Prifysgol Abertawe
Swansea University

CS-230 Software Engineering

L13: JavaFX – Part 2

Dr. Liam O’Reilly

Semester 1 – 2020

Recap – The Structure of a JavaFX Application

```
*** JavaFX Imports ***

public class Main extends Application {
    public void start(Stage primaryStage) {
        // Create a new pane to hold our GUI
        Pane root = ...

        // Build a GUI

        // Create a scene based on the pane.
        Scene scene = new Scene(root, 400, 400);

        // Show the scene
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

Loads of JavaFX Imports – You really need to use an IDE to help you.

You must extend application and override the start method. You will be passed a Stage object.

Build up your Scene.

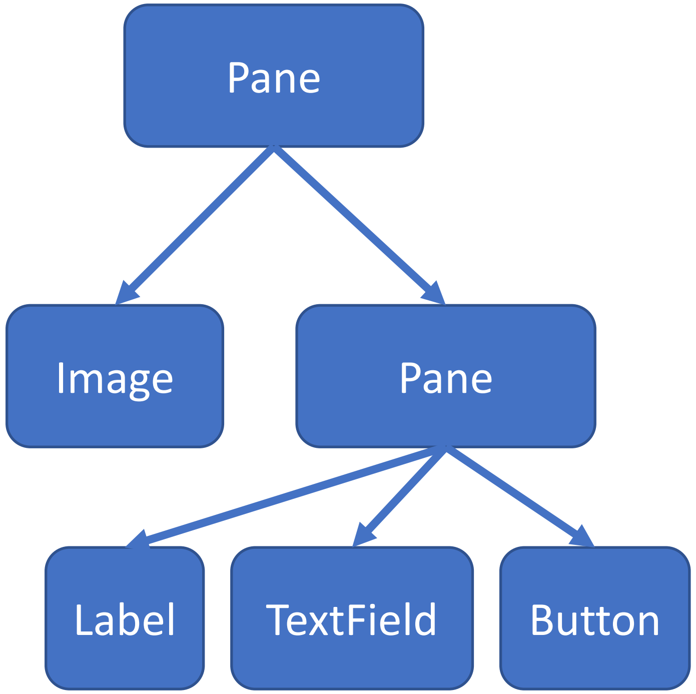
Set it to be on stage.

And show the stage.

Main method just kicks off the JavaFX system.

Recap – Scene Graphs

- JavaFX uses Scene Graphs.
- Each node is a visual/graphical object. E.g.,
