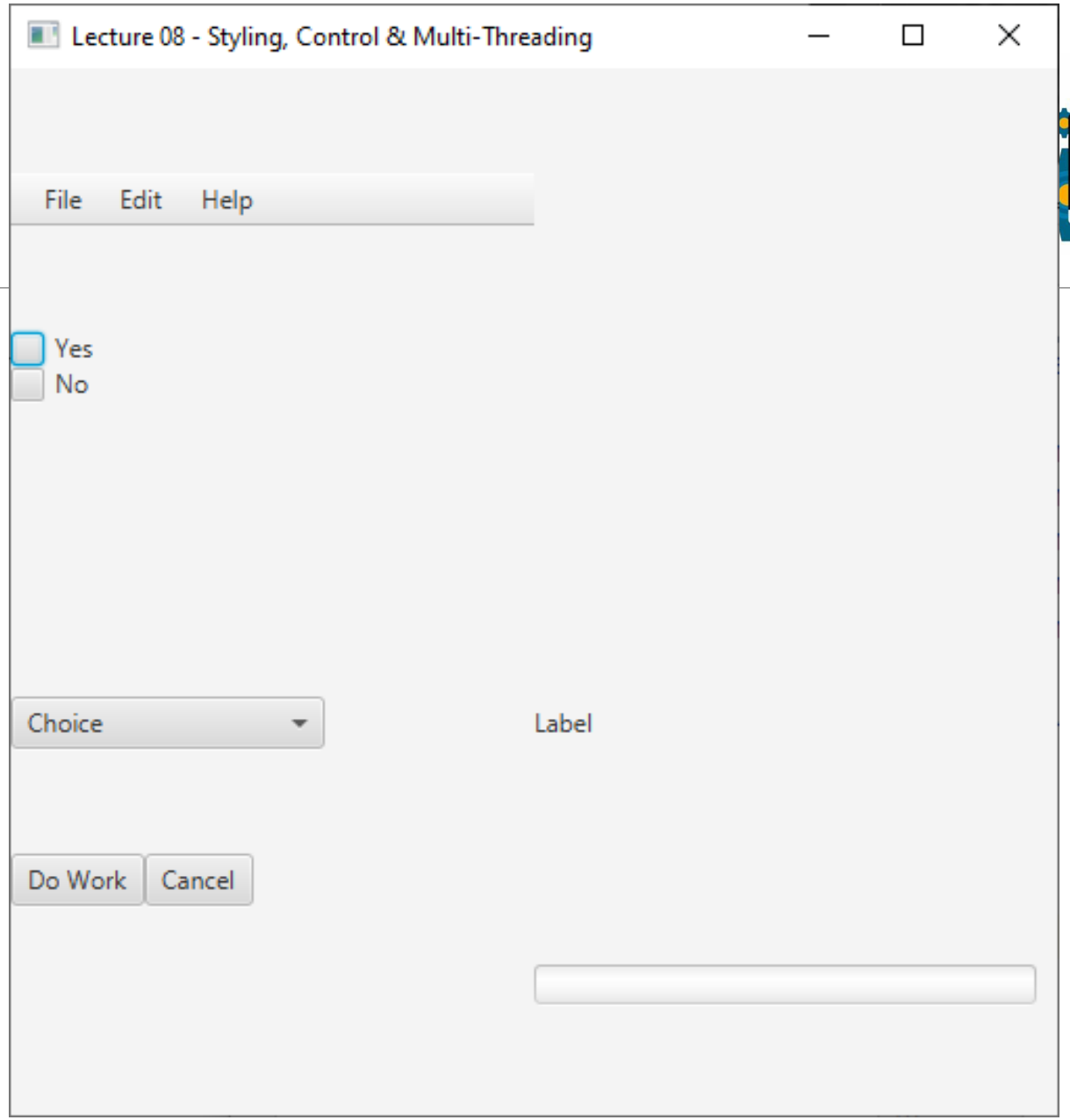
JavaFX Advanced Topics

PREPARATION


sample.fxml
File Edit View Insert Modify Arrange Preview Window Help
Library
Containers
Controls
Pagination
PasswordField
ProgressBar
ProgressIndicator
RadioButton
ScrollBar (horizontal)
ScrollBar (vertical)
Separator (horizontal)
Gluon
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
GridPane (2 x 4)
MenuBar (0, 0)
Menu File
Menu Edit
Menu Help
VBox (0, 1)
CheckBox Yes
CheckBox No
ComboBox (0, 2) Choice
HBox (0, 3)
Button Do Work
Controller
AnchorPane
GridPane (2 x 4)
File Edit Help
Yes
No
Choice
Label
Do Work
Cancel
Inspector
Properties : GridPane
Layout : GridPane
Anchor Pane Constraints
Internal
Hgap
Vgap
Padding
Size
Min Width
Min Height
Pref Width
Pref Height
Max Width
Max Height
Width
Height
USE_COMPUTED_SIZE
500
MAX_VALUE
Code : GridPane

Preparation

- The result.



University of Nottingham
UK | CHINA | MALAYSIA

JavaFX Advanced Topics

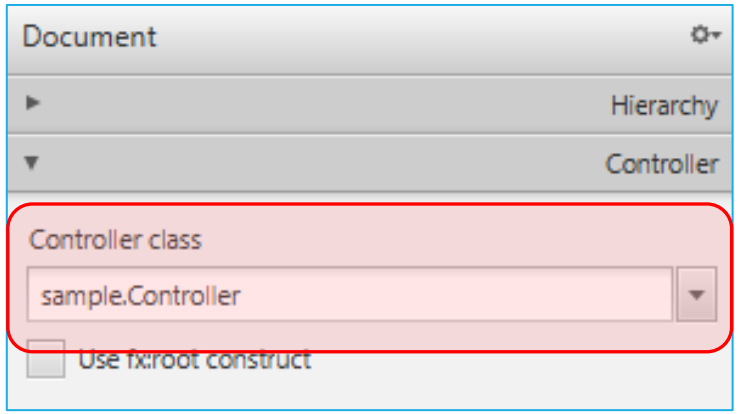
CONTROLS

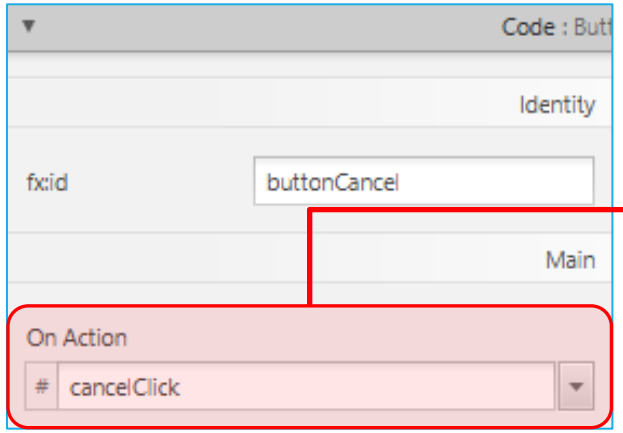
Work with FXML (Reminder)

- Define controller class in FXML file (if not done automatically)
- Write controller code.

This is basic FXML event handling.

- Tell button which method to call.






```
public class Controller {
    public void cancelClick(ActionEvent actionEvent) {
        System.exit( status: 0);
    }
}
```

Work with FXML

- Use "`fx:id`" in SceneBuilder to couple with a variable in `Controller` class.


```
public class PhoneControl {
    @FXML
    private ImageView ivPhoneImage;

    public void initialize() {
        Image image = new Image( s: "phone.png");
        ivPhoneImage.setImage(image);
    }
}
```

A Selection of Controls

- CheckBox
- ComboBox
- MenuBar
- ImageView

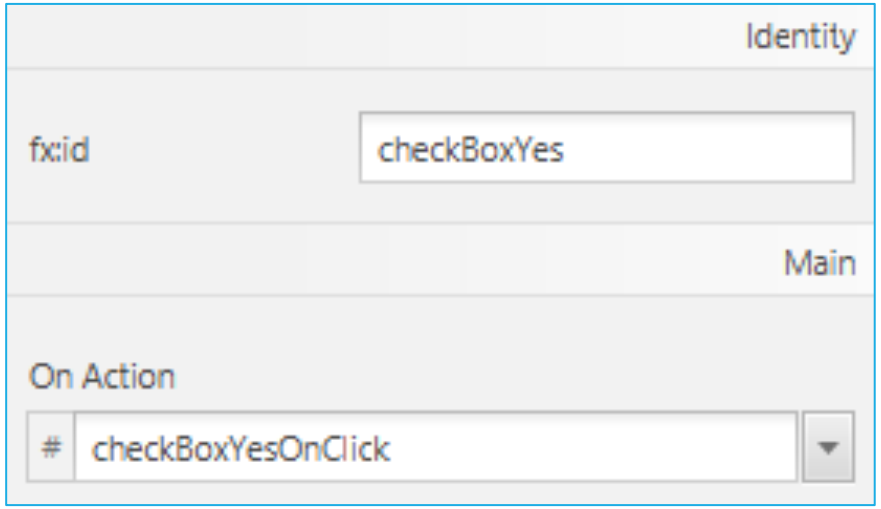
- NOTE: These are all `Node` subclasses.

The `initialize()` Method

- Why do we need the initialize() method?
  - The `FXMLLoader` creates an instance of the corresponding controller by invoking first the default constructor and then the `initialize()` method.
  - What is the difference between using the constructor of `initialize()` method to initialize the controller?
- The constructor is called first, then any `@FXML` annotated fields are populated, then the `initialize()` method is called.
  - This means the constructor does NOT have access to `@FXML` fields referring to components defined in the .fxml file, while `initialize()` does have access to them.

CheckBox

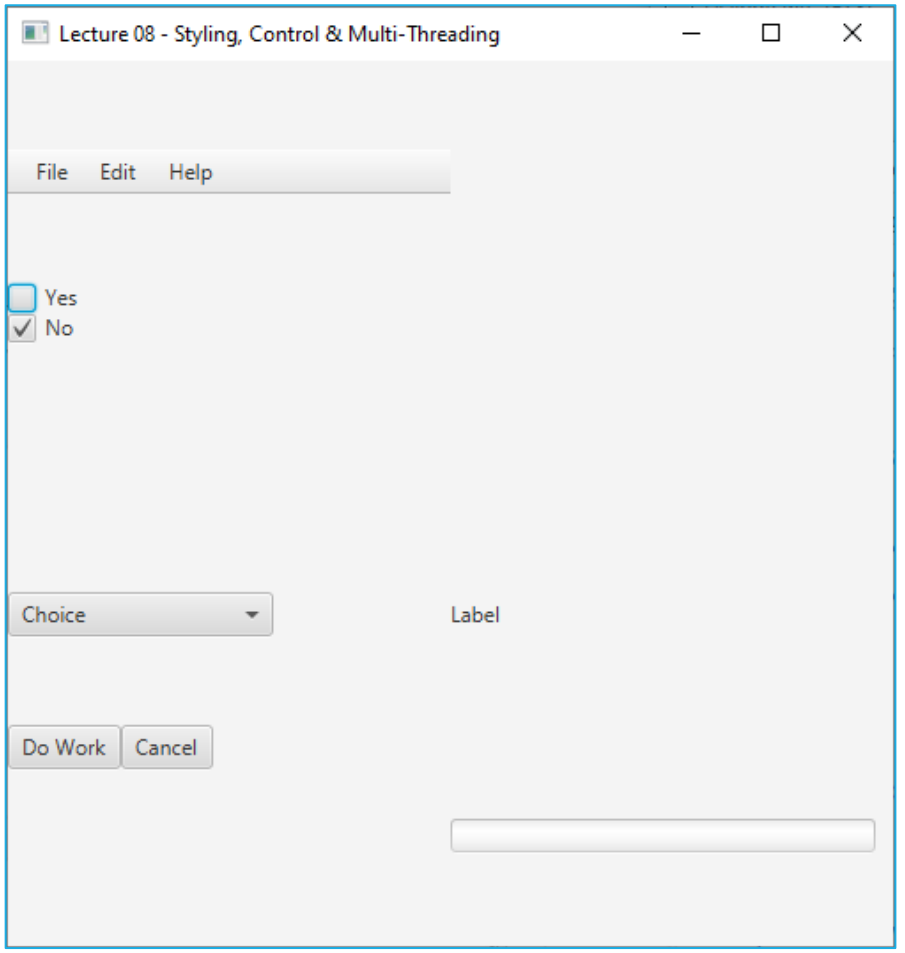
- Check boxes have a `boolean` state, depending whether checked or not.
- Steps.
  - Add "`checkBoxYesOnClick()`" method to the corresponding Controller class.
  - Give it an `fx:id` and listen for the action.



