

# CS 213 – Software Methodology

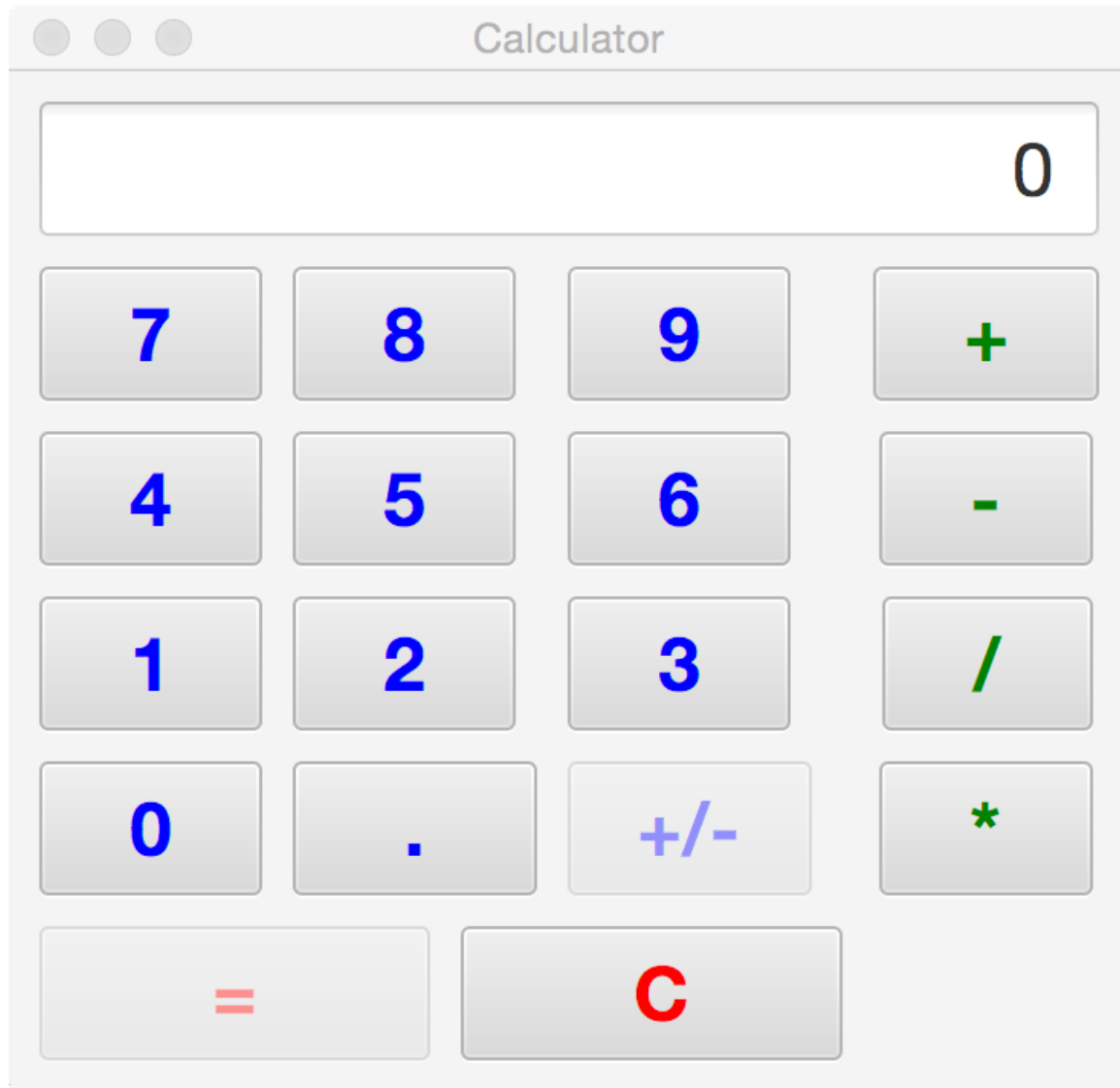
## Spring 2023

*Sesh Venugopal*

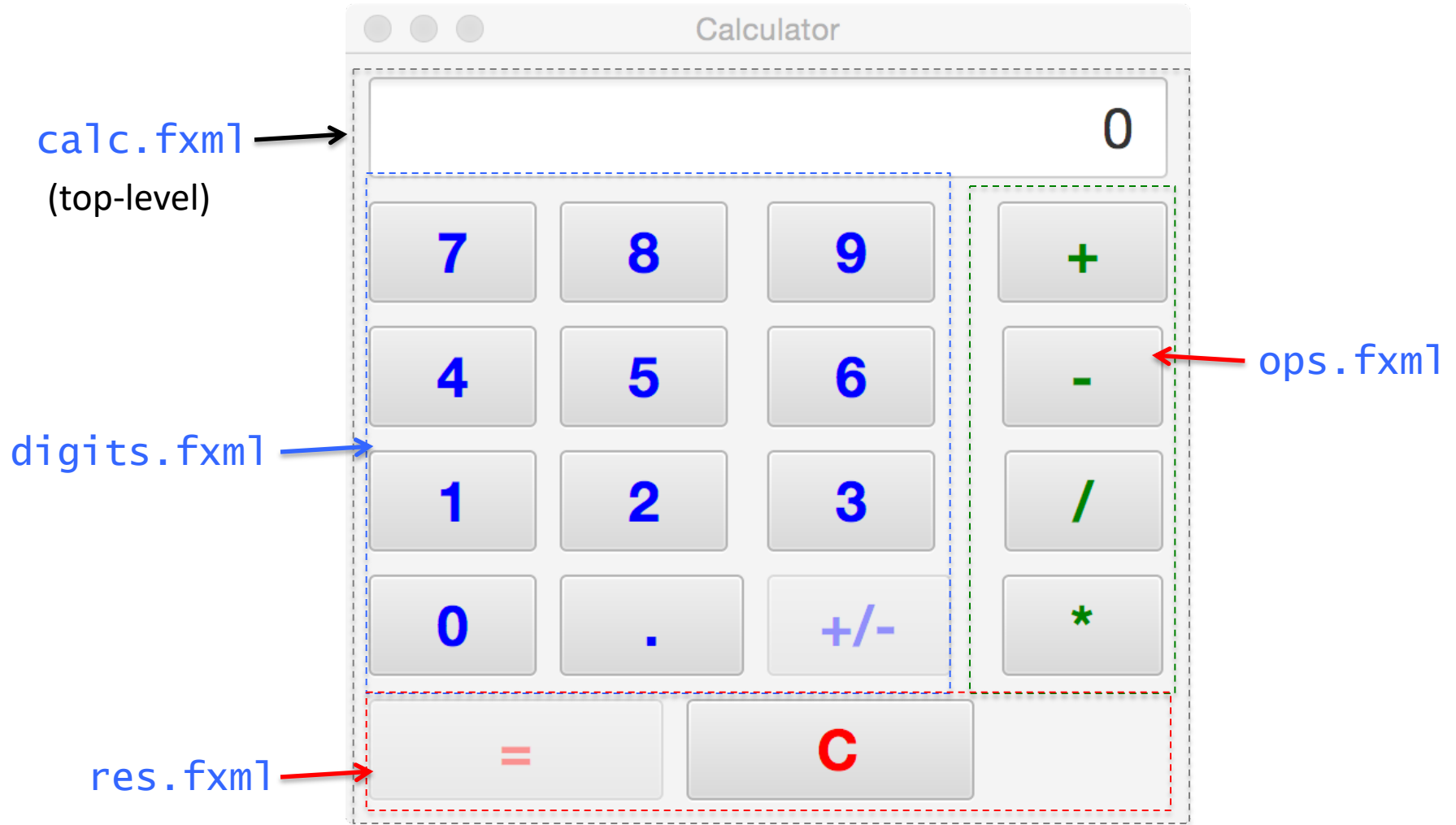
Mar 22

Calculator UI – Nested UI Structures

# Building a Calculator - UI



# Calculator UI – Nested UI Entities



# Nested UI Entities - FXMLs

calc.fxml

```
...
<?import java.net.*?>
<GridPane
  ...
  fx:controller="view.CalcController">
  ...
  <TextField fx:id="display" editable="false" alignment="CENTER_RIGHT"
    GridPane.columnSpan="2" />


  <fx:include fx:id="digits" source="digits.fxml" GridPane.rowIndex="1" />
  <fx:include fx:id="ops" source="ops.fxml" GridPane.columnIndex="1"
    GridPane.rowIndex="1" />
  <fx:include fx:id="res" source="res.fxml" GridPane.rowIndex="2" />

  <stylesheets>
    <URL value="@calc.css" />
  </stylesheets>

</GridPane>
```