  - **Geometrical (Graphical) objects** – (2D and 3D) such as circle, rectangle, polygon, etc.
  - **UI controls** – such as Button, Checkbox, Choice box, Text Area, etc.
  - **Containers** – (layout panes) such as Border Pane, Grid Pane, Flow Pane, etc.
    - These can contain more nodes as children.
  - **Media elements** – such as audio, video and image objects.



- The tree is drawn to the screen using an in order traversal.
- Transformations to parent nodes affect child node.

JavaFX Version History

- JavaFX 1.0
- JavaFX 1.1
- JavaFX 1.2
- JavaFX 1.3
- JavaFX 1.3.1
- JavaFX 2.0
- JavaFX 2.1
- JavaFX 2.2
- JavaFX 8
- JavaFX 9

JavaFX 2.2 and earlier are completely out-of-date.

Only look at documentation and tutorials based on JavaFX8 and onwards

Laying Out GUIs

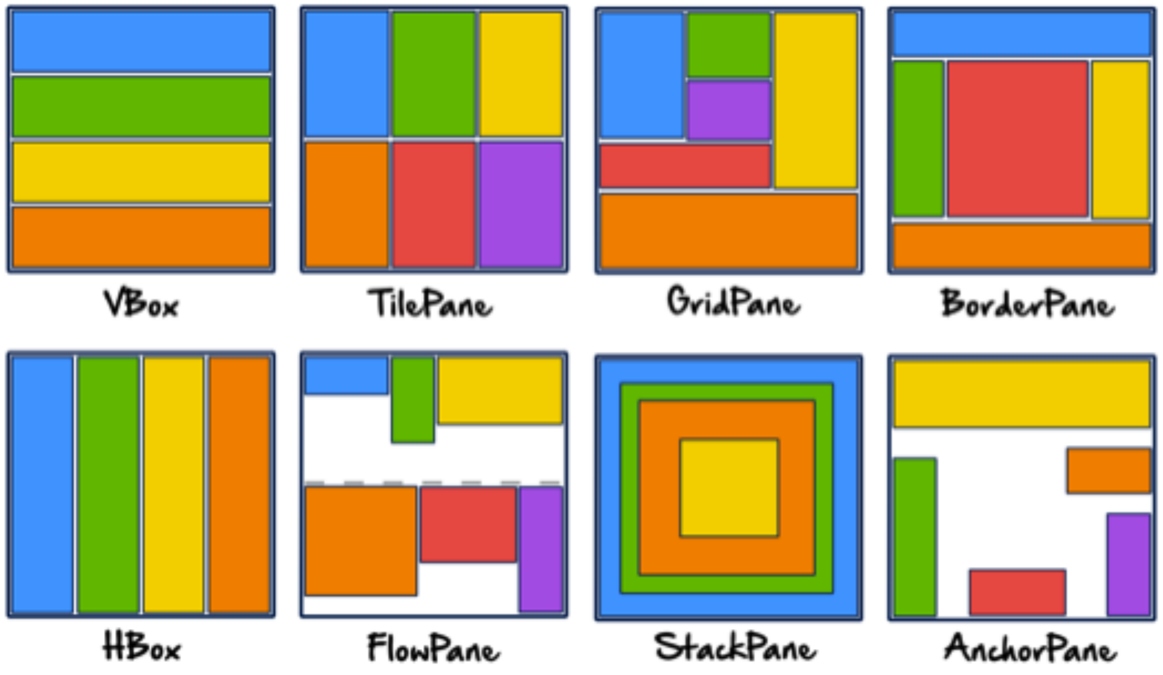
Layout Management

- Arranging components on the screen.
- User-interface components are arranged by placing them inside container nodes.
- There are a variety of “Pane” nodes that each layout their children in different ways.