CheckBox

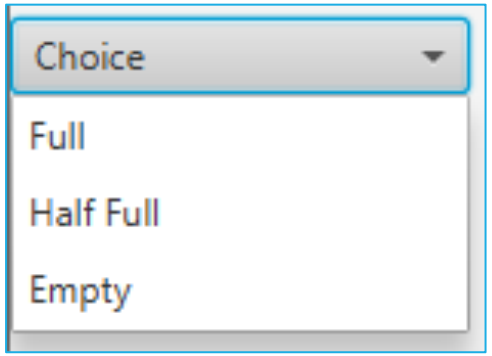
```
@FXML private CheckBox checkBoxYes;
@FXML private CheckBox checkBoxNo;

@FXML
private void initialize() {
    checkBoxYes.setSelected(true);
}

public void checkBoxYesOnClick(ActionEvent actionEvent) {
    if(checkBoxYes.isSelected())
        checkBoxNo.setSelected(false);
    else
        checkBoxNo.setSelected(true);
}
```

```
public void checkBoxYesOnClick(ActionEvent actionEvent) {
    checkBoxNo.setSelected(!checkBoxYes.isSelected());
}
```



ComboBox

- This is the “drop down box” style of control.
- It presents the user with a list of items, and the user can select one.
- Slightly different to other controls so far.
  - Need to get items into the list.
  - Need to check when an item had been selected.

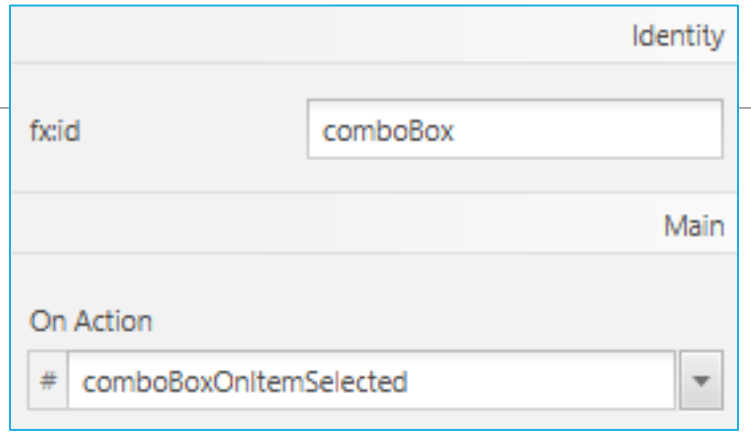


ComboBox

```
@FXML private CheckBox checkBoxYes;
@FXML private CheckBox checkBoxNo;
@FXML private ComboBox comboBox;

@FXML
private void initialize() {
    checkBoxYes.setSelected(true);

    ObservableList<String> comboBoxOptions =
            FXCollections.observableArrayList( ...es: "Full", "Half Full", "Empty");
}
```

Back to ComboBox

- How do you get selections?
  - Uses SceneBuilder and `@FXML`.
    - Choose `OnAction` event in SceneBuilder.
    - Then write a handler method.



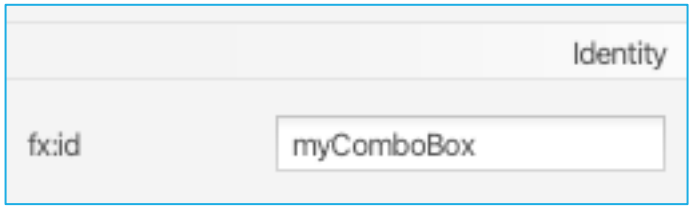

```
public void comboBoxOnItemSelected(ActionEvent actionEvent) {
    String s = (String) comboBox.getSelectionModel().getSelectedItem();
    System.out.println("Selected choice is \"" + s + "\"");
}
```

    - Can also do it the other way around …

ComboBox

- How do you get selections?
  - Coding directly.
    - Using an `EventHandler<>`.

```
comboBox.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent actionEvent) {
        String s = (String) comboBox.getSelectionModel().getSelectedItem();
        System.out.println("Selected choice is \"" + s + "\"");
    }
});
```



    - Using a lambda expression.

```
comboBox.setOnAction(event -> {
    String s = (String) comboBox.getSelectionModel().getSelectedItem();
    System.out.println("Selected choice is \"" + s + "\"");
});
```

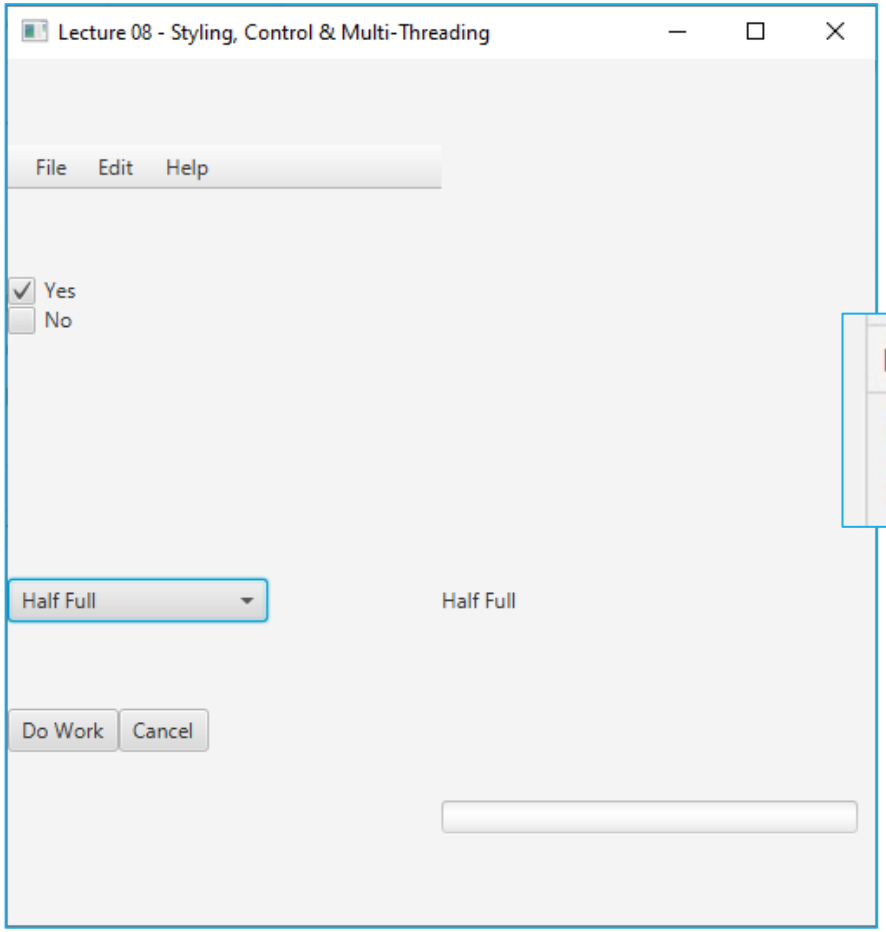
ComboBox

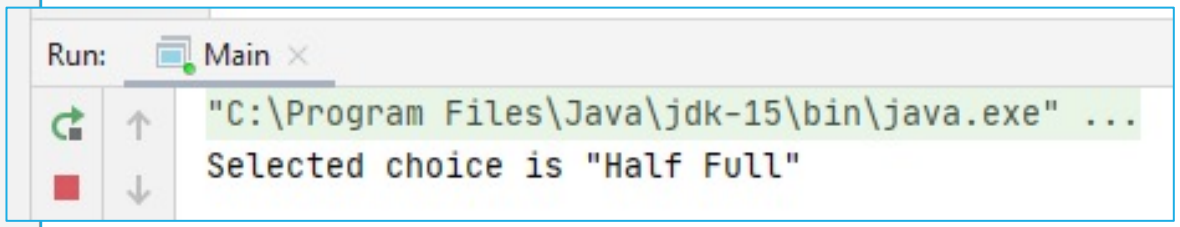
- There is another way to get the choice from a combo box, and in fact to link other kinds of controls with variables: Binding.

```
@FXML private ComboBox<String> comboBox;
@FXML private Label label;

@FXML
private void initialize() {
    checkBoxYes.setSelected(true);

    ObservableList<String> comboBoxOptions =
            FXCollections.observableArrayList( ...es: "Full", "Half Full", "Empty");
    comboBox.setItems(comboBoxOptions);

    label.textProperty().bind(comboBox.getSelectionModel().selectedItemProperty());
}
```

- Whenever the value of the ComboBox property changes, the Label text changes automatically.

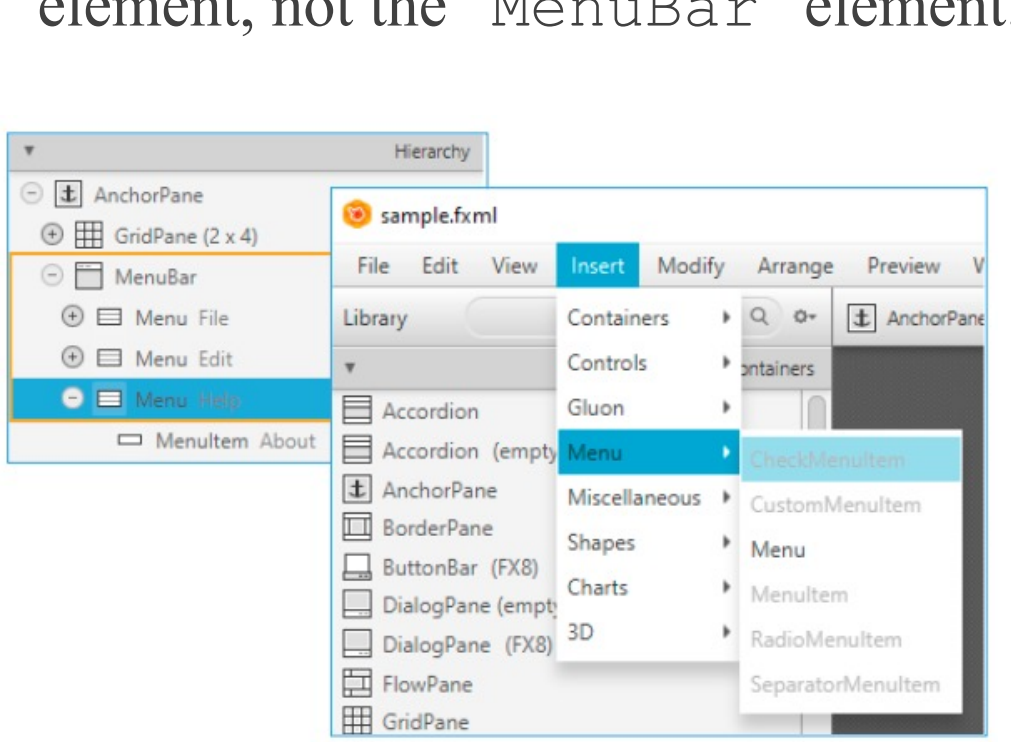
ComboBox

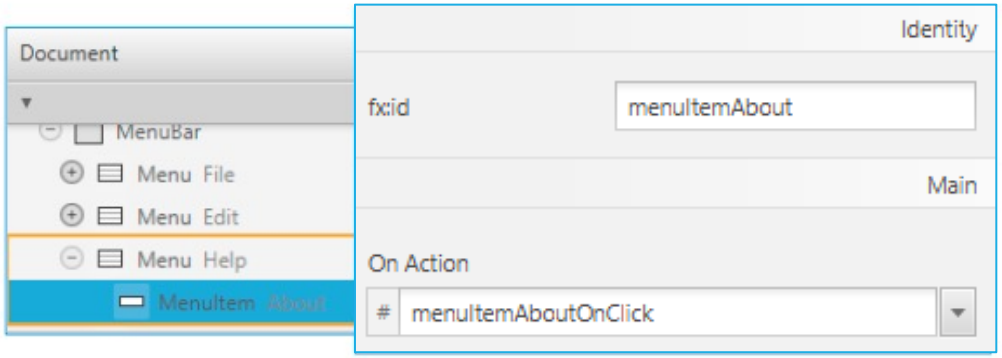

Lecture 08 - Styling, Control & Multi-Threading
File
Edit
Help
Yes
No
Half Full
Half Full
Do Work
Cancel


Run:
Main
"C:\Program Files\Java\jdk-15\bin\java.exe" ...
Selected choice is "Half Full"

MenuBar

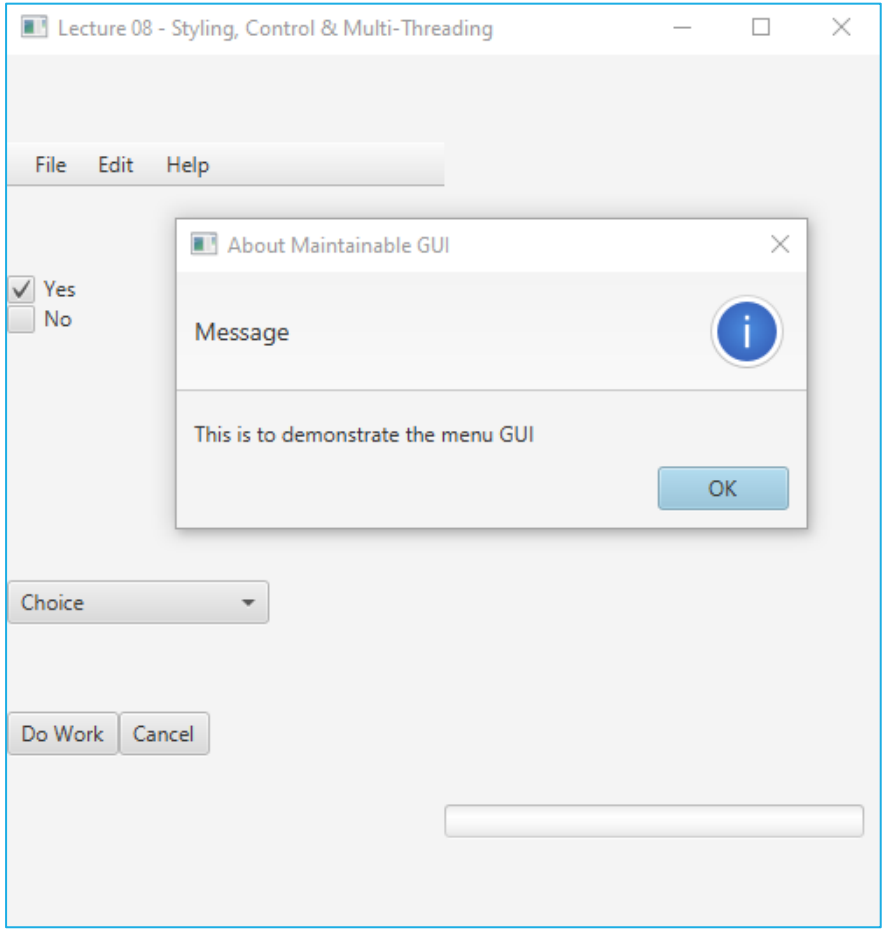
- To add another "`Menu`' element to the "`MenuBar`', need to be on a "`Menu`" element, not the "`MenuBar`" element.