Need this for URL tag used with CSS styling (see bottom)

You can nest UI entities with their own FXML layouts, with `fx:include`



```
.button {
  -fx-font-size: 18pt;
  -fx-font-weight: bold;
}
.text-field {
  -fx-font-size: 18pt;
}
```

# Matching Nested Entities with Controllers

calc.fxml

```
...
fx:controller="view.CalcController">
...
<fx:include fx:id="digits" source="digits.fxml" ... />
<fx:include fx:id="ops" source="ops.fxml" ... />
<fx:include fx:id="res" source="res.fxml"... />
```

CalcController.java


```
public class CalcController {
    ...
    @FXML
    protected DigitController digitsController;

    @FXML
    protected OperatorController opsController;

    @FXML
    protected ResultController resController;
    ...
}
```

**IMPORTANT!!**

The names of the controllers for the contained UI FXMLs must match the ids that go with fx:include in the container's FXML



The diagram illustrates the matching process between FXML IDs and Java controller names. A large red arrow points from the 'IMPORTANT!!' text to the 'fx:id' attributes in the FXML code. Three smaller red arrows point from the controller names in the Java code to the corresponding 'fx:id' values in the FXML code: 'digitsController' to 'digits', 'opsController' to 'ops', and 'resController' to 'res'. A red circle highlights the controller names in the Java code.

<GridPane

...  
fx:controller="view.DigitController">

...  
<Button fx:id="d7" onAction="#digitPressed" text=" 7 " />  
<Button fx:id="d8" onAction="#digitPressed" text=" 8 " ... />  
<Button fx:id="d9" onAction="#digitPressed" text=" 9 " ... />  
<Button fx:id="d4" onAction="#digitPressed" text=" 4 " ... />  
<Button fx:id="d5" onAction="#digitPressed" text=" 5 " ... />  
<Button fx:id="d6" onAction="#digitPressed" text=" 6 " ... />  
<Button fx:id="d1" onAction="#digitPressed" text=" 1 " ... />  
<Button fx:id="d2" onAction="#digitPressed" text=" 2 " ... />  
<Button fx:id="d3" onAction="#digitPressed" text=" 3 " ... />  
<Button fx:id="d0" onAction="#digitPressed" text=" 0 " ... /> —

...  
<fx:define> ← needs <?import java.util.\*?>

<ArrayList fx:id="digitButtons">

<fx:reference source="d0"/>

<fx:reference source="d1"/>

...  
<fx:reference source="d8"/>

<fx:reference source="d9"/>

</ArrayList>

</fx:define>

...  
</GridPane>

source values are the ids declared elsewhere in the FXML

This set up does away with the inefficiency of defining one @FXML field per button in the controller class

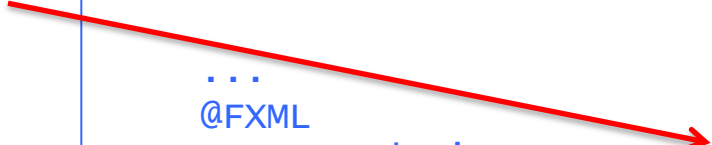
# Matching ArrayList of fx:define to Java Code

digits.fxml

```
...  
<fx:define>  
  <ArrayList fx:id="digitButtons">  
    ...  
  </ArrayList>  
</fx:define>  
...  
<stylesheets>  
  <URL value="@digits.css" />  
</stylesheets>
```

```
public class DigitController {
```


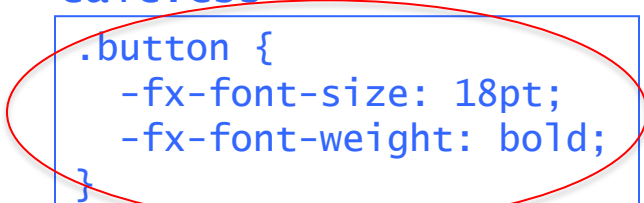
```
...  
@FXML  
protected List<Button> digitButtons;  
...  
}
```



## Container's styling is inherited by contained UIs

calc.css

```
.button {  
  -fx-font-size: 18pt;  
  -fx-font-weight: bold;  
}  
.text-field {  
  -fx-font-size: 18pt;_  
}
```



digits.css

```
.button {  
  -fx-text-fill: blue;  
}
```

digits will be boldfaced and have font size of 18 pt, as defined in calc.css, plus color blue as defined in digits.css