- Useful Websites / Tutorials:
  - JavaFX: https://docs.oracle.com/javase/8/javase-clienttechnologies.htm
  - JavaFX layouts: https://docs.oracle.com/javase/8/javafx/layout-tutorial/index.html

Layout Nodes

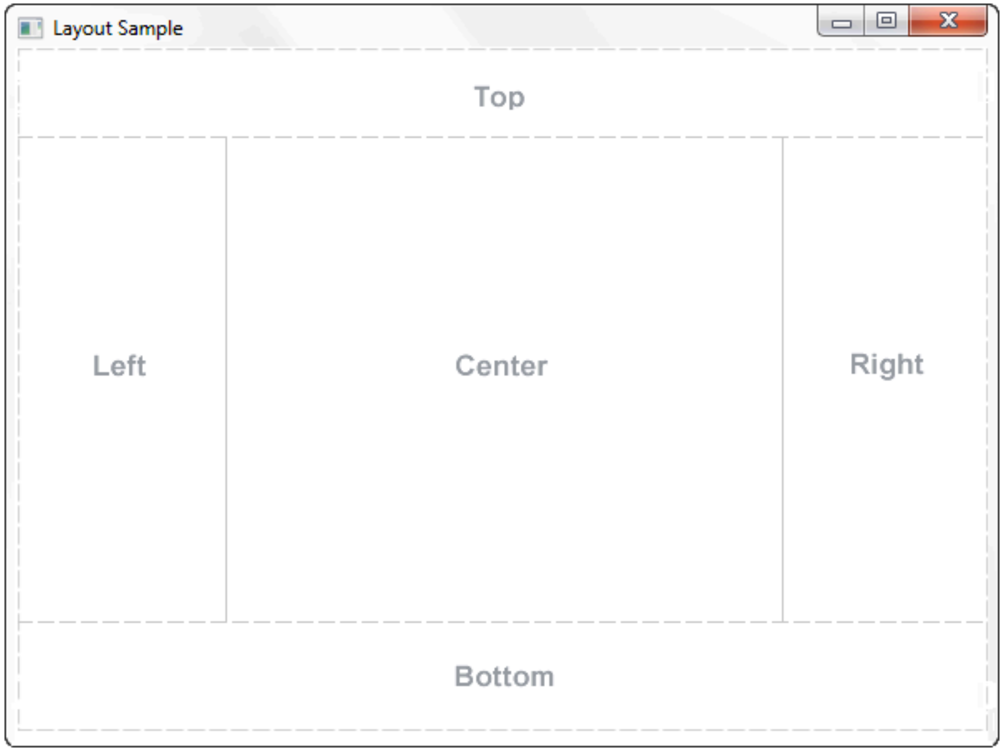
- Main Idea:
  - There are various panes that layout the children in different ways.
  - You must nest the panes in order to produce desirable layouts.



https://dzone.com/refcardz/javafx-8-1

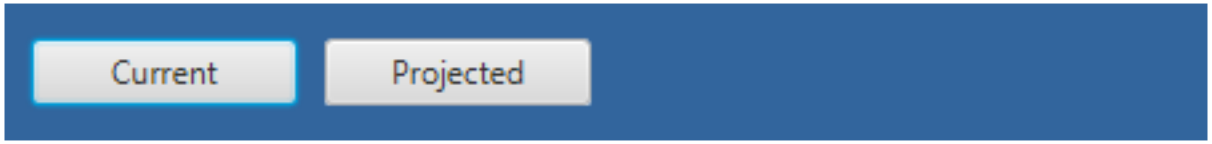
BorderPane

- BorderPane lays out its children in 5 regions.
- The edges will "shrink" to the minimum size required to fit the children.
- By default the center expands as needed (and resizes the center child).
- A border pane is useful for the classic look of a tool bar at the top, a status bar at the bottom, a navigation panel on the left, additional information on the right, and a working area in the center.

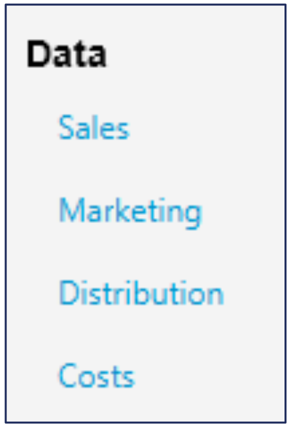


HBox & VBox

