Programming of Interactive Systems

Anastasia.Bezerianos@lri.fr

JavaFX

JavaFX

Java + Flash + Flex

As with Java, it is cross-platform

Can use tools for interface building WYSIWYG (SceneBuilder, more later)

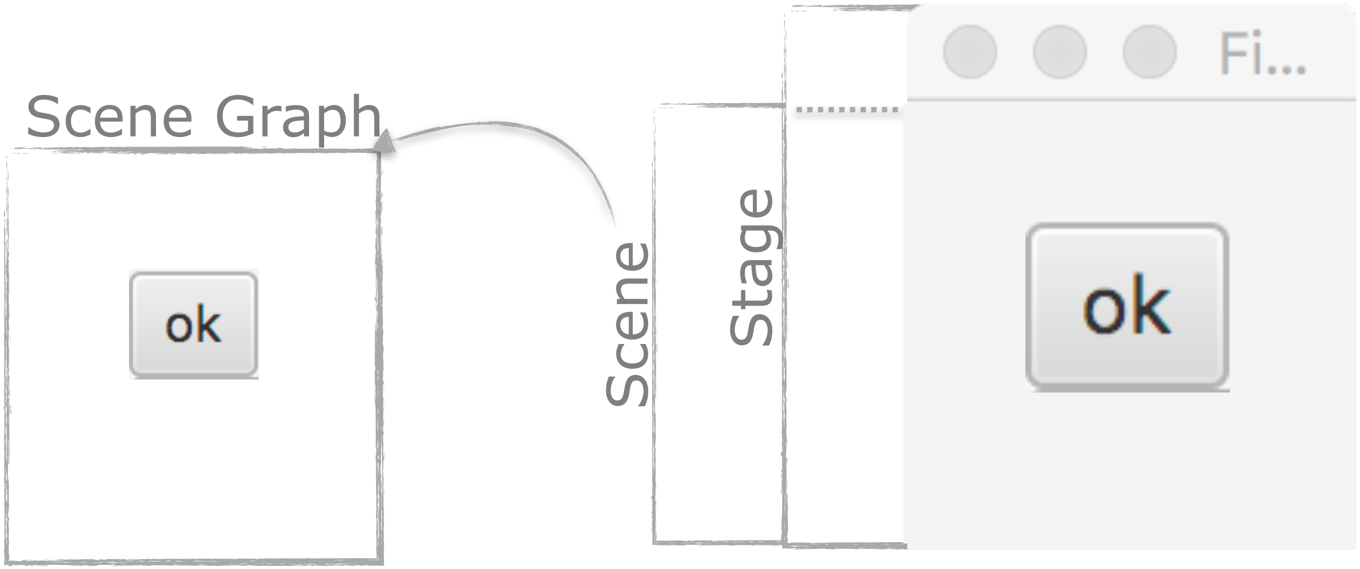
Supports advanced event handling (Swing/AWT)

CSS styling

Basic structure

Basic structure of a JavaFX program

- Application
- Override the `start(Stage)` method
- Stage ← Scene ← Nodes (Panes or Controls)



Application class

JavaFX programs include **one** class that extends Application (analogous to a single class with a main method for console programs).

javafx.application.Application

Application class

When running an Application class (a class that extends it), JavaFX does the following:

1. Constructs an instance of that Application class

2. Calls an `init()` method for application initialization
   ... don’t construct a Stage or Scene in `init()`

3. Calls the `start (javafx.stage.Stage)` method

4. Waits for the application to finish: either you call `Platform.exit()`, or the last window has been closed.

5. Calls the `stop()` method to release resources.
   init() and stop() have default do-nothing implementations.

Stage vs Scene

Stage

- represents windows, top level **container**
- many setter methods, e.g., `setTitle(), setWidth()`
- one stage is created by default by Application (ex primaryStage)
- you can have multiple stages and use (**set**) one or the other as your main stage (primaryStage in our example):

