CS 213 – Software Methodology

Spring 2019

Sesh Venugopal

Lecture 5 – Feb 5

Graphical User Interface

Recommend you use Java 8/FX 8

Java SDK version 8 comes bundled with FX, making it a lot easier to work with FX projects

You can still use the latest Eclipse version (2018-12?), just make sure you set complier compliance level to 1.8, and use a project specific JRE of Java SE 8 (1.8.x)

- Install e(fx)plugin, see:

https://www.eclipse.org/efxclipse/install.html

- To create FX project in Eclipse, do:

  File -> New -> Other -> JavaFX -> JavaFX Project

- Class containing main must be a subclass of `javafx.application.Application`

Preparing to build GUIs in Java FX

If you have Java version 11, you need to install FX separately and have Eclipse pair with it:
https://openjfx.io/openjfx-docs/#IDE-Eclipse

- Install e(fx)plugin, see:
https://www.eclipse.org/efxclipse/install.html

- To create FX project in Eclipse, do:
  File -> New -> Other -> JavaFX -> JavaFX Project
(and if you have separate FX installed – for Java v11 – then follow steps in Non-modular projects section of to have your project see the Java FX 11 libraries)

- Class containing main must be a subclass of `javafx.application.Application`

Fahrenheit-Celsius Converter

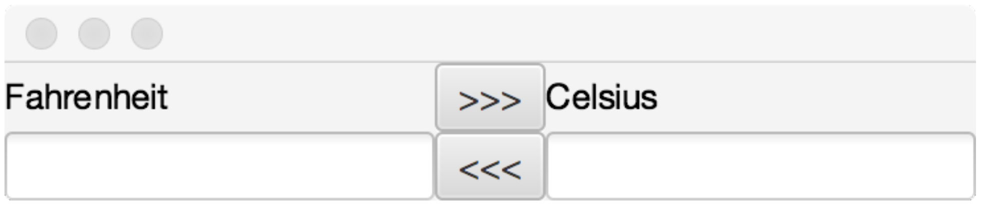
Version 1

Programmatic Layout

Programmatic Layout – Widgets/ Layout

```java
@Override
public void start(Stage primaryStage) {
    GridPane root = makeGridPane();
    Scene scene = new Scene(root);
    primaryStage.setScene(scene);
    primaryStage.show();
}

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

```java
private static GridPane makeGridPane() {

    // all the widgets
    Text fText = new Text("Fahrenheit");
    Text cText = new Text("Celsius");
    TextField f = new TextField();
    TextField c = new TextField();
    Button f2c = new Button(">>>");
    Button c2f = new Button("<<<");

    GridPane gridPane = new GridPane();
    gridPane.add(fText, 0, 0);
    gridPane.add(f2c, 1, 0);
    gridPane.add(cText, 2, 0);
    gridPane.add(f, 0, 1);
    gridPane.add(c2f, 1, 1);
    gridPane.add(c, 2, 1);

    return gridPane;
}
```

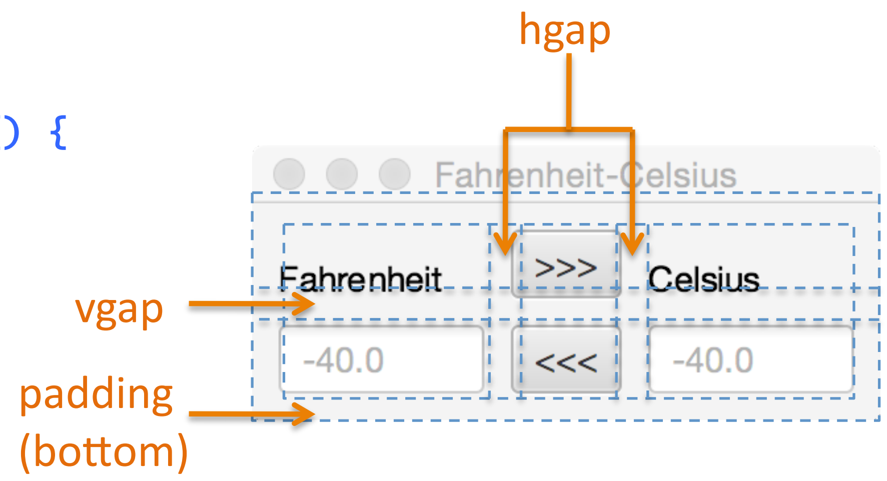


Programmatic Layout – Gaps/Alignment

```
@Override
public void start(Stage primaryStage) {
    ...
    primaryStage.setTitle("Fahrenheit-Celsius");
    primaryStage.setResizable(false);
    primaryStage.show();
}

private static GridPane makeGridPane() {

    // all the widgets
    ...

