# CS225L Lab 7: JavaFX

# Learning Outcomes

* ***Familiarization with browsing JavaFX documentation and Googling***
* Observe and modify behavior of GUI components
* Make stuff pretty using properties and CSS styles
* See that there are multiple ways to create the same visual style (e.g. setting a font)
* Acquire a taste of just a few of the other things that you can do to customize GUI components

# Pre-lab

JavaFX is the third generation of framework to let you create a graphical user interface (GUI) in Java. JavaFX is the successor of the Swing library, which was the successor to the Abstract Windows Toolkit (AWT). To create a JavaFX GUI, you must first create a class that extends javafx.application.Application, like so:

public class MyGUIClass extends Application { ...

//This will require you to implement a start method, which will look like this:

@Override

public void start ( Stage primaryStage ) {

// Start code here

}

The Stage object that is passed as an argument to the start method will be the foundation of your GUI. Your Stage will contain a Scene, which will contain a Pane. This pane is where you put your nodes, which are individual components (buttons, text fields, etc.). Suppose you have a simple button inside of a StackPane in your program. Your scene graph would look like this (Images from Oracle):



and your actual GUI would look like this:



**Lab Activities**

**Exercise 1: Layouts**

1. Start by importing the lab files into Eclipse as an existing project.
2. Compile and run the program to verify that you've imported it correctly. Each of the tabs should display a scene using the indicated pane, and each pane should have a number of button nodes visible.
3. Examine the layout produced for each type of pane as you change the size of the application's stage (by dragging the lower right corner). How do each of the panes move the nodes inside of them as the size of the pane changes? What happens to the tabs at the top when the window shrinks?
4. Change the operation of your [**FlowPane**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/FlowPane.html) object to set the horizontal and vertical gaps to 20 by using one of the available constructors.
5. Change the operation of the grid pane to also set the horizontal and vertical gaps to 20. Can you use a constructor for [**GridPane**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/GridPane.html) to accomplish this, or do you need to use methods?
6. Verify the correct operation and appearance of the modified program.

**Exercise 2: Styles**

JavaFX allows you to easily create highly customizable interfaces. One of the ways that this is accomplished in JavaFX is by using CSS (Cascading Style Sheets). CSS values may be set using the setStyle(String value) method of any object that extends [**Node**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Node.html). You may read more about [**using CSS values in JavaFX on the Oracle website**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html).

1. Add the following snippet to your code in the initLayoutsDemo method:

DropShadow dropShadow = new DropShadow ();

dropShadow.setRadius (5.0);

dropShadow.setOffsetX (3.0);

dropShadow.setOffsetY (3.0);

dropShadow.setColor ( Color.color (0.4 , 0.5 , 0.5));

borderBt3.setEffect ( dropShadow );

(Further reading: [**DropShadow**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/effect/DropShadow.html), [**Color**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/paint/Color.html))

***Note about imports***: Make sure that you import the classes from packages starting with javafx, like javafx.scene.effect.DropShadow and javafx.scene.paint.Color. If you import classes from the awt package, you will run into compiler issues about type mismatches and methods not being defined.

Make sure that you add the snippet after the borderBt3 object is instantiated (i.e. anywhere after borderBt3 = new Button("Button 3");). Why is this necessary?

Run your code. How does this change the appearance of the left button in the border pane?

1. Using a CSS value, give the fourth button in the tile pane a red background (use property z-fx-background-color).
2. Change the fonts:
   1. Using the setFont(...) method, set the font of the second button on the FlowPane to Comic Sans MS, size 16, using a new Font object. Again, make sure you use [**javafx.scene.text.Font**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/text/Font.html) instead of java.awt.Font!
   2. Using the setFont(...) method, set the font of the third button on the FlowPane to the same font in step 3a, using the Font.font(...) method.
   3. Using a CSS value, set the font of the fourth button on the FlowPane to the same font in step 3a. Hint: Use either the -fx-font-family and -fx-font-size properties, or just the -fx-font property. When specifying a specific font for the font family, do you need to put the font name in quotes?
3. Verify the correct operation and appearance of the modified program.