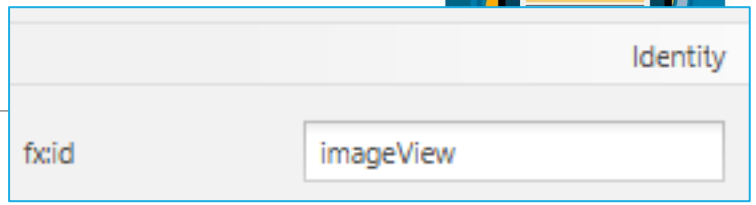



```
public void menuItemAboutOnClick(ActionEvent actionEvent) {
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("About Maintainable GUI");
    alert.setContentText("This is to demonstrate the menu GUI");
    alert.show();
}
```

MenuBar


Lecture 08 - Styling, Control & Multi-Threading
File
Edit
Help
About Maintainable GUI
Yes
No
Message
This is to demonstrate the menu GUI
OK
Choice
Do Work
Cancel

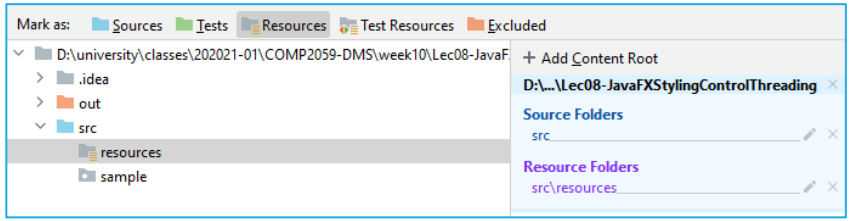
Add Images



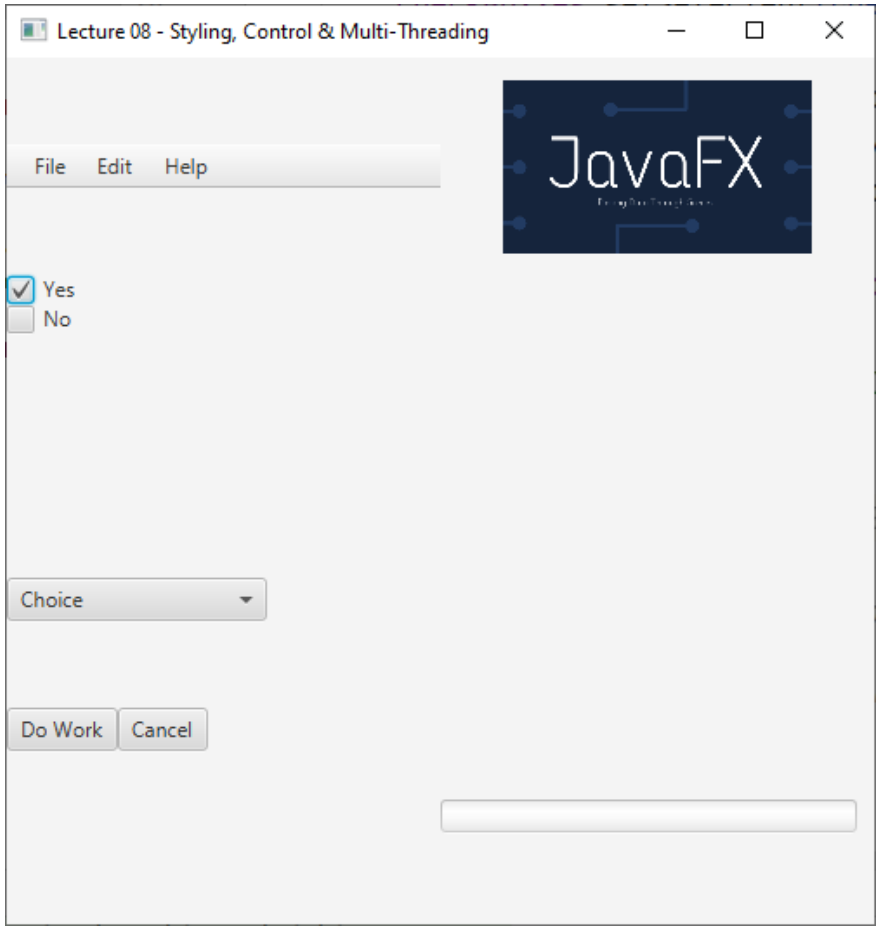
- Can be done with `ImageView` control.
  - Give it an `fx:id`.
  - In the code:

```
@FXML
private void initialize() {
    checkBoxYes.setSelected(true);

    ObservableList<String> comboBoxOptions =
            FXCollections.observableArrayList( ...es: "Full", "Half Full", "Empty");
    comboBox.setItems(comboBoxOptions);

    label.textProperty().bind(comboBox.getSelectionModel().selectedItemProperty());

    Image image = new Image( s: "javafx.png");
    imageView.setImage(image);
```

- Watch for working directories when loading files.



Add Images


Lecture 08 - Styling, Control & Multi-Threading
File
Edit
Help
JavaFX
Yes
No
Choice
Do Work
Cancel

University of Nottingham
UK | CHINA | MALAYSIA

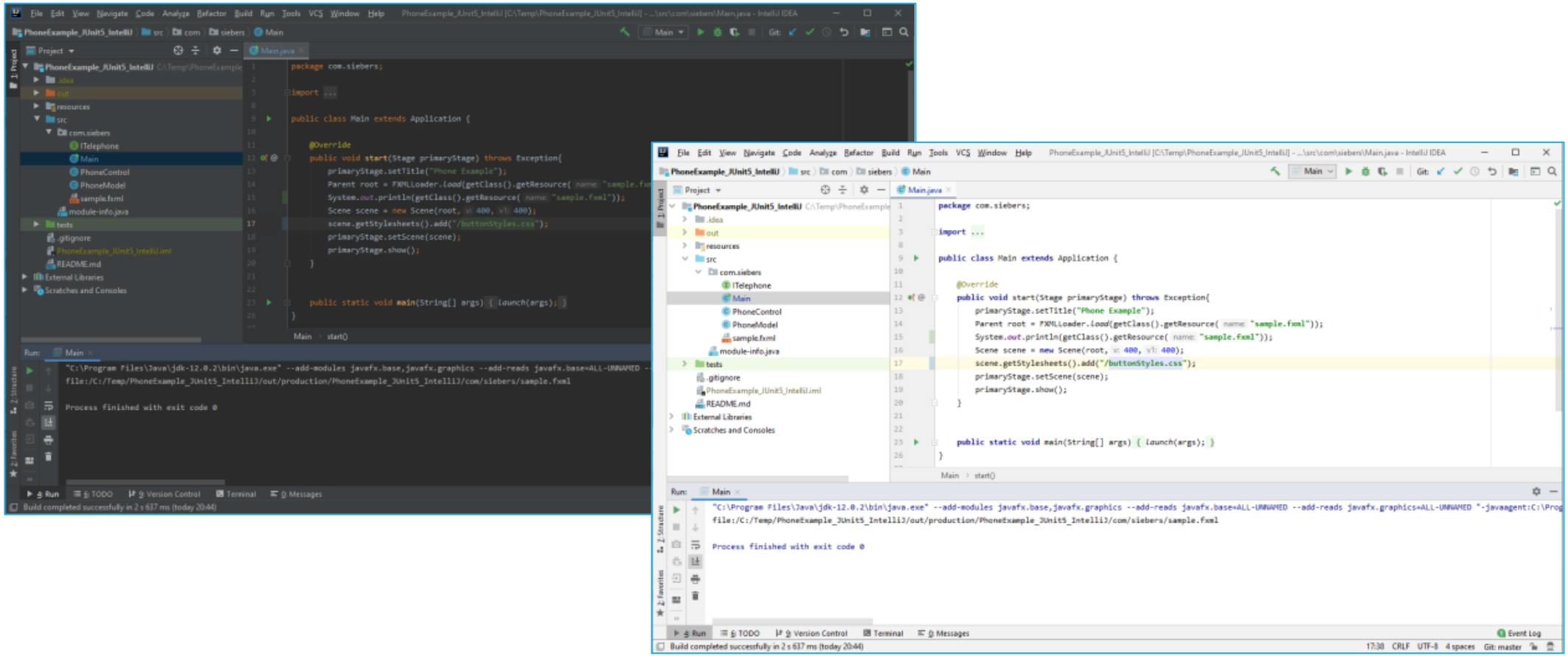
JavaFX Advanced Topics

STYLING

Customising GUI Appearance

- In addition to laying out the elements on the screen, also want to specify how they look.
- Remember appearance, design and code should be kept separated.
- Here, will change the appearance without adjusting anything else.
  - Suppose the company acquires a software package.
  - First thing might want to do is to bring the appearance in line with the rest of the product range.

Applications with Distinctive Themes

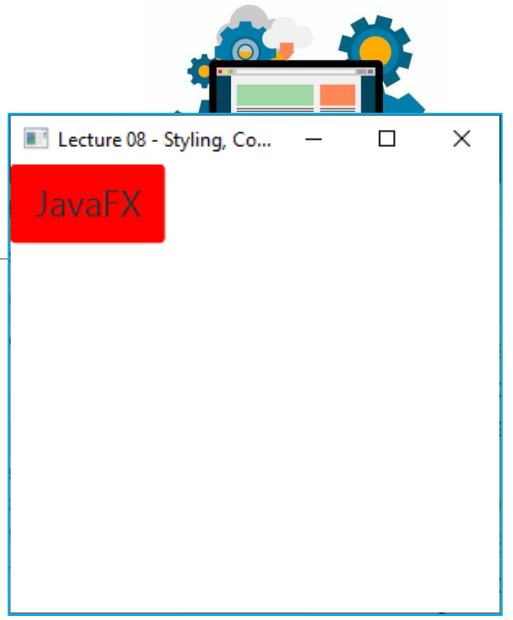


Cascading Style Sheet

- A Cascading Style Sheet (CSS) is a means to define the look (visual properties) of UI elements in a GUI application.
  - Primary developed for use in web pages.
  - Allows separation of presentation and content/behaviour.
- CSS provides the syntax to define rules.
  - A rule consists of selector and property/value pair(s).

```
.button {
    -fx-background-color: red;
    -fx-text-size: 22;
}
```

User Defined Style Sheets



- Create *.css style sheet in “resources” folder.
  - Remember to add the resources folder to the “build-path”.
  - JavaFX CSS naming conventions.
    - All style class names are lowercase node names.
    - If node name consists of multiple words, style class names use hyphen.
    - Property names start with “`-fx-`” followed by instance variables (lowercase + hyphenated).