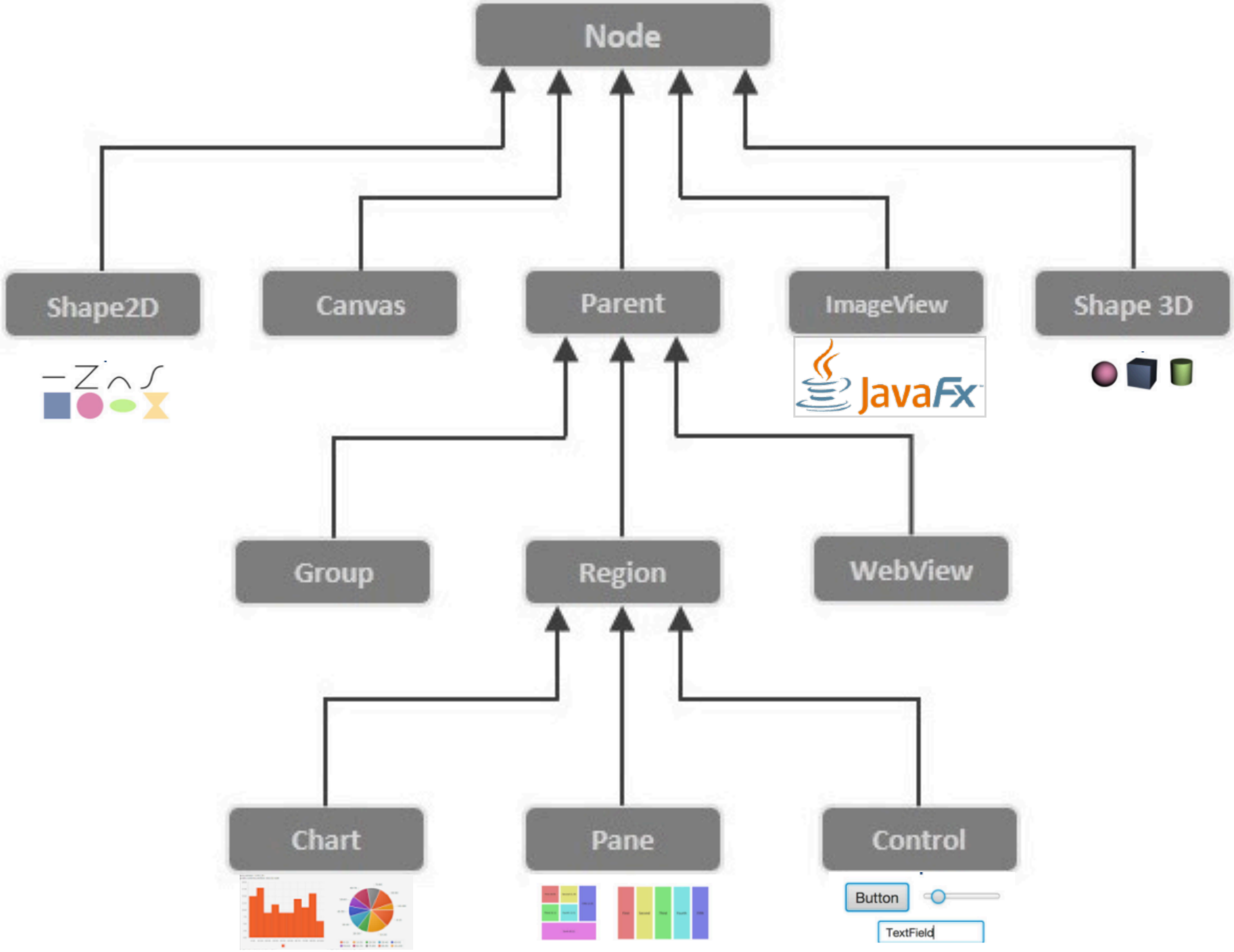
  construct a Stage for each window in your application,
  e.g., for dialogs and pop-ups.