    f.setPrefColumnCount(5);
    f.setPromptText("-40.0");
    c.setPrefColumnCount(5);
    c.setPromptText("-40.0");
    gridPane.setHgap(10);
    gridPane.setVgap(10);
    gridPane.setPadding(new Insets(10,10,10,10));
    GridPane.setValignment(fText, VPos.BOTTOM);
    GridPane.setValignment(cText, VPos.BOTTOM);

    return gridPane;
}
```

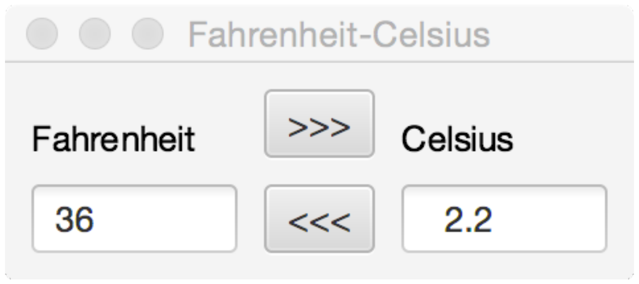
t r b l

text aligned with bottom of its grid cell



Programmatic Layout – Event Handling

```
private static GridPane makeGridPane() {

    ... // all the widgets

    ... // gaps and alignment

    // event handling
    f2c.setOnAction(new EventHandler<ActionEvent>() {
        public void handle(ActionEvent e) {
            float fval = Float.valueOf(f.getText());
            float cval = (fval-32)*5/9;
            c.setText(String.format("%5.1f", cval));
        }
    });

    c2f.setOnAction(new EventHandler<ActionEvent>() {
        public void handle(ActionEvent e) {
            float cval = Float.valueOf(c.getText());
            float fval = cval*9/5+32;
            f.setText(String.format("%5.1f", fval));
        }
    });

    return gridPane;
}
```



Fahrenheit-Celsius Converter

Version 2

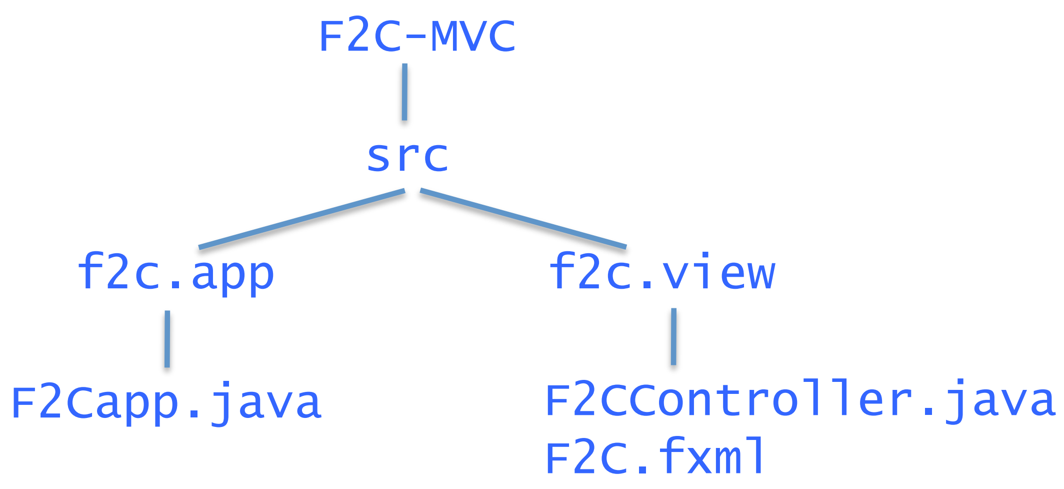
Separating View (UI in fxml) from Controller (Java code)

The MVC Code Architecture (Model-View-Controller)

Model is the set of classes that store and manage the data

View is the set of Java classes and non-Java design artifacts (e.g. xml, css, etc.) that implement the user interface

Controller is the set of classes that broker between Model and View



(There is not always a separate Model, and each of M, V, and C need not always be in its own separate package)

View: Layout using fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.text.*?>
<?import javafx.geometry.*?>
```

Don't forget imports!! (Editor won't flag errors for unresolved tags.)

```
<GridPane
    xmlns="http://javafx.com/javafx/8.0.60"
    xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="f2c.view.F2CController"
    vgap="10" hgap="10">
```

Name space for Java FX tags (e.g. Text)

Name space for FXML tags (e.g. fx:controller)