- The HBox layout pane lays out its children in a single row from left to right.



- The VBox layout pane arranges the children from top to bottom in a column.

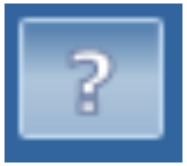


Note: This node has had some CSS styling applied.

StackPane

The StackPane layout pane places all of the nodes within a single stack with each new node added on top of the previous node.

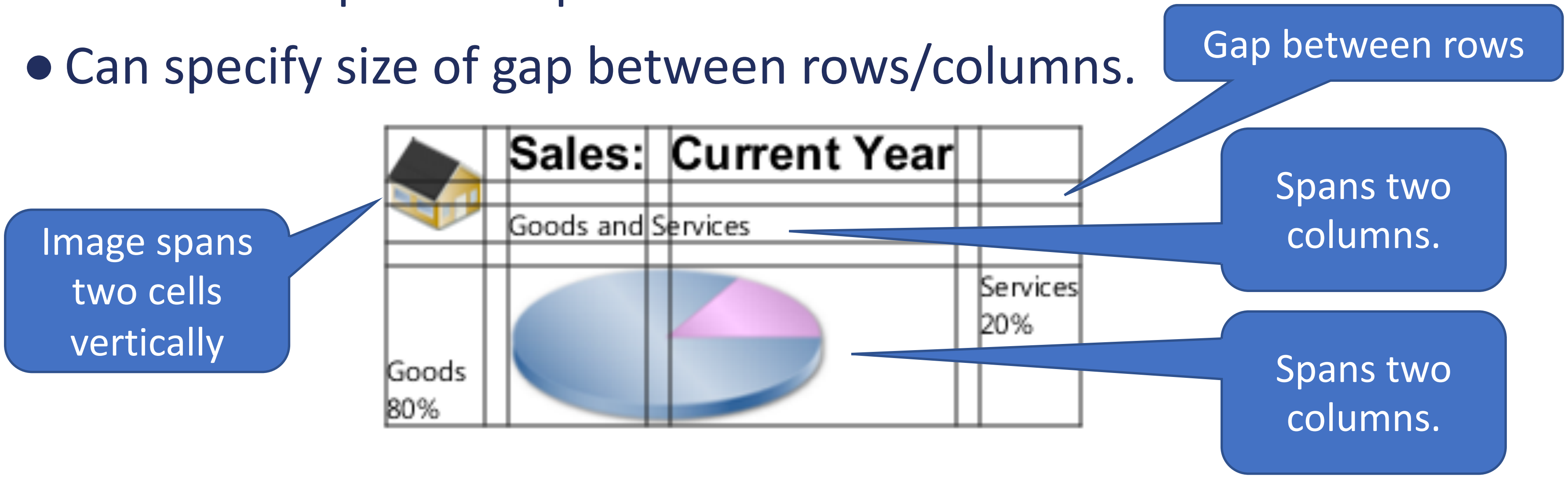
- Provides an easy way to overlay text on a shape or image or to overlap common shapes to create a complex shape.

- Example: 

  A help icon that is created by stacking a question mark on top of a rectangle with a gradient background.

GridPane

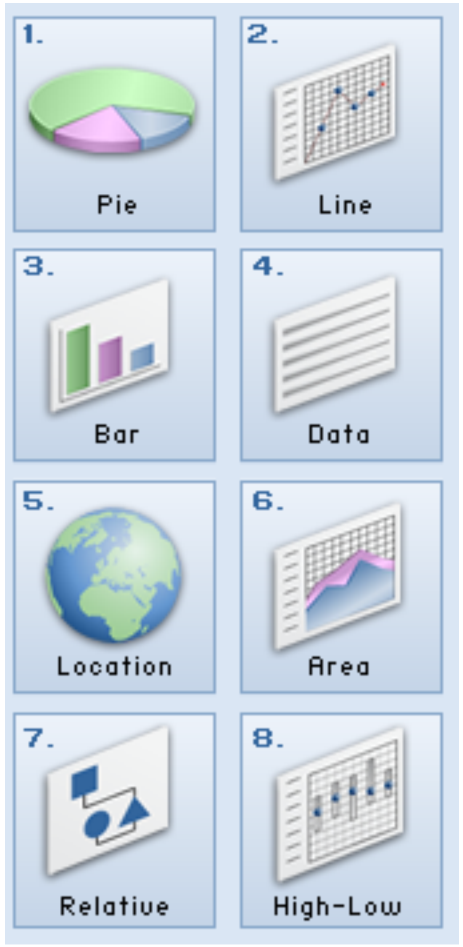
- GridPane provides flexible grid of rows and columns in which to lay out nodes.
- Nodes can be placed in any cell in the grid.
- Nodes can span multiple cells.
- Can specify size of gap between rows/columns.



- Useful for creating forms or any layout that is organised in rows and columns.