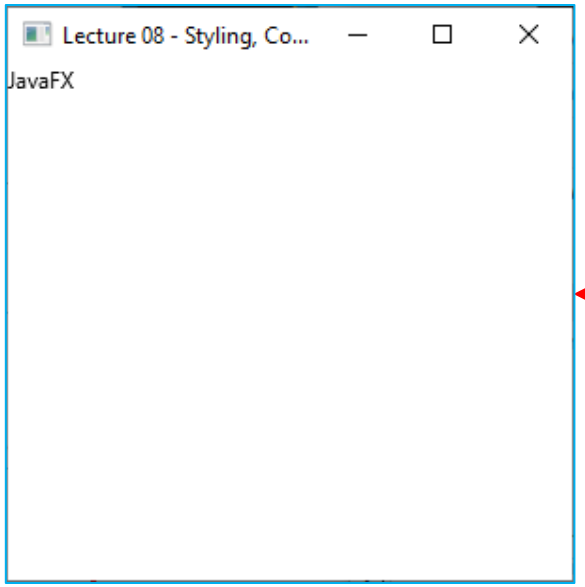
```
.button {
    -fx-background-color: red;
    -fx-font-size: 22;
}
```

```
Group root = new Group();
Scene scene = new Scene(root, v: 300, v1: 275);
scene.getStylesheets().add(getClass().getResource( name: "styles.css").toString());

Button button = new Button( s: "JavaFX");
root.getChildren().add(button);
```

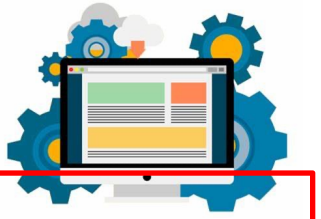
Default Style Sheets

- Default look for JavaFX application is "`modena.css`".

```
Scene scene = new Scene(root, v: 300, v1: 275);
Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);
// scene.getStylesheets().add(getClass().getResource("styles.css").toString());
```



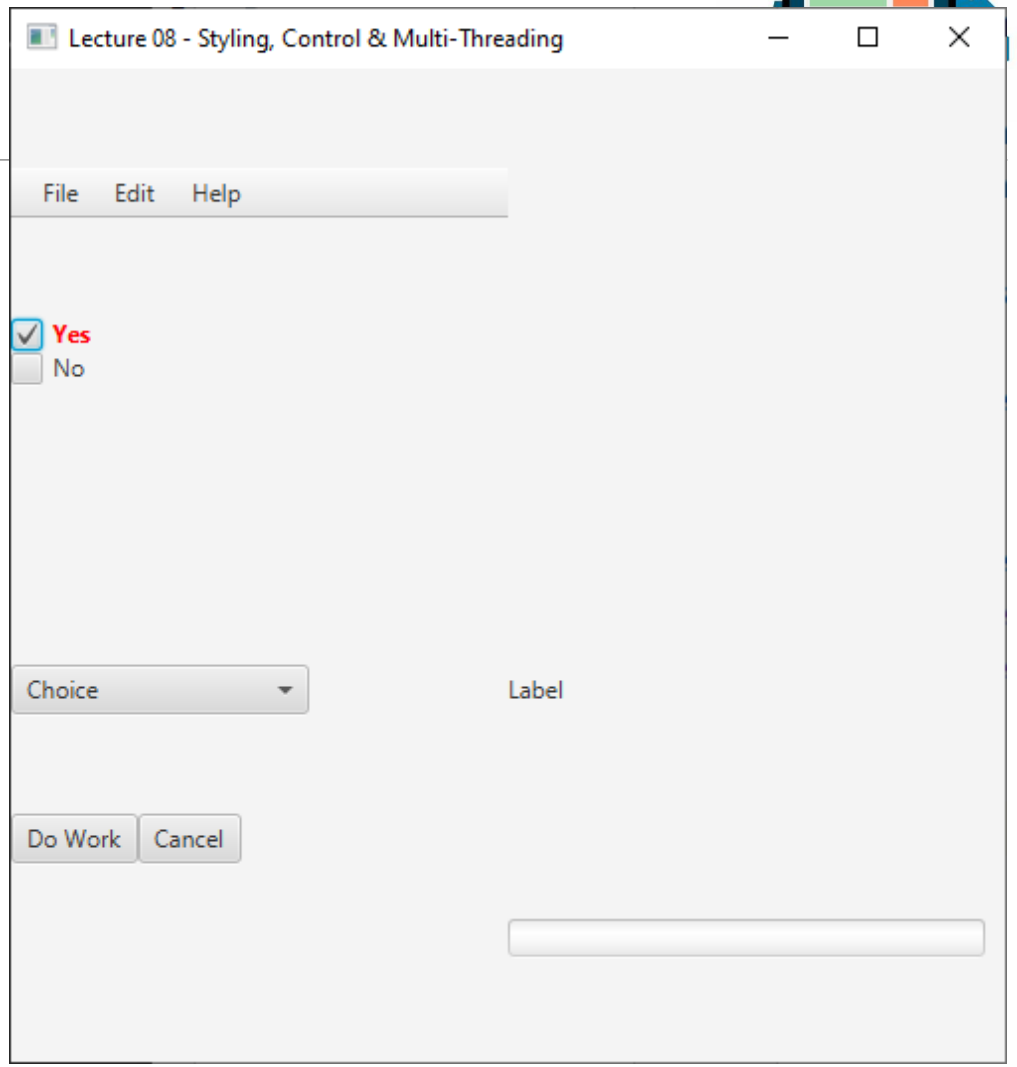
Adding Inline Styles



- Can add styles to individual nodes.
  - Use: `setStyle(String inlineStyle)` to set the `style` and `getStyle()` to get the style of a node.

```
public class Controller {
    @FXML
    private CheckBox checkBoxYes;

    @FXML
    private void initialize() {
        checkBoxYes.setSelected(true);
        checkBoxYes.setStyle("-fx-text-fill: red; -fx-font-weight: bold");
    }
}
```

Adding Inline Styles



Priorities of Styles

- Ranking order (highest to lowest).
  - Inline style.
  - Parent style sheet (parent class).
  - Scene style sheet. (CSS)
  - Values set in the code using JavaFX API (e.g. using "`setFont()`").
  - User agent style sheet (JavaFX runtime).

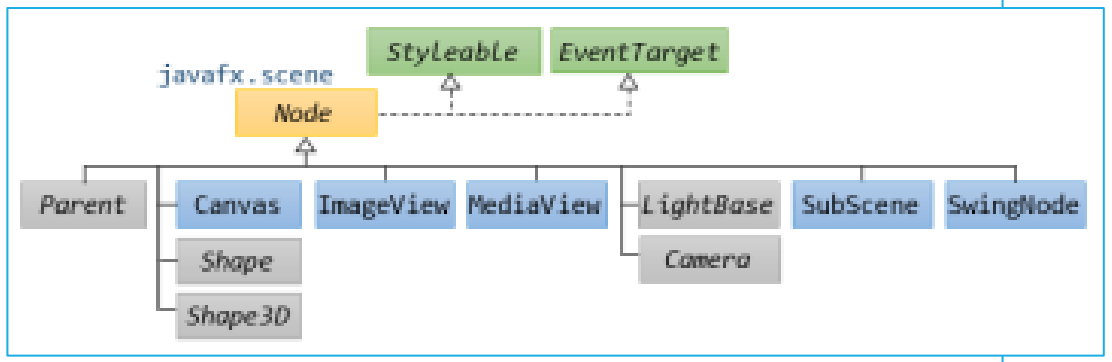
Inherit CSS Properties

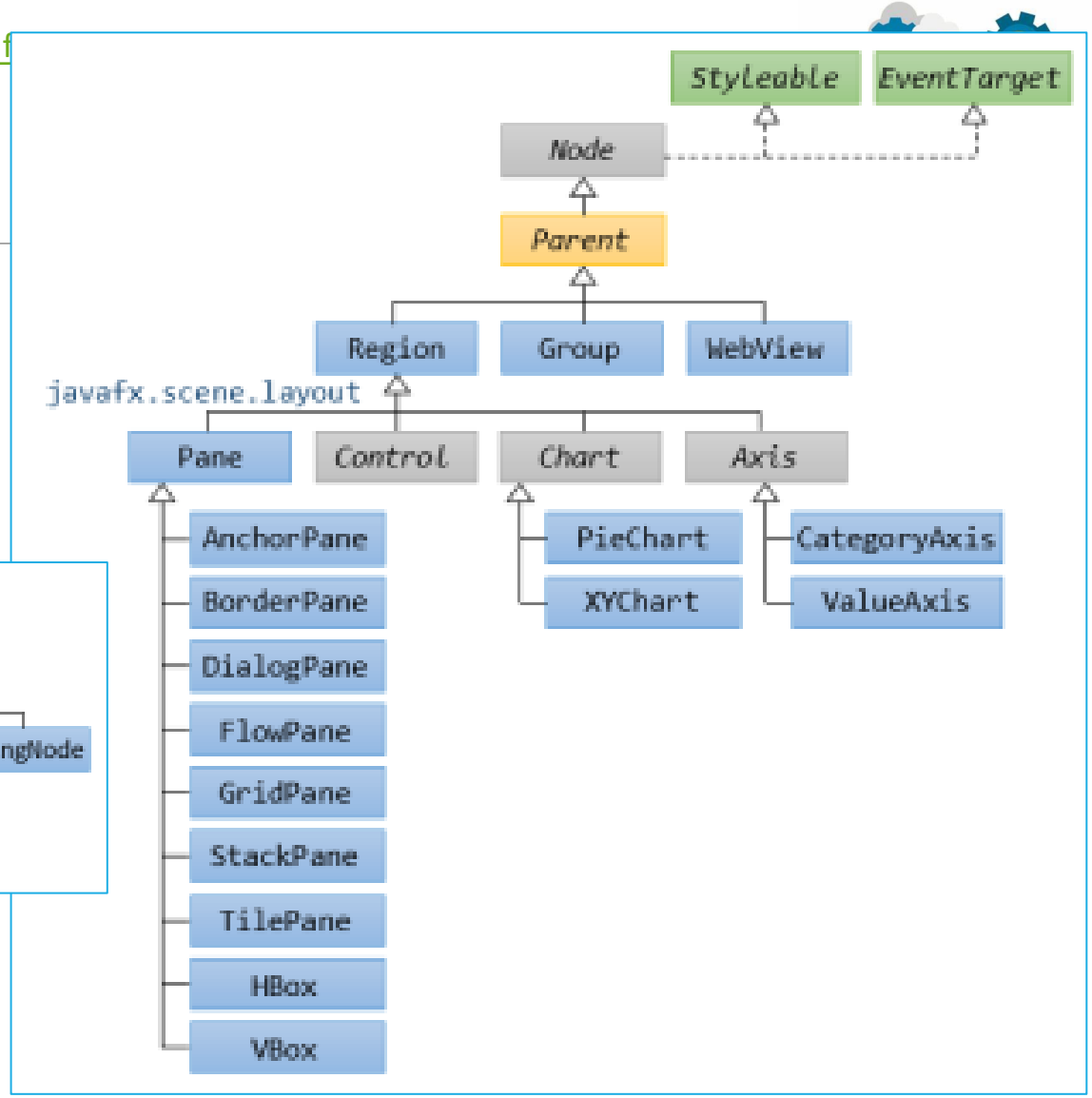
- There are two categories
  - Inheritance of CSS property types
    - All CSS properties declared in a JavaFX class are inherited by all its subclasses (e.g. if declare a property in the `Node` class, it will be available for all node types, as they are subclasses of `Node`).

```
// using FXML and CSS
Parent root = FXMLLoader.load(getClass().getResource( name: "sample.fxml"));

// parent style sheet
root.setStyle("-fx-font-size: 20;");

Scene scene = new Scene(root,  v: 500,  v1: 500);
```

