Reminders

- An email has been broadcasted with grading scheme changes and dates for the completion of the remaining deliverables.
- Assn 4 is due March 23, 7pm.
- Quiz 3 March 26 – to be confirmed.
- Assn 5 due last day of term – April 3, 7pm.
- Final Exam, written in onQ – date to be confirmed, but expect the same date as used by the old written exam schedule.

This Video

- JavaFX Scene Graph.
- Containers and Layout Managers, in General.
- BorderPane.

JavaFX Scene Graph

- Here is part of the existing Node hierarchy:

JavaFX Scene Graph, Cont.

- Left-hand branch are containers – used to contain and organize other controls.
- Right-hand branch are the controls.
- Usually a container type object will be the “root”.

JavaFX Scene Graph, Cont.

- There are many more classes extending nodes in this hierarchy. For example, some classes extending from Labeled:

Aside - JavaFX API Docs

- JavaFX 14 API docs are linked to:
  https://openjfx.io/javadoc/14/
- For more information on JavaFX from the Oracle site:
  https://www.oracle.com/technetwork/java/javase/documentation/javafx-docs-2159875.html
JavaFX Scene Graph, Cont.
• Typically you build your GUI by first instantiating existing node objects. For example,

```
Label myLabel = new Label("Hello!");
```

• In order to display the node in the window, you need to add the object you instantiated to the existing Scene Graph structure.

Aside – Generating Import Statements
• Eclipse is great at helping you generate a missing import statement if you try to use a class that you have not yet imported.

• **But:** if you have a choice, make sure to choose the javafx one, rather than the class from javax.swing or java.awt.

Pane Objects
• In order to design your window you need to understand how the different Pane objects work.

• From the JavaFX API Tree:

```
    Pane
      - AnchorPane
      - BorderPane
      - DialogPane
      - FlowPane
      - GridPane
      - HBox
      - PopupControl.CSSBridge
      - StackPane
      - TextPane
      - TilePane
      - VBox
```

Aside – Layout Managers
• Each pane is associated with a different layout manager – an algorithm that determines where controls are placed within the pane and often their size.

• Provides flexibility during design and during runtime when the size of the window is changed or the window is viewed on a different OS (or even just a different monitor resolution!) than the one on which it was designed.

Pane Objects, Cont.
• **Won't discuss:**

• DialogPane – Used as the base container for customized dialog windows. We can just use the built-in dialogs. We don't need to design our own.

• PopupControl.CSSBridge – Seems to be mostly for internal use.

• StackPane and TilePane.
Layout Managers – A Reference

• A Nice Reference – “Working with Layouts in JavaFX”:
  http://docs.oracle.com/javafx/2/layout/jfxpub-layout.htm

Aside – Absolute Positioning

• Used by Microsoft’s Visual Studio .NET, for example.
• Control positions and sizes (in pixels) are all absolute. The visual editor helps you assign default values and to line up controls.
• But .NET is designed to run on one OS only...
• You can still use absolute positioning in JavaFX, but you are giving away some convenience.

BorderPane

• BorderPane lays out components in top, left, right, bottom, and center positions:

BorderPane, Cont.

• For example, to add a Button to the Top position:
  BorderPane bPane = new BorderPane();
  Button myButton = new Button("Click Me!");
  bPane.setTop(myButton);
• A component adopts its preferred size if it is smaller than the available region.
• Center swells to occupy neighbouring positions if they are empty.

BorderPane, Cont.

• You can add margins around a component by using an Insets object with the static .setMargin() method.
• You can also specify alignment within a position using a value from the Pos enum and the static .setAlignment() method.
• This layout is good for blocking out an entire window into sub-regions and then you use another layout in the regions of interest.

BorderPane, Cont.

• See FXProjectBorderPane
• Test control positioning, alignment and sizing.
• Setting individual control properties one at a time will be painful, especially when the properties are all being set to the same value...
Aside – Running JavaFX Lecture Notes Code

- Create a blank JavaFX project.
- Remove any files in the application folder.
- In a file browser, unzip the archive (*.zip) file to the application folder of the project.

- At the moment this is just Main.java and some also have application.css.
- Back in eclipse, use <F5> to refresh the project listing.

Aside - Component Sizing

- Unless you define a preferred size the calculated size is based on the size of the rendered label plus a default padding size between the label and the border.
- You can also change the size of the margin in code.