- Hint: Use the gridLinesVisible property to display grid lines. Useful for debugging.

FlowPane & TilePane

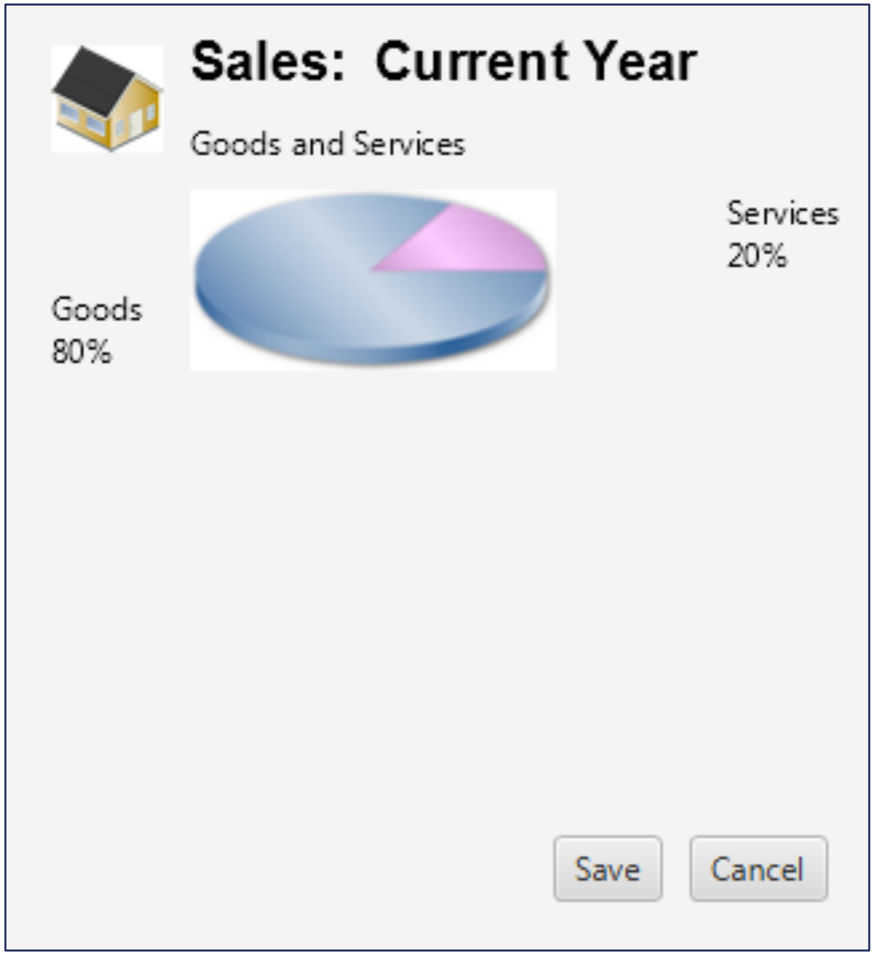
- FlowPane children laid out consecutively and wrap at the boundary.
- Nodes can flow vertically (in columns) or horizontally (in rows).



- A TilePane is similar to a FlowPane.
- It places all of the nodes in a grid in which each cell, or tile, is the same size.

AnchorPane

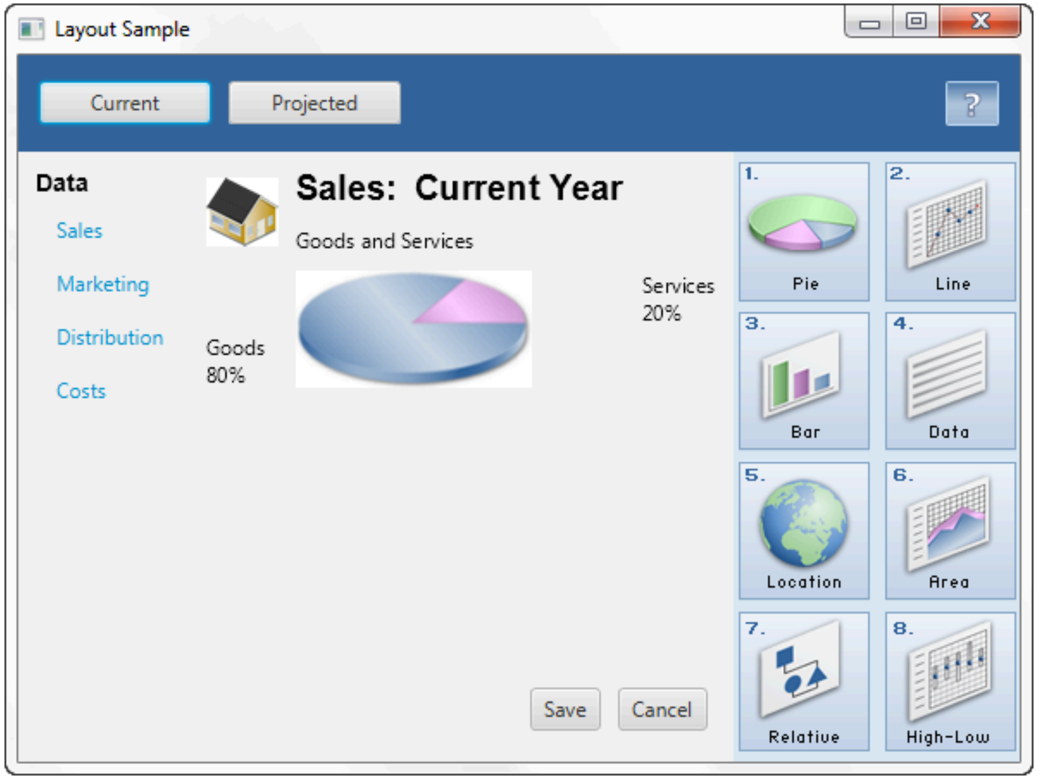
- The AnchorPane layout pane enables you to anchor nodes to the top, bottom, left side, right side, or center of the pane.
- As the window is resized, the nodes maintain their position relative to their anchor point.
- Example:
  An AnchorPane containing a GridPane anchored to the top and a HBox pane with two buttons anchored to the bottom and the right side.



Nesting

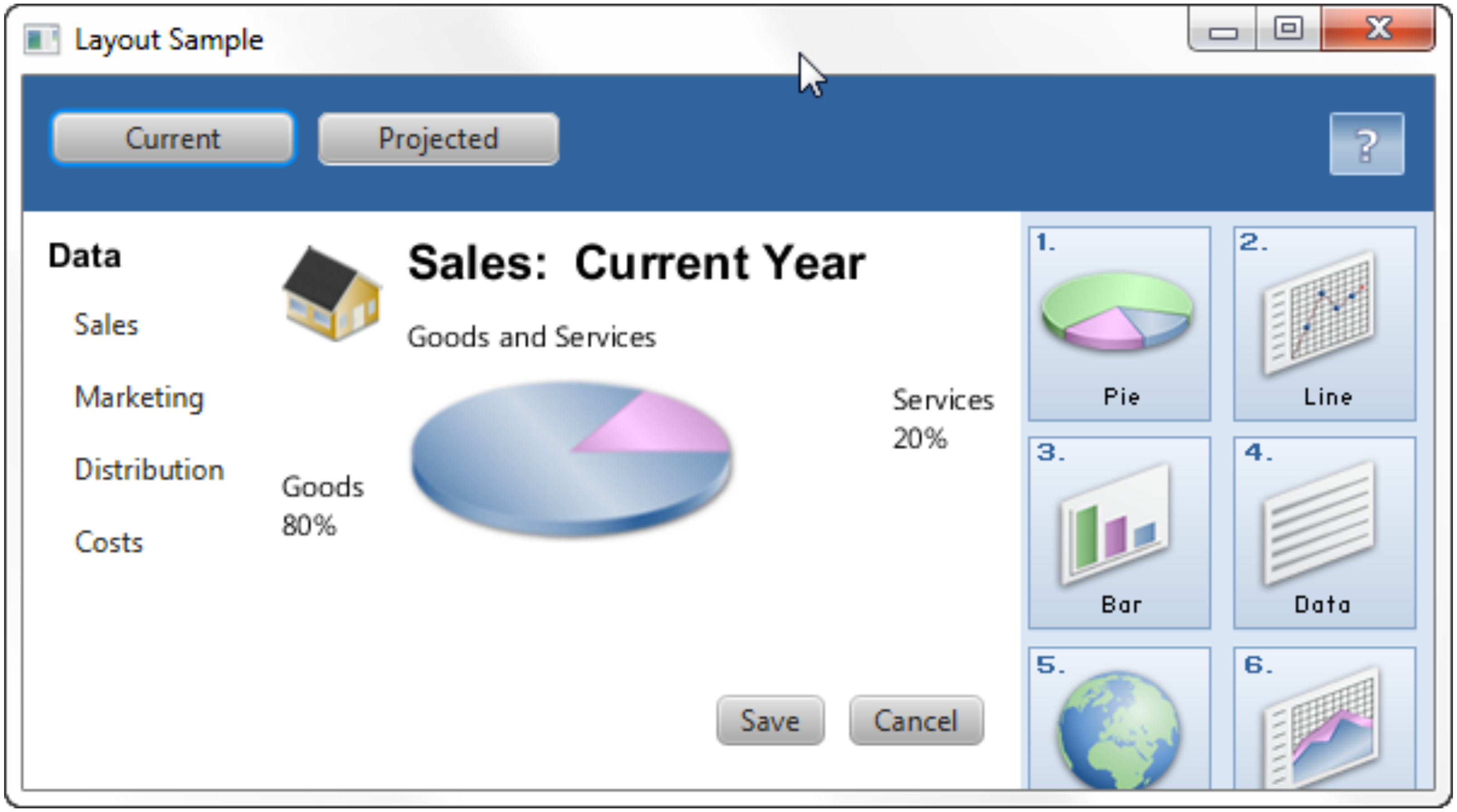
- Nest layouts within one another to create complex layouts.

- Example:
  Border Layout:
  - Top: HBox