https://www.ntu.edu.sg/home/ehchua/programming/java/Javaf

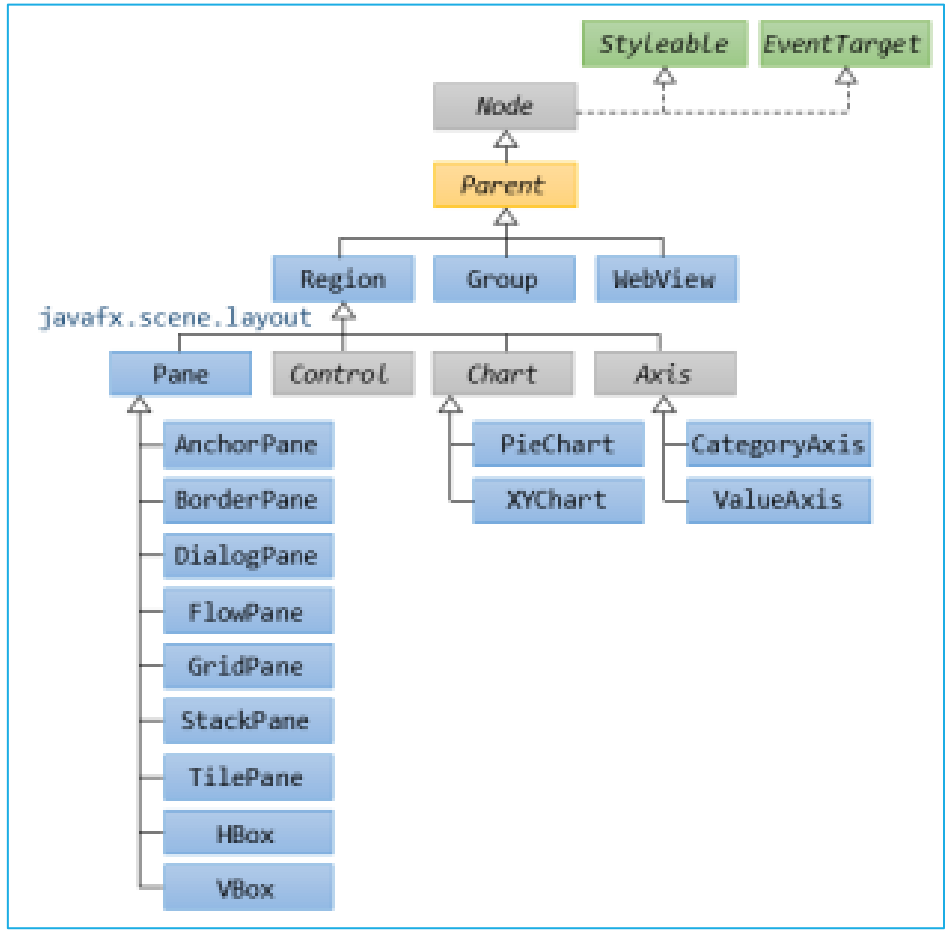
JavaFX Hierarchy

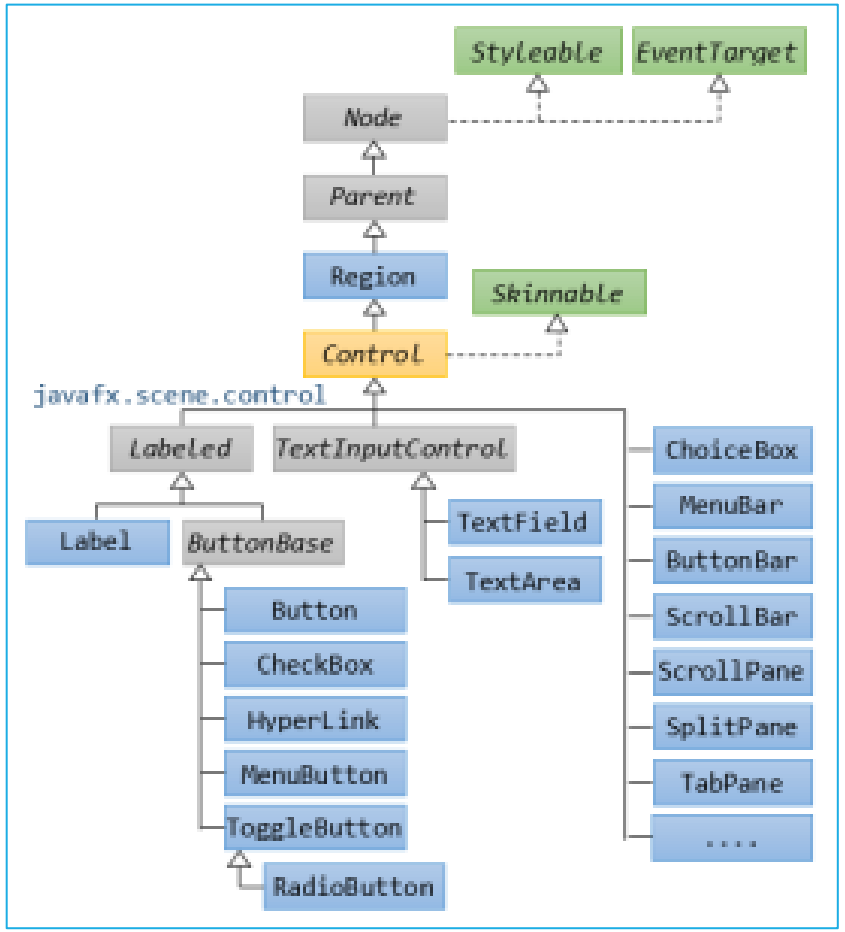




https://www.ntu.edu.sg/home/ehchua/programming/java/Javafx1_intro.html

JavaFX Hierarchy



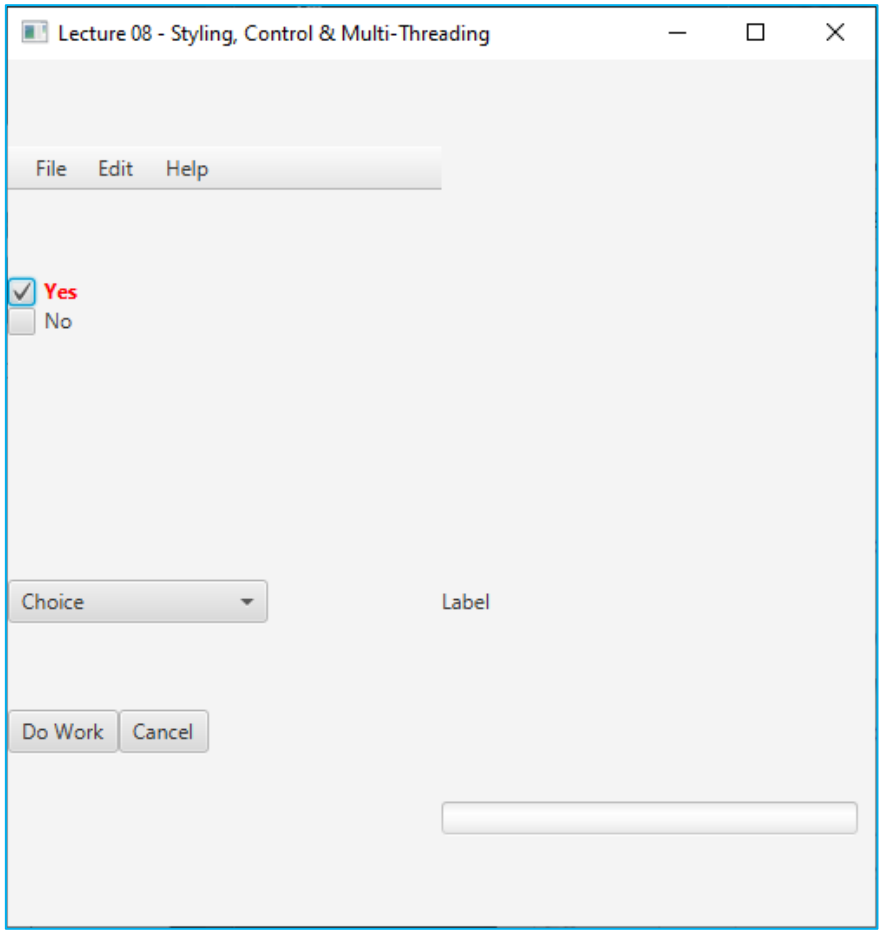


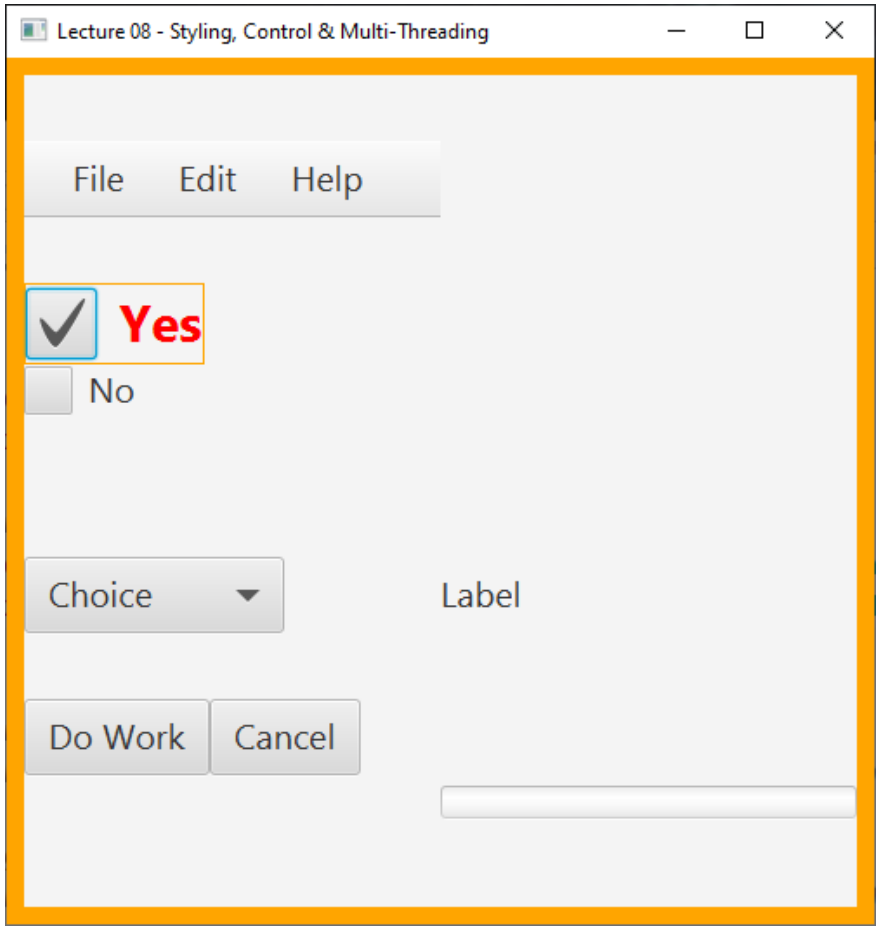
Inherit CSS Properties

- There are two categories
  - Inheritance of CSS property values
    - A CSS property for a node (e.g. `CheckBox`) may inherit its value from its parent (e.g. `GridPane`).

```
gridPane.setStyle("-fx-border-width: 10; -fx-border-color: orange; -fx-font-size: 20;");
checkBoxYes.setStyle("-fx-text-fill: red; -fx-font-weight: bold; " +
                     "-fx-border-color: inherit; -fx-font-size: 30");
```