Scene

- each stage has a scene (scene graph container)
- scenes hold controls (Buttons, Labels, etc.)
- you can put controls directly in scenes, or use **Panes** for better layout hierarchies:

  construct Scene(s) for collections of widgets you want to be
  grouped and visible together

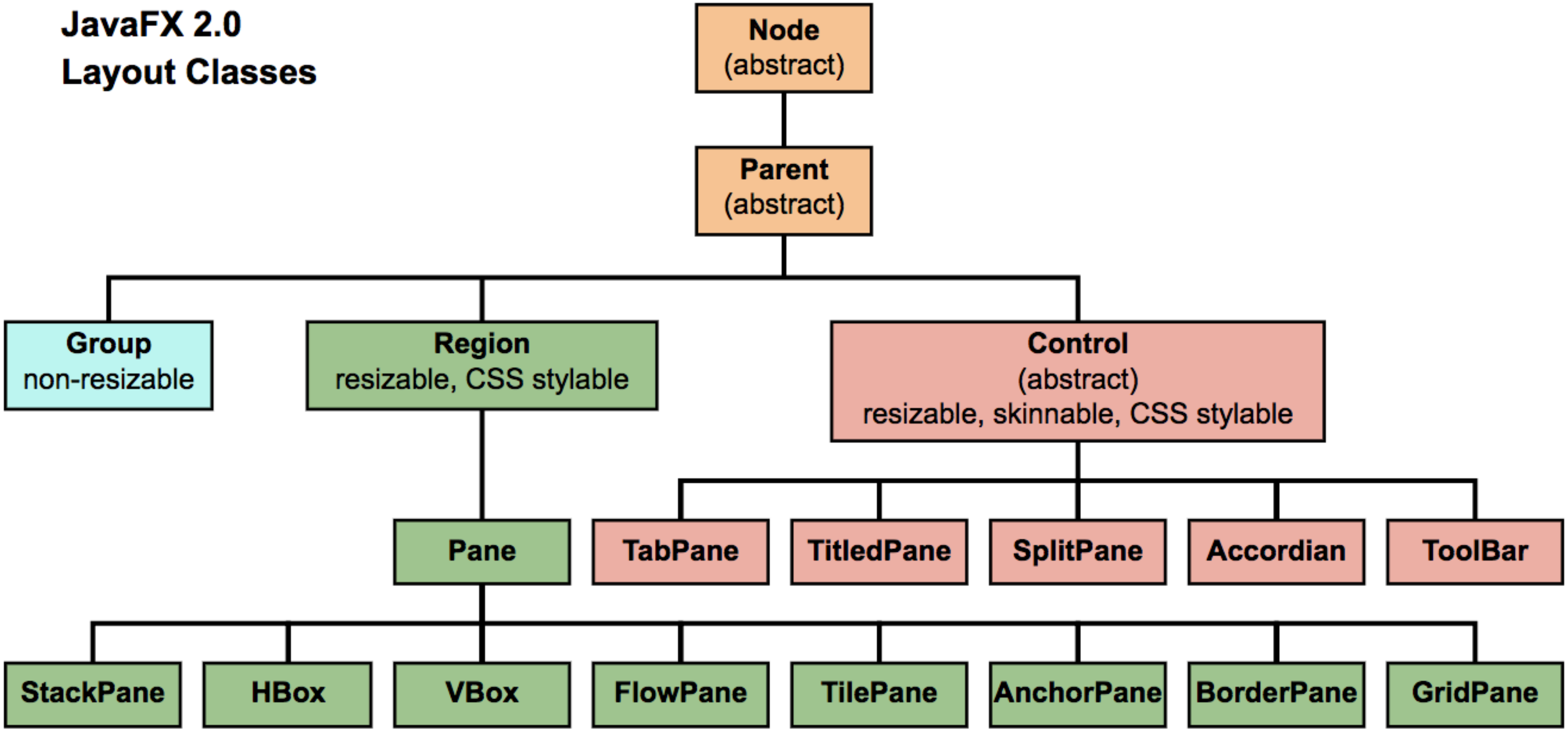
UI hierarchy



JavaFX layouts

JavaFX - complex structures

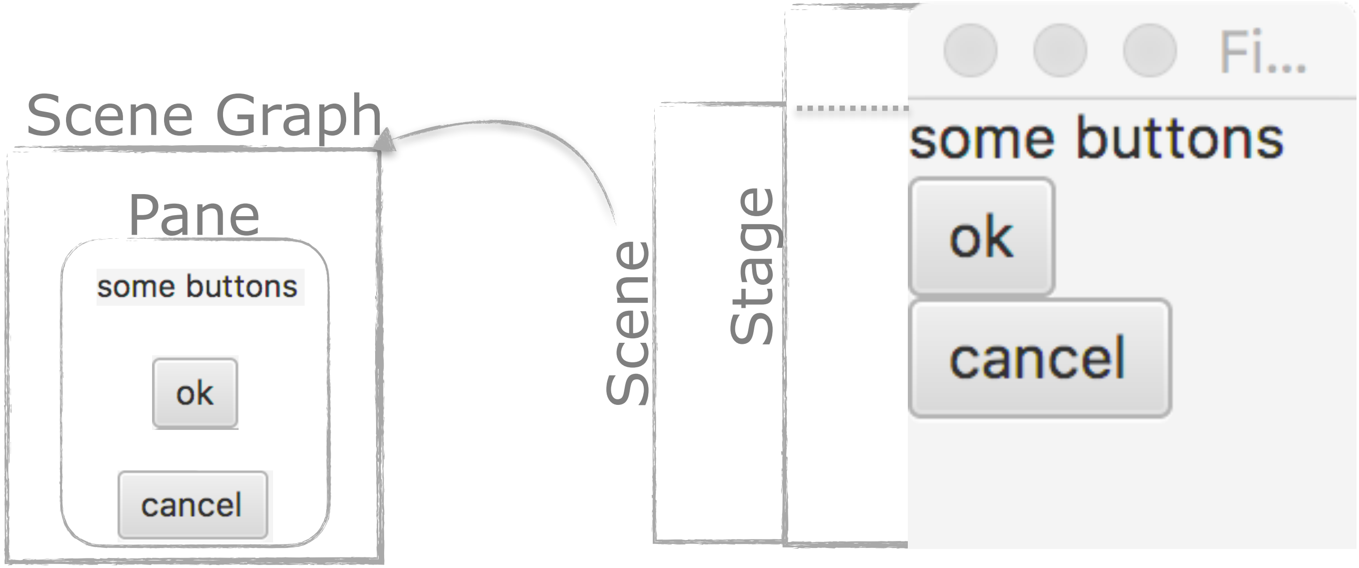
VBox is one of many *Pane* class objects
that help us organize nodes in a container