Row and column indexes default to 0

```
    <Text text="Fahrenheit" GridPane.valignment="BOTTOM"/>
    <Button text="&gt;&gt;&gt;" GridPane.columnIndex="1" />
    <Text text="Celsius" GridPane.columnIndex="2" GridPane.valignment="BOTTOM"/>
    <TextField prefColumnCount="10" promptText="-40.0" GridPane.rowIndex="1" />
    <Button text="&lt;&lt;&lt;" GridPane.rowIndex="1" GridPane.columnIndex="1" />
    <TextField prefColumnCount="10" promptText="-40.0"
           GridPane.rowIndex="1" GridPane.columnIndex="2" />
    <padding>
       <Insets top="10" right="10" bottom="10" left="10"/>
    </padding>
</GridPane>
```

View: Set up SceneBuilder

- Get SceneBuilder at Gluon:

http://gluonhq.com/products/scene-builder/

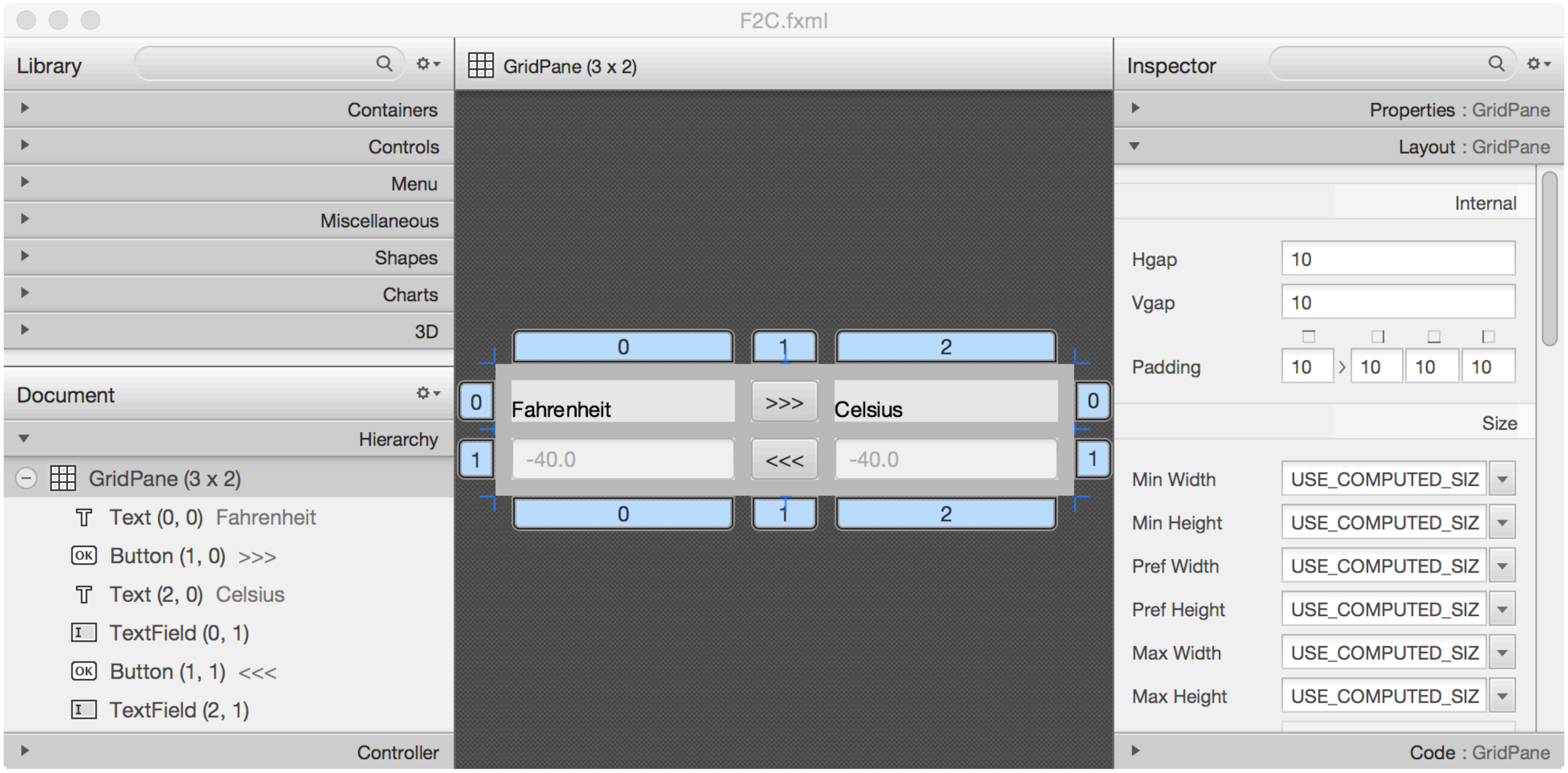
(This will allow SceneBuilder to be opened from within Eclipse)