Inherit CSS Properties

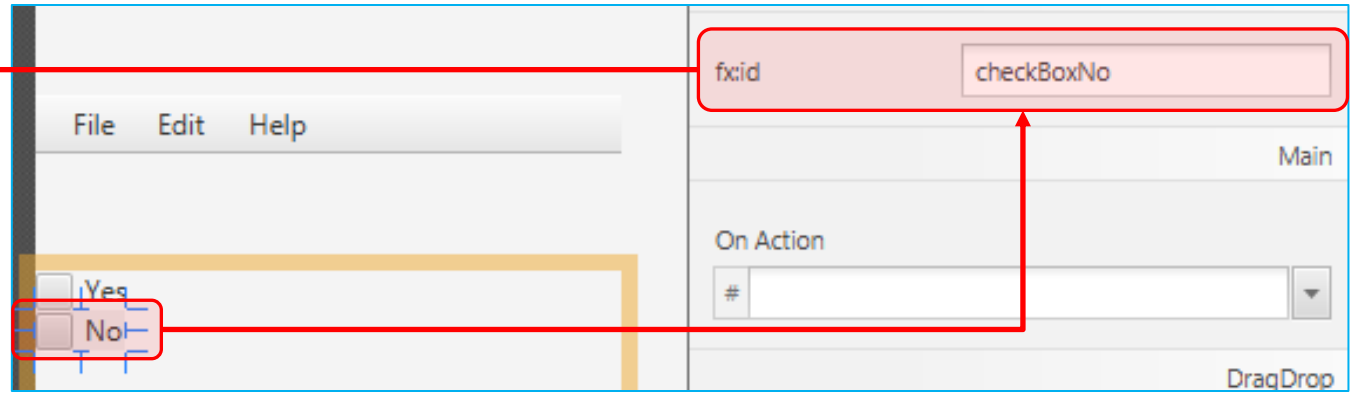




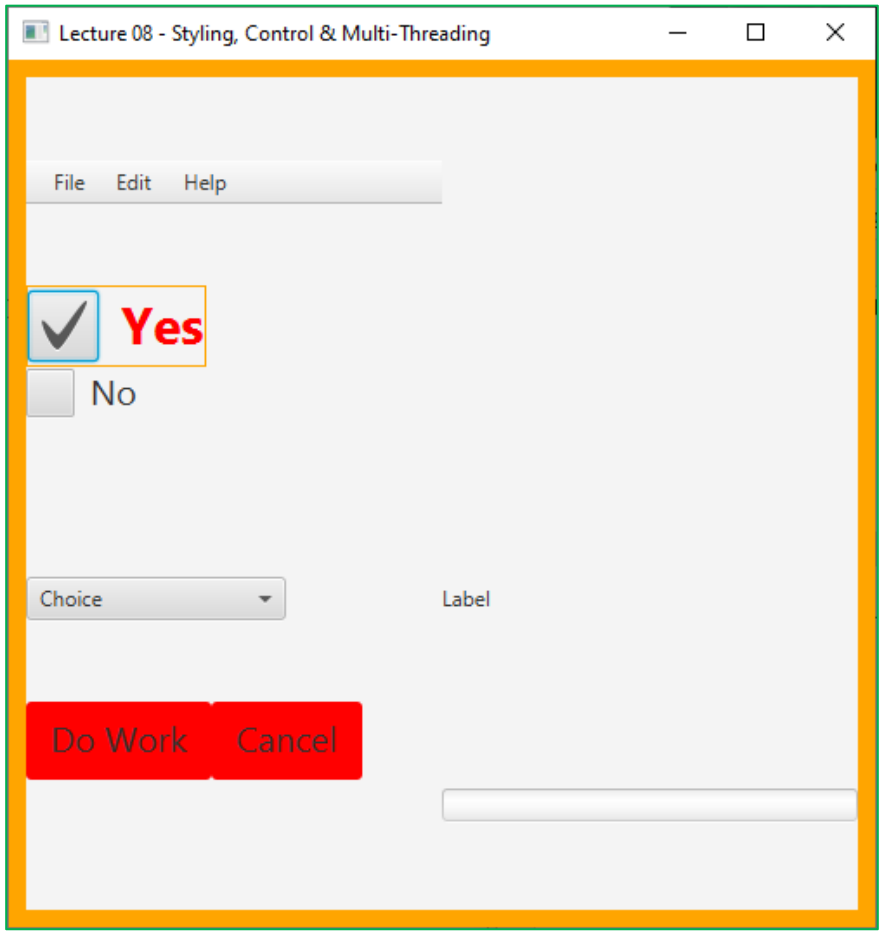
Use ID Selectors

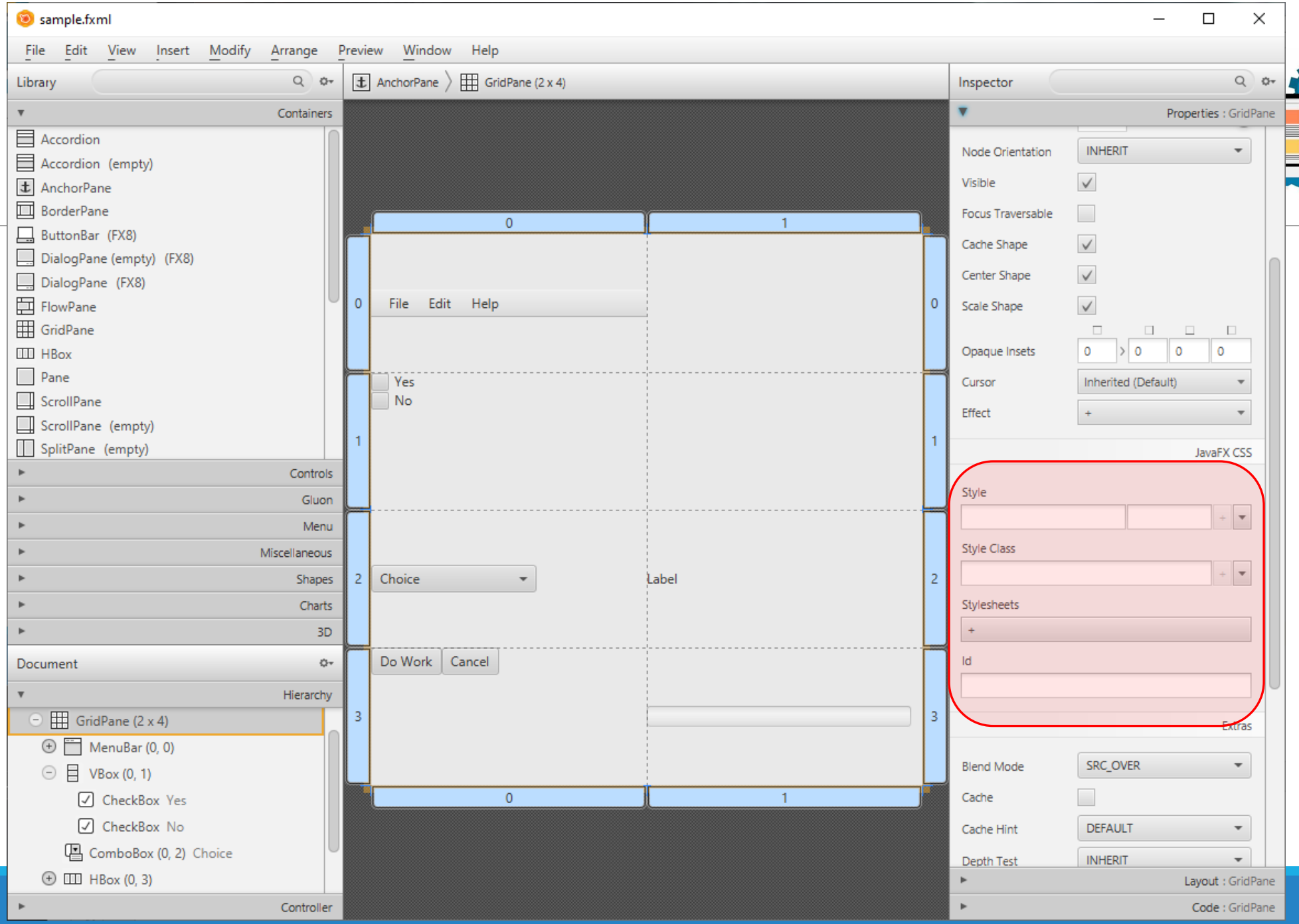
- Can also use the ID's created in SceneBuilder to set styles for individual nodes in the CSS style sheet.

```
.check-box {
    -fx-font-size: 30;
}

#checkBoxNo {
    -fx-font-size: 20;
}
```



Use ID Selectors




sample.fxml
File Edit View Insert Modify Arrange Preview Window Help
Library
Containers
Accordion
Accordion (empty)
AnchorPane
BorderPane
ButtonBar (FX8)
DialogPane (empty) (FX8)
DialogPane (FX8)
FlowPane
GridPane
HBox
Pane
ScrollPane
ScrollPane (empty)
SplitPane (empty)
Controls
Gluon
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
GridPane (2 x 4)
MenuBar (0, 0)
VBox (0, 1)
CheckBox Yes
CheckBox No
ComboBox (0, 2) Choice
HBox (0, 3)
Controller
AnchorPane
GridPane (2 x 4)
File Edit Help
Yes
No
Choice
Label
Do Work
Cancel
Inspector
Properties : GridPane
Node Orientation
INHERIT
Visible
Focus Traversable
Cache Shape
Center Shape
Scale Shape
Opaque Insets
Cursor
Inherited (Default)
Effect
JavaFX CSS
Style
Style Class
Stylesheets
Id
Extras
Blend Mode
SRC_OVER
Cache
Cache Hint
DEFAULT
Depth Test
INHERIT
Layout : GridPane
Code : GridPane

Add StyleSheet with FXML

stylesgridpane.css

```
1  #gridPane {
2      -fx-border-width: 10;
3      -fx-border-color: orange;
4  }
```

JavaFX CSS

Style

Style Class

Stylesheets

@ stylesgridpane.css

Id

File Edit Help

Yes
No

Choice

Label

Do Work Cancel

University of Nottingham
UK | CHINA | MALAYSIA

JavaFX Advanced Topics

MULTI-THREADING: DEALING WITH UNRESPONSIVE GUIS

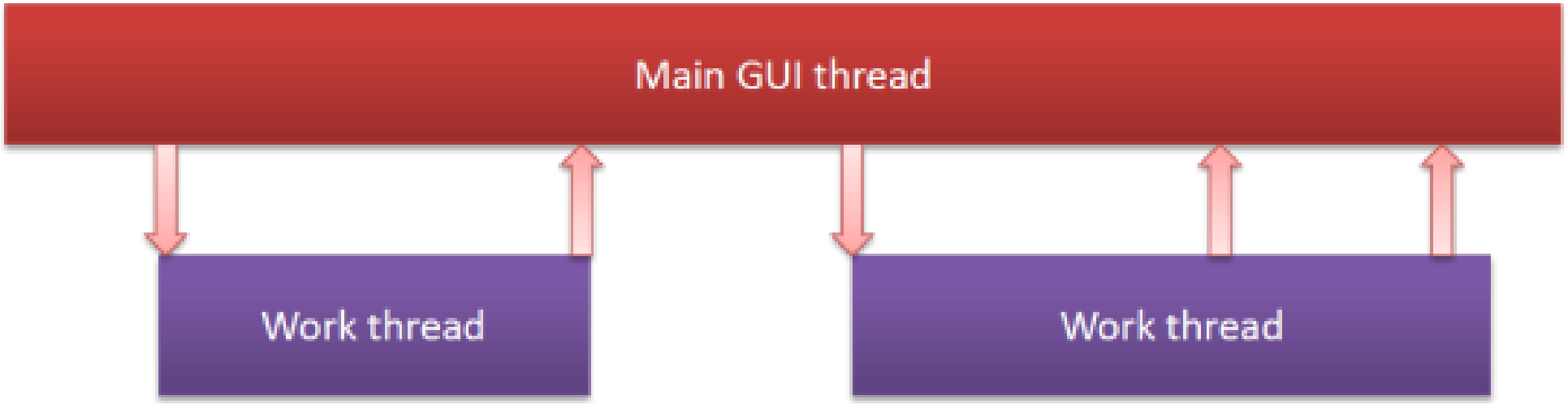
Stop GUIs Becomes Unresponsive

- The facts:
  - JavaFX launches the UI on a JavaFX Application thread.
  - This thread should be left handling the UI interaction.
  - Heavy computation should be done elsewhere to prevent freezing!

- JavaFX provides a solution:
  - Package "`javafx.concurrent`".
  - A way for JavaFX to create and communicate with other threads.

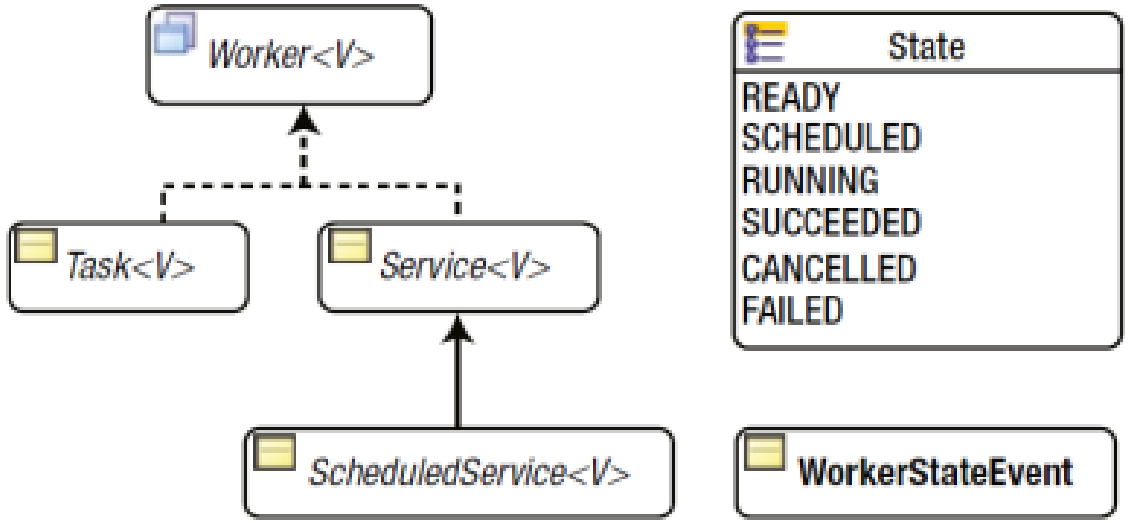
Stop GUIs Becomes Unresponsive

- JavaFX provides some helper classes.



Concurrency Framework

- Concurrency.
  - The ability of different parts of a program to be executed out-of-order or in partial order, without affecting the outcome.
- Classes in the JavaFX Concurrency Framework.



Concurrency Framework

- Framework consists of one interface + four classes + one enumerator.
  - Instance of a `Task` class represents a one-shot task.
  - Instance of a `Service` class represents a reusable task.
  - Instance of a `ScheduledService` class represents a reusable task that runs repeatedly after a specific interval.
  - Instance of `WorkerStateEvent` class represents an event that occurs as state of a Worker changes; can add event handlers to all three types of tasks to listen to the changes in their states.
  - Constants in the `Worker.State` enum represents different states of a worker.

Stop GUIs Becomes Unresponsive

- Worker interface.
  - Specifies the methods available to background threads when working with JavaFX.
  - Various useful methods such as `isRunning()`, `getProgress()`, `cancel()` …
- Task class.
  - Abstract class which implements the `Worker` interface.
  - Program the actual work that needs to be done on a separate thread.
    1. Extends the `Task` class.
    2. Implements `call()` to do the work.
       1. Don't directly touch UI components from here.
       2. Can update the UI with `updateProgress()`, `updateMessage()`, `updateTitle()` methods.
    3. Starts a new thread, passing "`Worker`" as a parameter.