  - Left: VBox
  - Right: FlowPane
  - Center:
    AnchorPane:
    - GridPane
    - HBox



Good Nesting Allows Good Resizing

- If you use AnchorPanes and BorderPanes wisely then you can create well-designed, resizeable layouts.



FXML

What is FXML?

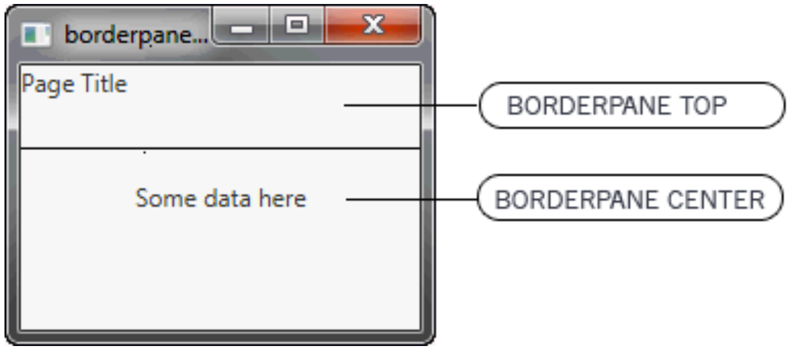
- FXML is an XML-based language.
- You specify the structure of the user interface (i.e., Scene Graph) in a separate xml file.
- Thus, you separate the code of building the GUI from the application logic.
    - Separation of the presentation and application is generally accepted to be good practice.
- Uses Controllers to handle events.
    - Based on a Design Pattern called Model-View-Controller.

Resources

- Useful Websites / Tutorials:
  - JavaFX: https://docs.oracle.com/javase/8/javase-clienttechnologies.htm
  - FXML: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/fxml_tutorial.htm

    https://docs.oracle.com/javase/8/javafx/fxml-tutorial/

Simple Example



JavaFX without FXML