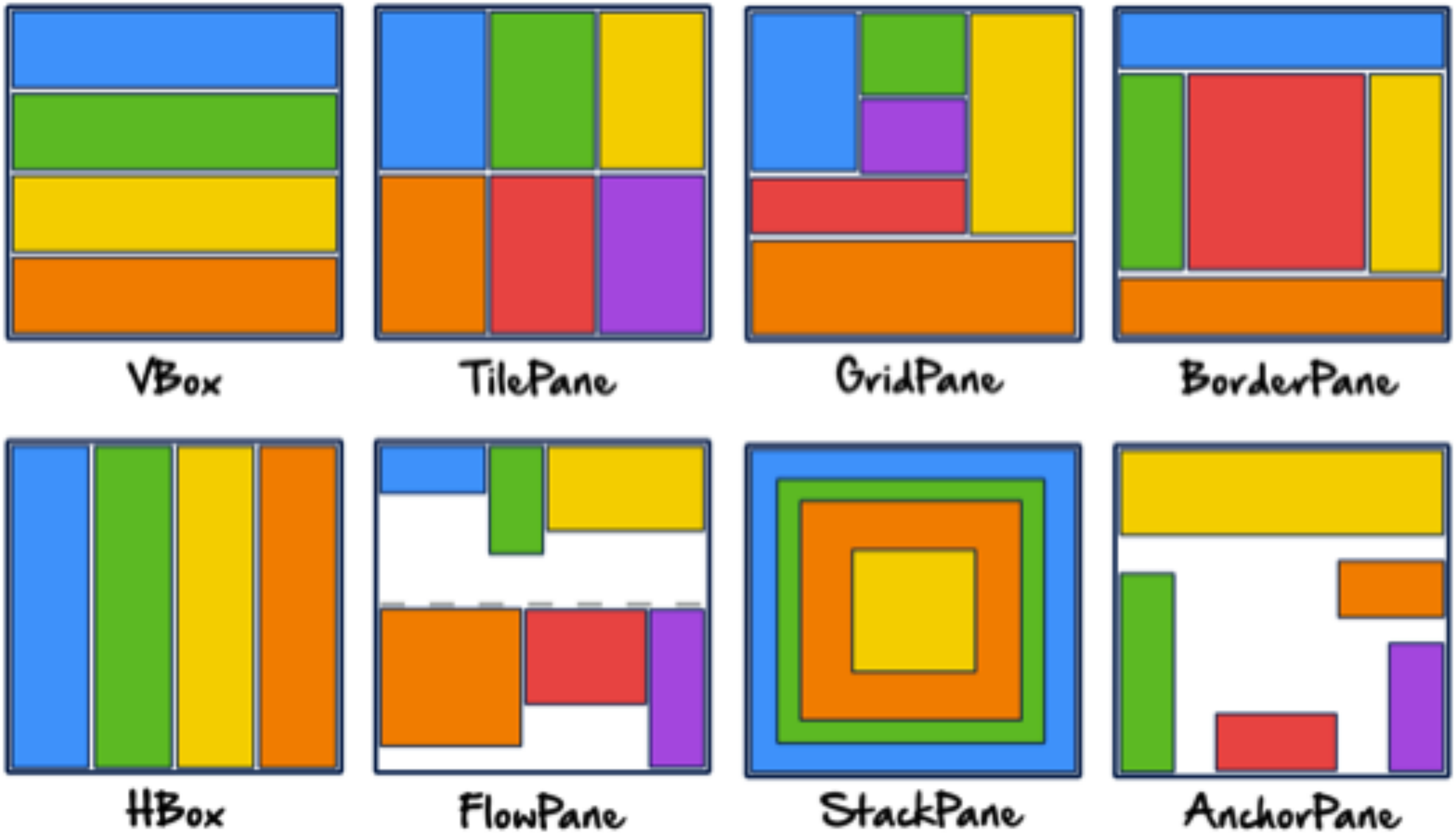
A more realistic structure

A more realistic structure of a JavaFX program

- Application
- Override the `start(Stage)` method
- Stage ← Scene ← Panes ← UI Nodes