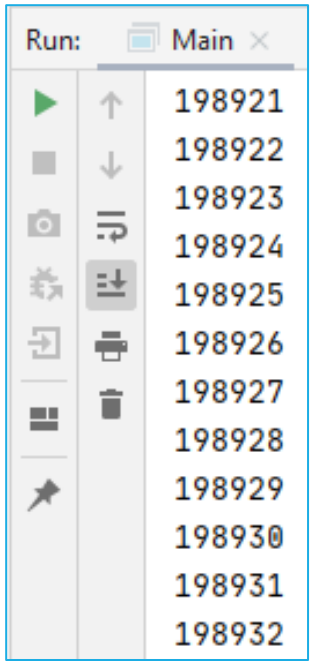
Stop GUIs Becomes Unresponsive

- Example: Build a progress bar to show some works done in another thread.

```
public class ProgressWorker extends Task {

    @Override
    protected Object call() throws Exception {
        int n = 250000;

        for(int i = 0; i < n; i++) {
            System.out.println(i);
            updateProgress(i, n);

            if(isCancelled())
                break;
        }

        return null;
    }
}
```

```
@FXML private ProgressBar progressBar;
private ProgressWorker progressWorker = new ProgressWorker();

public void doWorkOnClicked(ActionEvent actionEvent) {
    new Thread(progressWorker).start();
    progressBar.progressProperty().bind(progressWorker.progressProperty());
}

public void cancelOnClicked(ActionEvent actionEvent) {
    progressWorker.cancel();
}
```

Stop GUIs Becomes Unresponsive



University of Nottingham
UK | CHINA | MALAYSIA

JavaFX Advanced Topics

MULTIPLE WINDOWS

Multiple Window

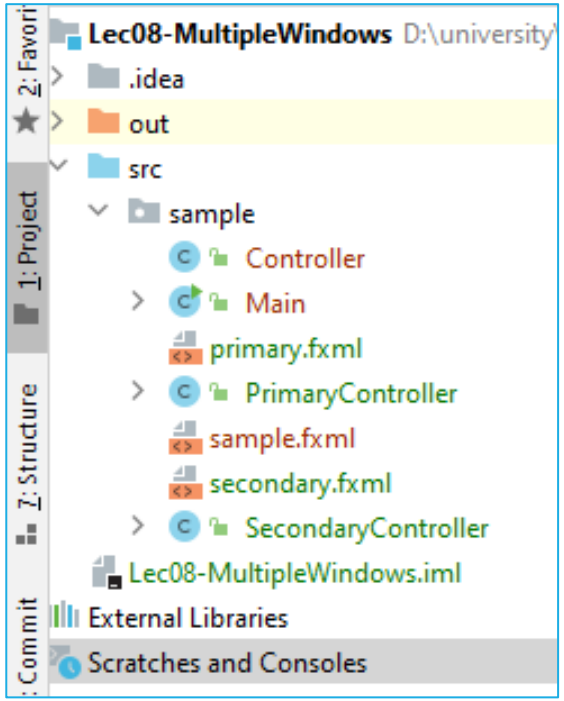
```java
public class Main extends Application {

    private static Scene scene;

    @Override
    public void start(Stage primaryStage) throws Exception {
        scene = new Scene(loadFXML("primary"));
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void setRoot(String fxml) throws IOException {
        scene.setRoot(loadFXML(fxml));
    }

    private static Parent loadFXML(String fxml) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource( name: fxml + ".fxml"));
        return fxmlLoader.load();
    }

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

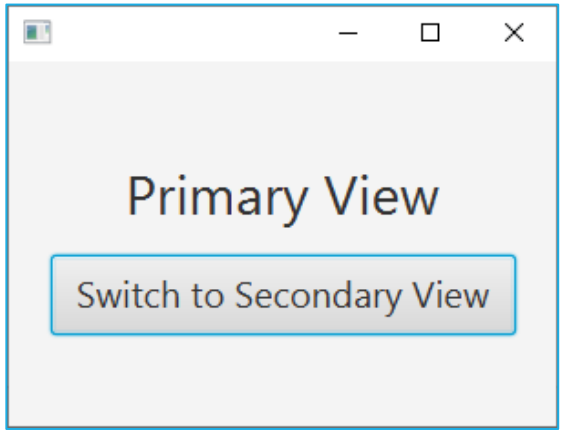
```java
public class PrimaryController {

    public void switchOnClicked(ActionEvent actionEvent) throws IOException {
        Main.setRoot("secondary");
    }

}
```

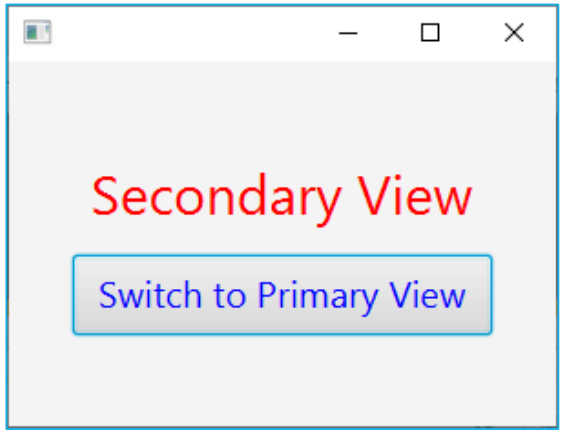
```java
public class SecondaryController {

    public void switchOnClicked(ActionEvent actionEvent) throws IOException {
        Main.setRoot("primary");
    }

}
```

Multiple Windows

- Creating packages within the resource folder allowed for an IDE independent implementation.







University of Nottingham
UK | CHINA | MALAYSIA

JavaFX Advanced Topics

ANIMATIONS

Key Concepts for Timeline Animation

- Timeline.
  - Denotes the progression of time during animation with an associated key frame at a given instance.
- Key Frame.
  - Represents the state of the node being animated at a specific instant on the timeline.
- Key Value.
  - Represents the value of a property of a node along with an interpolator to be used.
- Transition.
  - Denotes the built-in transition animation (the `Transition` class) to perform common animated effects.

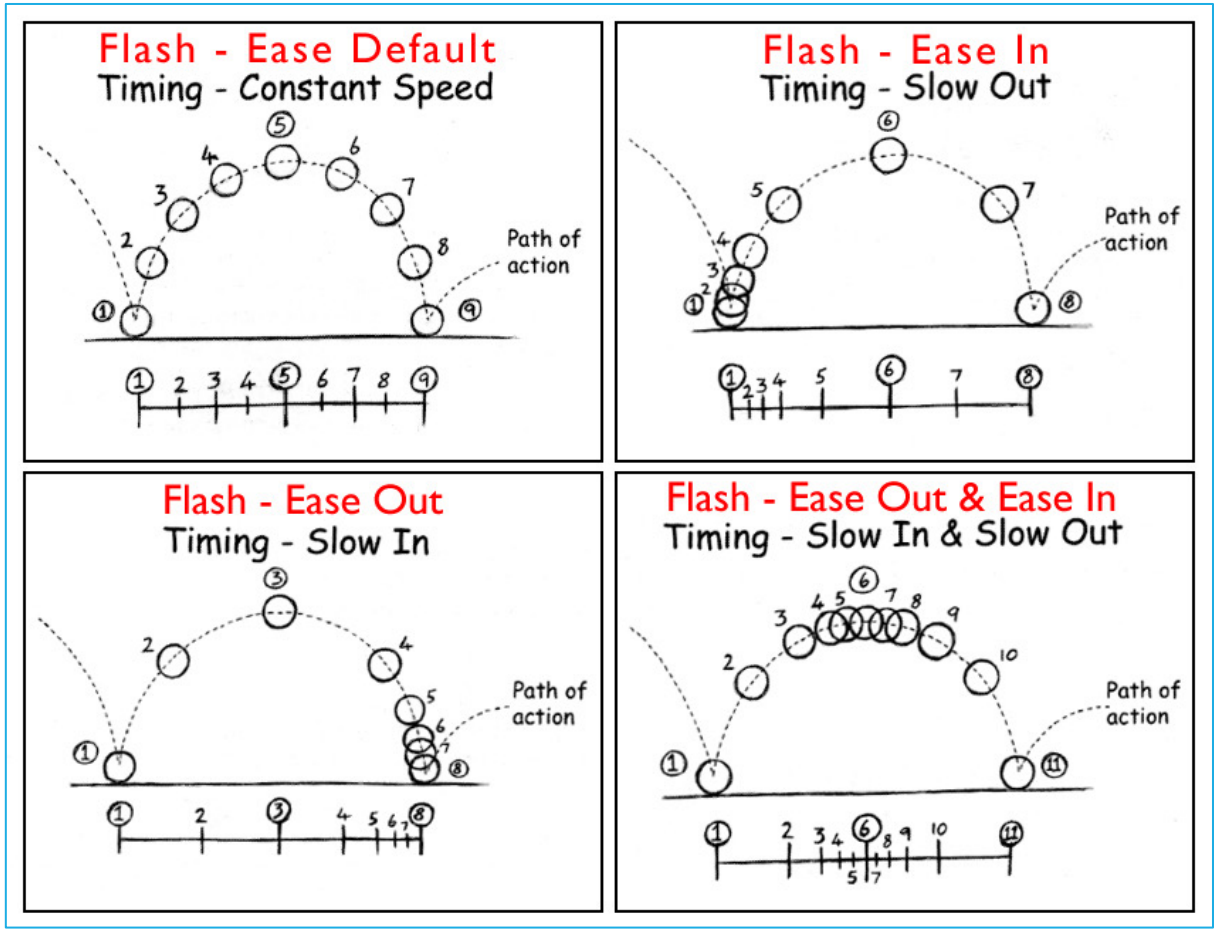
Key Values

- Code snippet below defines a `KeyValue` instance that targets a rectangle node's opacity property starting at 1 and ending the value with 0.

```
Rectangle rectangle = new Rectangle(0, 0, 50, 50);
KeyValue keyValue = new KeyValue(rectangle.opacityProperty(), 0);
```

- The `KeyValue` object doesn't interpolate the value, but simply defines the start and end values of a property to interpolate between.
- Also, the `KeyValue` can be defined with different types of interpolators such as linear, ease in or ease out.

Key Values


Flash - Ease Default
Timing - Constant Speed
Path of action
Flash - Ease In
Timing - Slow Out
Path of action
Flash - Ease Out
Timing - Slow In
Path of action
Flash - Ease Out & Ease In
Timing - Slow In & Slow Out
Path of action

Key Values

- The code below defines a key value that will animate a rectangle from left to right by 100 pixels having an `Interpolator.EASE_OUT` interpolator.

```
Rectangle rectangle = new Rectangle(0, 0, 50, 50);
KeyValue keyValue = new KeyValue(rectangle.xProperty(), 100, Interpolator.EASE_OUT);
```

- Easing out will slow down the fade before the end key value.

Key Frames

- When an animation (`Timeline`) occurs, each timed event, called a keyframe (a `KeyFrame` object), is responsible for interpolating key values (`KeyValue` objects) over a period of time (`javafx.util.Duration`).
- A constructor requires a timed duration (`Duration`) to interpolate over key values when creating a `KeyFrame` object.
- The `KeyFrame` constructors all accept one or more key values by using a variable containing an argument list or a collection of `KeyValue` objects.

Key Frames

- Code snippet below defines a keyframe that has a duration of 1000 milliseconds and two key values that represent the rectangle's x and y properties to demonstrate moving a rectangle in a diagonal direction from upper left to lower right.

```
Rectangle rectangle = new Rectangle(0, 0, 50, 50);
KeyValue xValue = new KeyValue(rectangle.xProperty(), 100);
KeyValue yValue = new KeyValue(rectangle.yProperty(), 100);

KeyFrame keyFrame = new KeyFrame(Duration.millis(1000), xValue, yValue);
```

- A `KeyFrame` object is defined to interpolate over the `x` and `y` properties.
- The code animates the rectangle to move in a southeast direction for one second.

Timeline

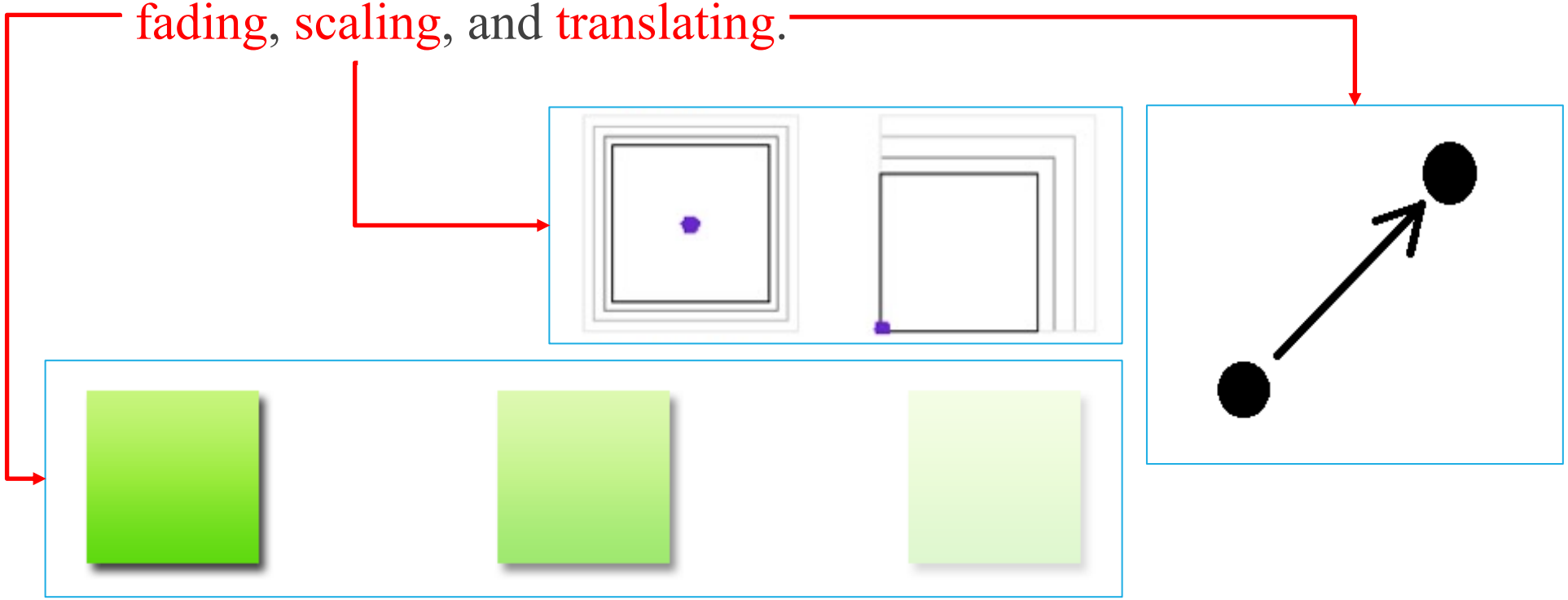
- A `Timeline` is one animation sequence consisting of many `KeyFrame` objects that run sequentially.