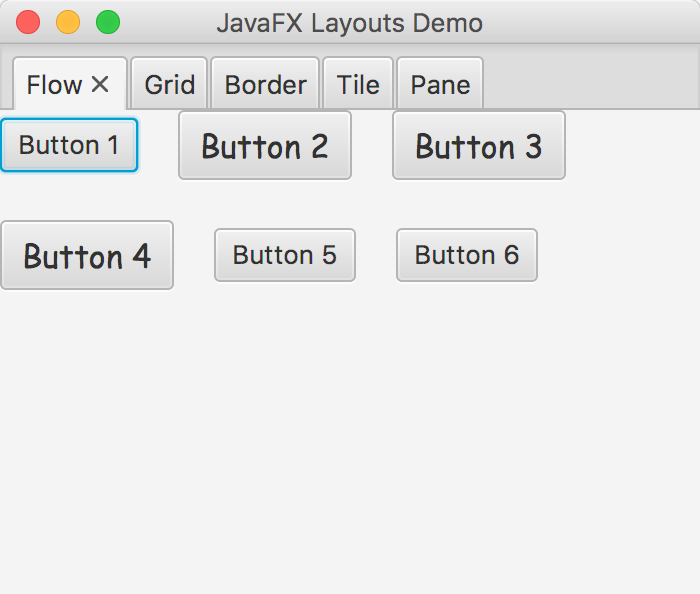
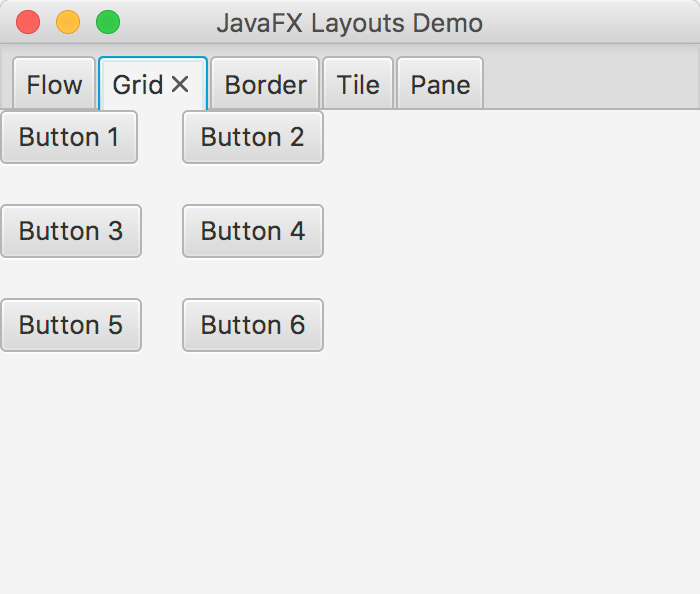
**Exercise 3: More GUI Fun**

Try out some other fun things!

Hints for 2 & 3 are in the footnotes of this page.

1. Shapes: Spice up the Pane by adding some shapes to it. Some shapes include [**Circle**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Circle.html), [**Line**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Line.html), and [**Rectangle**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Rectangle.html). (for a complete list, see the sub-classes of [**Shape**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Shape.html)).
2. Rotation: Rotate the center button in the border pane by .[[1]](#footnote-1)
3. [**Cursors**](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Cursor.html): Modify the tilePane object such that the cursor becomes a crosshair when it is in the pane.[[2]](#footnote-2) re
4. Verify the correct operation and appearance of the modified program.