- You can construct UIs exclusively using SceneBuilder interface, or you can write up the UI in an editor and optionally verify/polish using SceneBuilder
- To open SceneBuilder from Eclipse on an fxml file:

  Right click on fxml file -> open with -> other -> external programs (radio button) -> SceneBuilder

(or you can set Preferences -> JavaFX in Eclipse for the SceneBuilder executable and then right click on fxml file -> Open with SceneBuilder)

Verify fxml Layout with SceneBuilder



GridPane (3 x 2)
- Text (0, 0) Fahrenheit
- Button (1, 0) >>>
- Text (2, 0) Celsius
- TextField (0, 1)
- Button (1, 1) <<<
- TextField (2, 1)

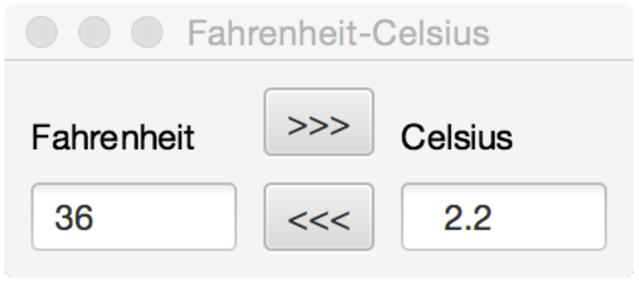
Hgap	10
Vgap	10
Padding	10 10 10 10
Min Width	USE_COMPUTED_SIZ
Min Height	USE_COMPUTED_SIZ
Pref Width	USE_COMPUTED_SIZ
Pref Height	USE_COMPUTED_SIZ
Max Width	USE_COMPUTED_SIZ
Max Height	USE_COMPUTED_SIZ

(In SceneBuilder, do Preview -> Show Preview in Window to simulate behavior)

fxml Layout – Id’ing widgets/event handler

```
...

<Text text="Fahrenheit" GridPane.valignment="BOTTOM"/>

<Button fx:id=“f2c” text="&gt;&gt;&gt;" GridPane.columnIndex="1"
    onAction=“#convert” />

<Text text="Celsius" GridPane.columnIndex="2" GridPane.valignment="BOTTOM"/>

<TextField fx:id=“f” prefColumnCount="10" promptText="-40.0"
    GridPane.rowIndex="1" />

<Button fx:id=“c2f” text="&lt;&lt;&lt;" GridPane.rowIndex="1"
    GridPane.columnIndex="1" onAction=“#convert” />

<TextField fx:id=“c” prefColumnCount="10" promptText="-40.0"
          GridPane.rowIndex="1" GridPane.columnIndex="2" />

<padding>
    <Insets top="10" right="10" bottom="10" left="10"/>
</padding>
```

Controller – Java Code



```
package f2c.view;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;

public class F2CController {

    @FXML Button f2c;
    @FXML Button c2f;
    @FXML TextField f;
    @FXML TextField c;

    public void convert(ActionEvent e) {
        Button b = (Button)e.getSource();
        if (b == f2c) {
            float fval = Float.valueOf(f.getText());
            float cval = (fval-32)*5/9;
            c.setText(String.format("%5.1f", cval));
        } else {
            float cval = Float.valueOf(c.getText());
            float fval = cval*9/5+32;
            f.setText(String.format("%5.1f", fval));
        }
    }
}
```

@FXML directive links widget to fxml element:
var name in code = id in layout

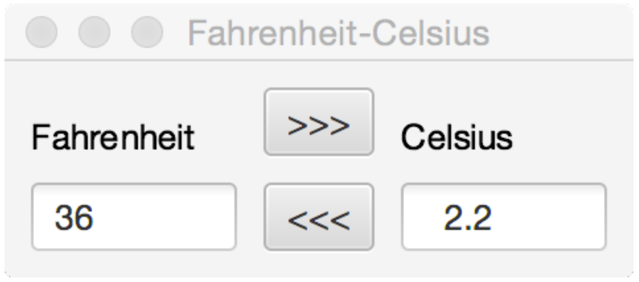
Name of method = name assigned in # directive in fxml file for onAction attribute

Main App for View/Controller

```
package f2c.app;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
...

public class F2CApp extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(getClass().getResource("/f2c/view/F2C.fxml"));

        GridPane root = (GridPane)loader.load();

        Scene scene = new Scene(root);
        ...
    }

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

Top-level layout tag in fxml file

Loading means creating objects for various widgets and layouts in the fxml file


Fahrenheit-Celsius
Fahrenheit
>>>
Celsius
36
<<<
2.2