```
Timeline timeline = new Timeline();
timeline.setCycleCount(Timeline.INDEFINITE);
timeline.setAutoReverse(true);
timeline.getKeyFrames().addAll(keyFrame1, keyFrame2);
timeline.play();
```

- The cycle count is the number of times the timeline to play animation; use `Timeline.INDEFINITE` to play the animation indefinitely.
- The auto-reverse property is a Boolean flag to indicate that the animation can play the timeline backward (the keyframes in reverse order).
- By default, the cycle count is set to 1, and auto-reverse is set to `false`.

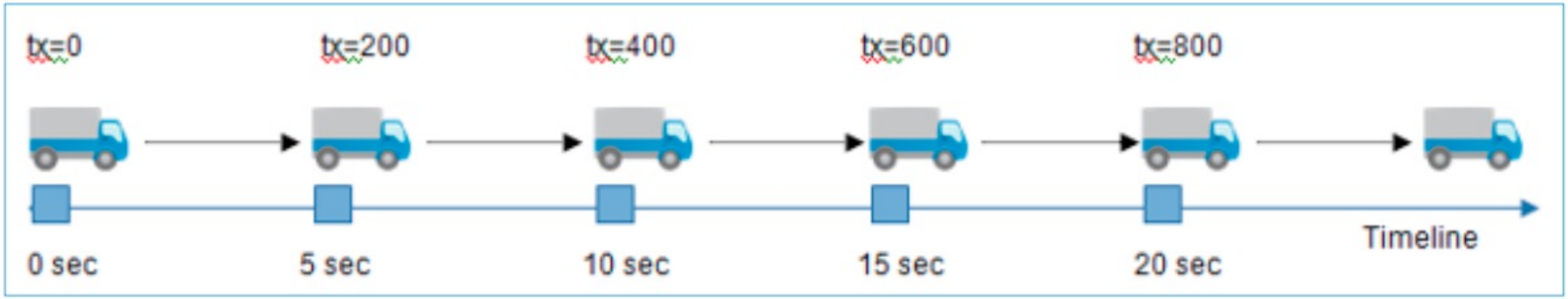
Timeline

- However, there are easier ways to express common animation effects such as fading, scaling, and translating.



Key Concepts for Timeline Animation

- Lorries represent key frames at specific instants of the timeline.
- The key values (here the value for the `translateX` property) associated with key frames are shown at the top.
- By default, linear interpolation is used between the key frames.



Key Concepts for Timeline Animation

- Using timeline animation involves the following steps.
  - Constructor key frames.
  - Create timeline object with key frames.
  - Set the animation properties.
  - Use the `play()` method to run the animation.
- The timeline keeps all key frames in the `ObservableList<KeyFrame>` object.
- The `getKeyFrames()` method returns the list.

Sample Animation

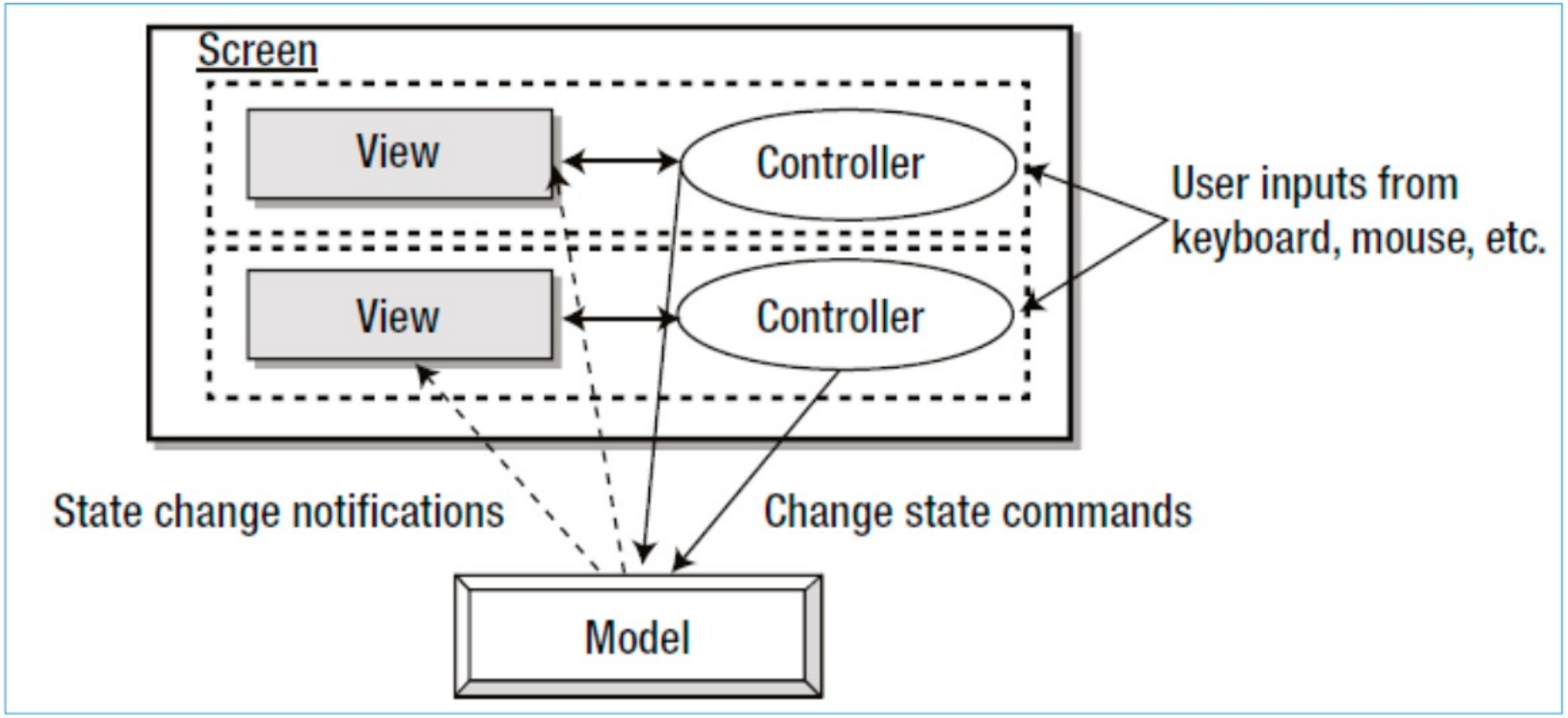
```
Rectangle rectangle = new Rectangle( v: 0,  v1: 0,  v2: 100,  v3: 50);

KeyValue keyValueX = new KeyValue(rectangle.xProperty(),  t: 100, Interpolator.EASE_OUT);
KeyValue keyValueY = new KeyValue(rectangle.yProperty(),  t: 100, Interpolator.EASE_OUT);
KeyValue keyValueOpacity = new KeyValue(rectangle.opacityProperty(),  t: 0);

KeyFrame keyFrame = new KeyFrame(Duration.millis(1000), keyValueX, keyValueY, keyValueOpacity);

Timeline timeline = new Timeline();
timeline.setCycleCount(Timeline.INDEFINITE);
timeline.setAutoReverse(true);
timeline.getKeyFrames().addAll(keyFrame);
timeline.play();

Group root = new Group();
root.getChildren().add(rectangle);

primaryStage.setTitle("Lecture 08 - Animation");
primaryStage.setScene(new Scene(root,  v: 300,  v1: 300));
primaryStage.show();
```

Transitions

- `javafx.animation.FadeTransition`
  - Targets the node's opacity property for a fading animation effect.
- `javafx.animation.PathTransition`
  - Enables a node to follow a generated path.
- `javafx.animation.ScaleTransition`
  - Targets a node's `scaleX`, `scaleY`, and `scaleZ` properties to resize a node.
- `javafx.animation.TranslateTransition`
  - Targets a node's `translateX`, `translateY`, and `translateZ` properties to move a node across the screen.

See more details at https://docs.oracle.com/javafx/2/animations/basics.htm#CJAJJAGI

MVC Design Pattern



Useful Resources

FOR GUI SKILLS IN MORE DEPTH

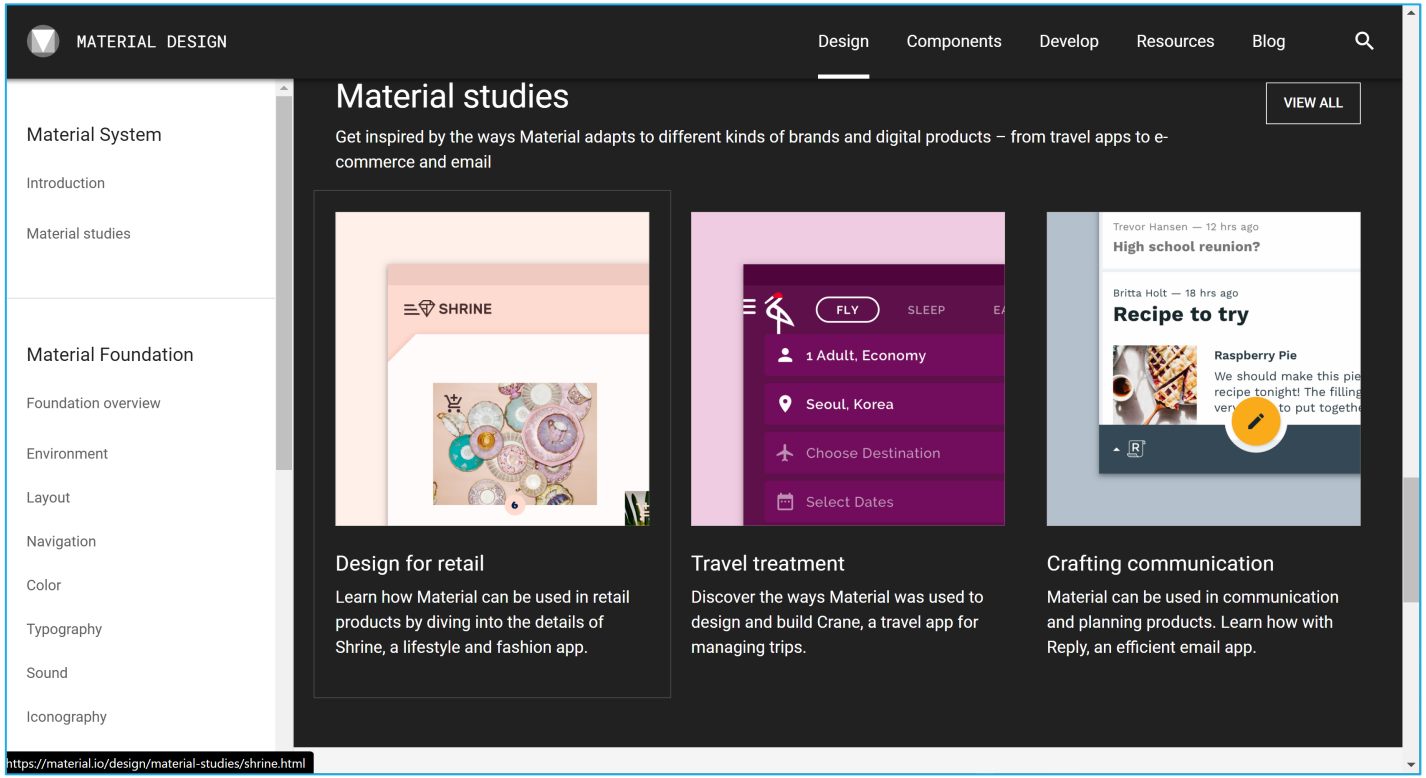
References

- CSS
  - JavaFX CSS Tutorial: https://examples.javacodegeeks.com/desktop-java/javafx/javafx-css-tutorial/.
  - JavaFX CSS Reference Guide: https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html.
  - Styling FX Buttons with CSS: http://fxexperience.com/2011/12/styling-fx-buttons-with-css/.

- JavaFX
  - JavaFX Tutorial: http://tutorials.jenkov.com/javafx/index.html

References

- Excellent material design guide.
  - https://material.io/design



References

- Multithreading & Concurrent Programming.
  - https://www.ntu.edu.sg/home/ehchua/programming/java/J5e_multithreading.html.

- JavaFX Animation Example.
  - https://examples.javacodegeeks.com/desktop-java/javafx/javafx-animation-example-2/.

Put Your Mind in Maintenance Mode