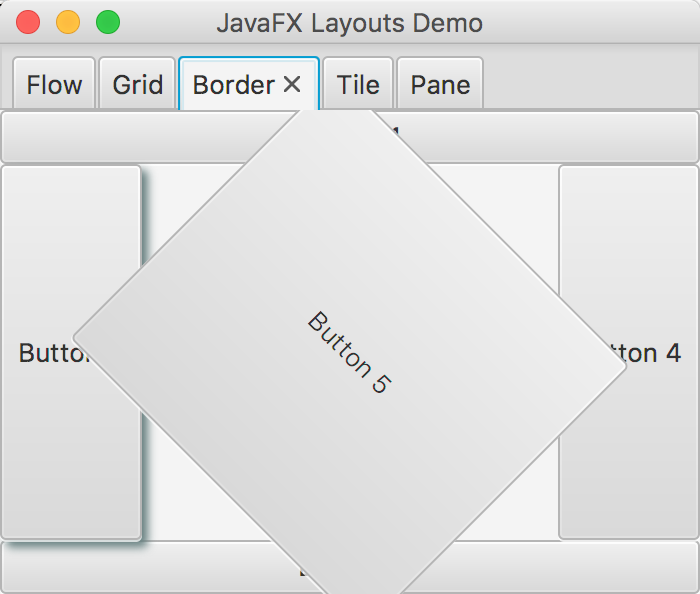
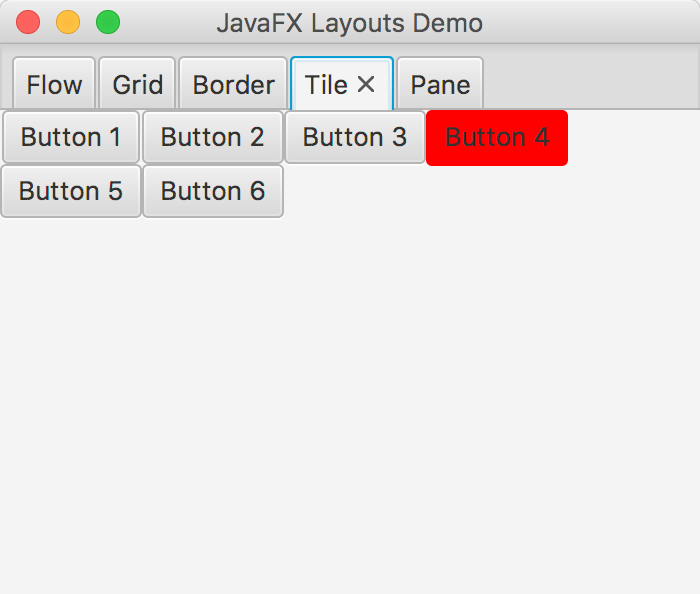
**Example Outputs:**

** **

1.5 vGap and hGap = 20

1.4 vGap and hGap = 20

2.3 Font and size change

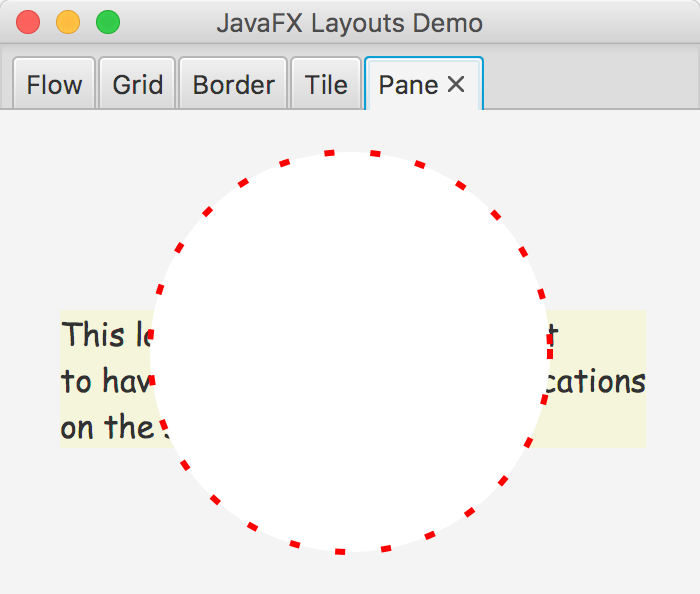
** **

2.2 Button 4 red background

3.3 crosshair cursor

2.1 Button 3 shadow

3.2 Rotation

****

3.1 Shapes drawn

1. (˙˙˙)ǝʇɐʇoɹʇǝs ǝs∩ [↑](#footnote-ref-1)
2. (˙˙˙)ɹosɹnƆʇǝs ǝs∩ [↑](#footnote-ref-2)