```
BorderPane border = new BorderPane();

Label topPaneText = new Label("Page Title");
border.setTop(topPaneText);

Label centerPaneText = new Label ("Some data here");
border.setCenter(centerPaneText);
```

FXML

```
*** Imports go here ***
<BorderPane>
   <top>
      <Label text="Page Title"/>
   </top>
   <center>
      <Label text="Some data here"/>
   </center>
</BorderPane>
```

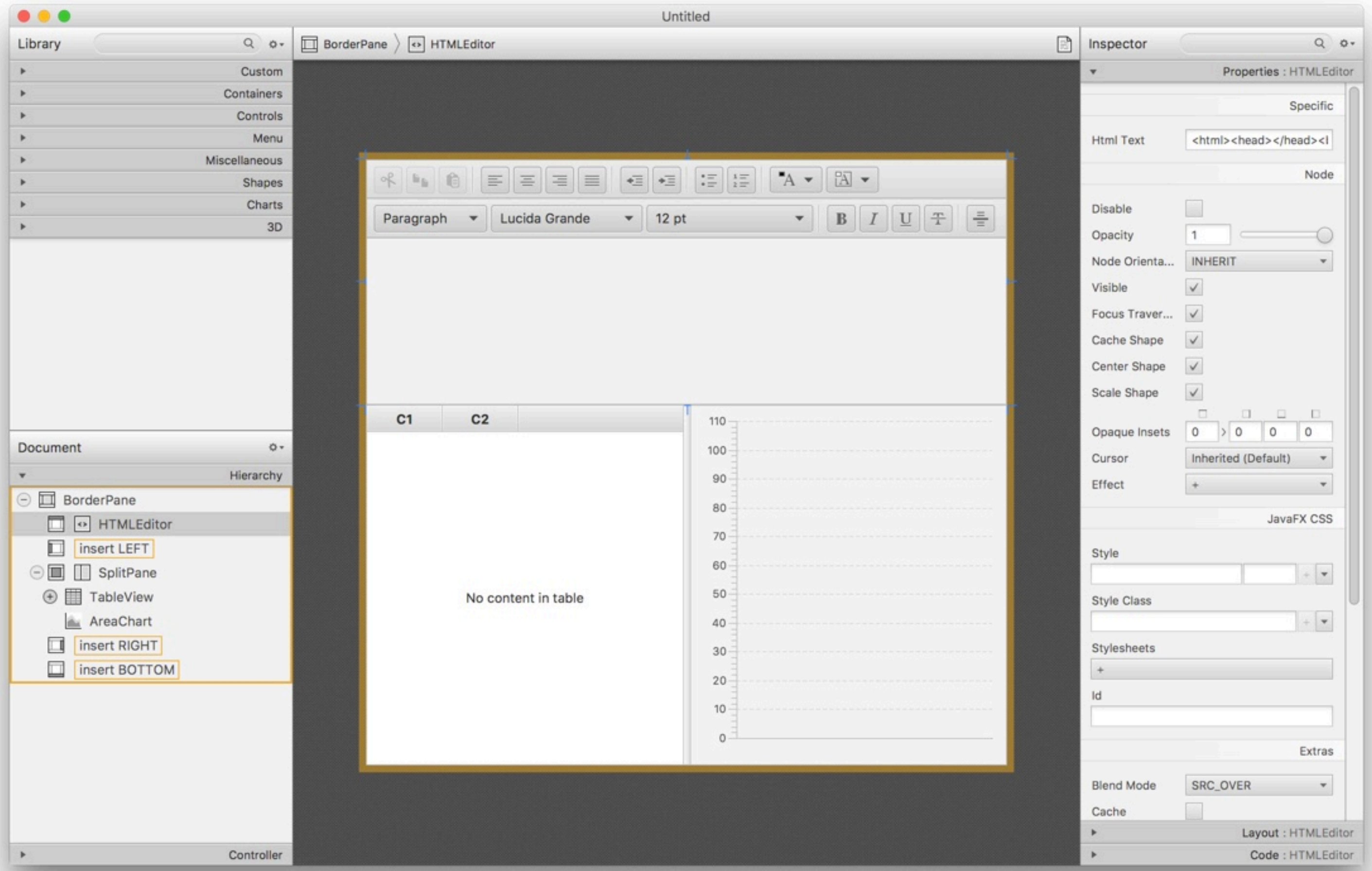
In FXML we specify the scene graph that JavaFX will use in a separate xml file.

Benefits of FXML

- XML file contains less text – so easier to ‘see’ the GUI design.
  - Easier to read.
  - Easier to maintain and change.
- FXML is not a compiled language; you do not need to recompile the code to see the changes.
- Easier localisation of GUI. Localise content of an FXML file as it is read
- You can use FXML with any Java Virtual Machine (JVM) language, such as Java, Scala, or Clojure.
  - For this course we will only use Java.
- Can create GUI using Scene Builder tool. This outputs the FXML file.

SceneBuilder

- Scene Builder (provided by Gluon here because Oracle only ships it in source code form).

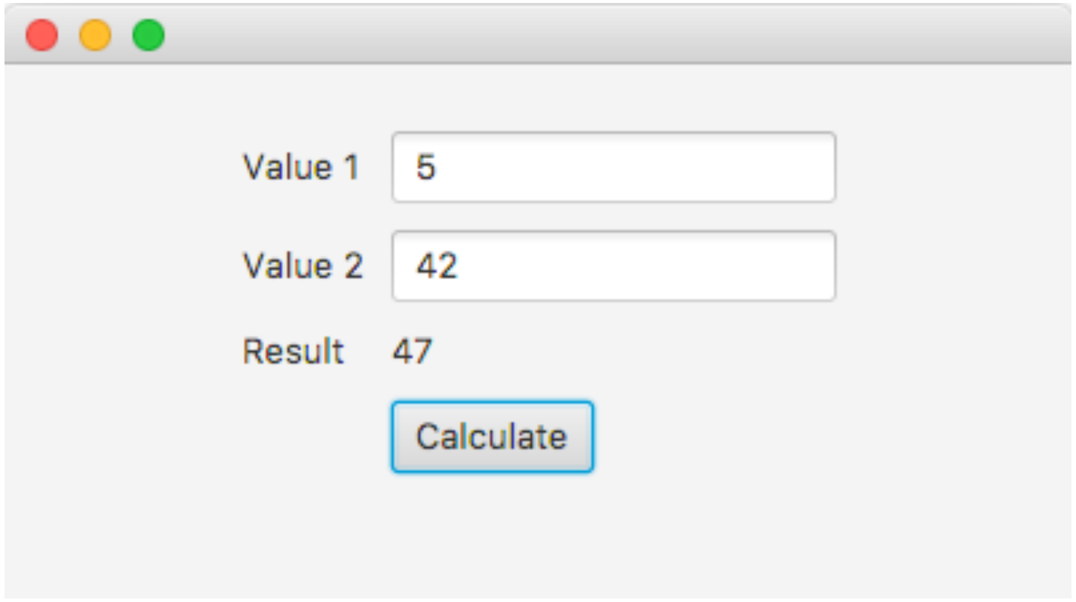


Html Text: <html><head></head><l
Disable: ☐
Opacity: 1
Node Orienta...: INHERIT
Visible: ✓
Focus Traver...: ✓
Cache Shape: ✓
Center Shape: ✓
Scale Shape: ✓
Opaque Insets: 0 0 0 0
Cursor: Inherited (Default)
Effect: +
Blend Mode: SRC_OVER
Cache: ☐

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

Calculator Example with FXML

We will build our (extremely poor) calculator again, but this time using FXML.


Value 1 5

Value 2 42

Result 47

Calculator Example with FXML – The FXML File

```
<?xml version="1.0" encoding="UTF-8"?>
<?import java.net.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

<VBox fx:controller="CalculatorController" xmlns=http://javafx.com/javafx/8
    xmlns:fx="http://javafx.com/fxml/1">
  <GridPane alignment="center" hgap="10" vgap="10">
    <padding><Insets top="25" right="25" bottom="10" left="25"/></padding>

    <Label text="Value 1" GridPane.columnIndex="0" GridPane.rowIndex="0" />
    <TextField fx:id="value1Box" GridPane.columnIndex="1" GridPane.rowIndex="0" />

    <Label text="Value 2" GridPane.columnIndex="0" GridPane.rowIndex="1" />
    <TextField fx:id="value2Box" GridPane.columnIndex="1" GridPane.rowIndex="1" />

    <Label text="Result" GridPane.columnIndex="0" GridPane.rowIndex="2" />
    <Label fx:id="resultBox" GridPane.columnIndex="1" GridPane.rowIndex="2" />

    <Button onAction="#handleCalculateButtonAction" text="Calculate”
      GridPane.columnIndex="1" GridPane.rowIndex="3" />
</GridPane></VBox>
```

Standard XML line

FXML imports for nodes used

Controller class will handle the functionality of this class

Padding adds some gap around GridPane

Fx:id – id to link Java variable to

OnAction - Method to invoke in controller when button is clicked

Calculator Example with FXML – Main Class

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.fxml.FXMLLoader;

public class Main extends Application {
    public void start(Stage primaryStage) {
        try {
            Pane root = (Pane)FXMLLoader.load(getClass().
                                getResource("Calculator.fxml"));
            Scene scene = new Scene(root,400,200);
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

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

Load the scene from the FXML file

The load can fail (e.g., syntax errors in the FXML file), so we must handle the exception

Calculator Example with FXML – Controller Class

```
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;

public class CalculatorController {

    @FXML private TextField value1Box;
    @FXML private TextField value2Box;
    @FXML private Label resultBox;

    @FXML
    private void handleCalculateButtonAction(ActionEvent event) {
        int x = Integer.parseInt(value1Box.getText());
        int y = Integer.parseInt(value2Box.getText());
        int z = x + y;
        resultBox.setText(Integer.toString(z));
    }
}
```

@FXML tag is used to bind variables and methods to the FXML scene.

Final Thoughts

- We have only touched the surface.
- You don’t have to use FXML.
  - You can just use plain JavaFX for A2.
- JavaFX / FXML is not hard, but you will need to lookup tutorials and documentation to use it.
- Spend some time playing with it before you seriously start trying to use it in A2.