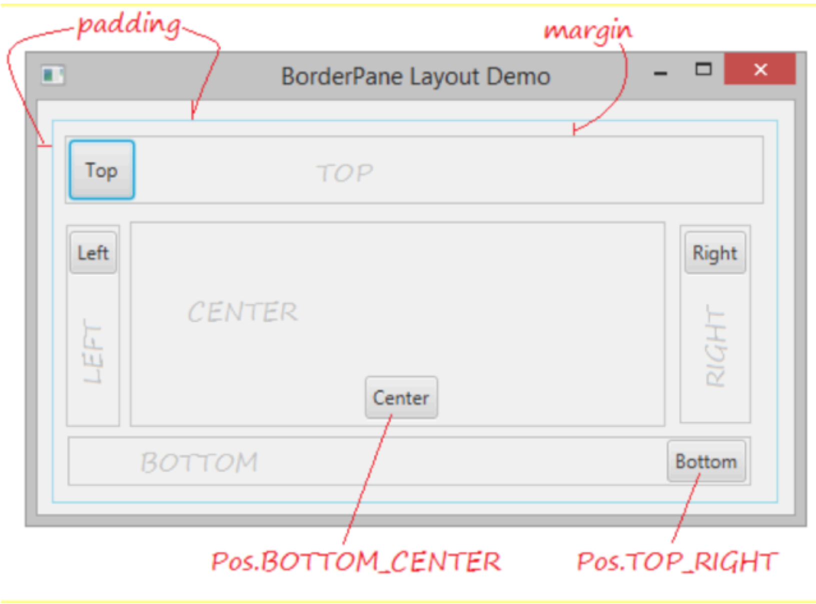
Panes for layout

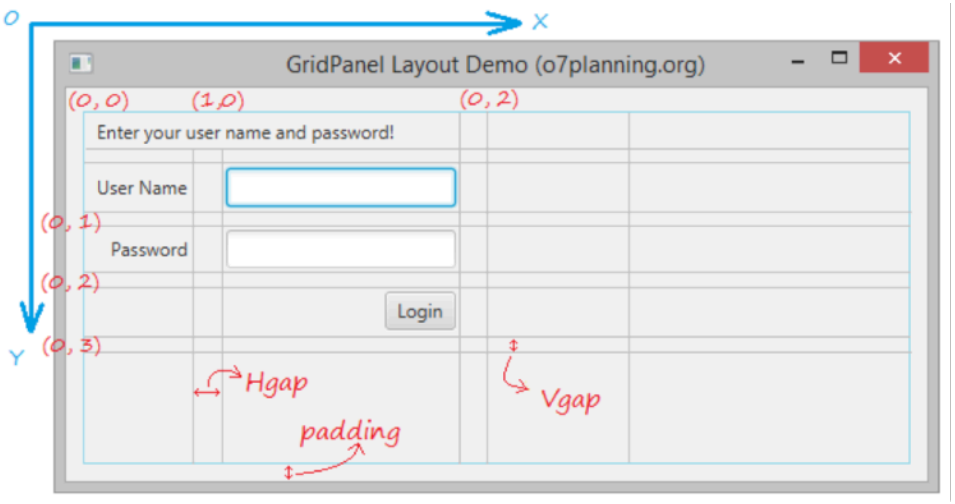


more tutorials at https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm
image from JavaFX 8 By Hendrik Ebbers & Michael Heinrichs

Improving layout

Layout Panes have different properties to help create layouts that persist during resizing (margin, padding, Vgap/Hgap, alignment)





Enter your user name and password!

https://o7planning.org/en/10629/javafx-borderpane-layout-tutorial

Panes for layout - examples

```
import javafx.application.Application;

public class CombinedLayouts extends Application{

     // main here …

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

         HBox hbox = new HBox();
         hbox.setPadding(new Insets(15, 12, 15, 12)); // padding all around
         hbox.setSpacing(10); // space between nodes
         hbox.setStyle("-fx-background-color: #336699;"); // familiar?

         Button buttonCurrent = new Button("Current");
         buttonCurrent.setPrefSize(100, 20); // preferred size

         Button buttonProjected = new Button("Projected");
         buttonProjected.setPrefSize(100, 20);

         hbox.getChildren().addAll(buttonCurrent, buttonProjected);

         BorderPane root = new BorderPane();
         root.setTop(hbox); // a Pane added to another Pane

         Scene scene = new Scene (root, 200, 200);
         primaryStage.setTitle("Complex Window!");
         primaryStage.setScene(scene);
         primaryStage.show();

     }
}
```

https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm and https://openjfx.io/javadoc/15/

JavaFX and CSS

Css of a single Node

```
import javafx.application.Application;

public class CombinedLayouts extends Application{

     // main here …

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

         HBox hbox = new HBox();
         hbox.setPadding(new Insets(15, 12, 15, 12)); // padding all around
         hbox.setSpacing(10); // space between nodes

         hbox.setStyle("-fx-background-color: #336699;"); // CSS of a single node

         Button buttonCurrent = new Button("Current");
         buttonCurrent.setPrefSize(100, 20); // preferred size

         Button buttonProjected = new Button("Projected");
         buttonProjected.setPrefSize(100, 20);

         hbox.getChildren().addAll(buttonCurrent, buttonProjected);

         BorderPane root = new BorderPane();
         root.setTop(hbox); // a Pane added to another Pane

         Scene scene = new Scene (root, 200, 200);
         primaryStage.setTitle("Complex Window!");
         primaryStage.setScene(scene);
         primaryStage.show();
     }
}
```

Simple CSS applied to one Node at a time

https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm and https://openjfx.io/javadoc/15/

Consistent design

Imagine we have one or more windows and decide we want to change their visual style everywhere …

CSS (cascading style sheets)

… it describes how HTML elements are to be displayed on screen, paper, or in other media …

… and saves a lot of work. It can control the layout of multiple scenes all at once

Consistent design

create a CSS file

give name mycss.css (do not convert your project!)

Inside the css add some styling properties:

```
.root {
     -fx-background-image: url("background.jpeg");
}

.label {
    -fx-font-size: 12px;
    -fx-font-weight: bold;
    -fx-text-fill: #333333;
    -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
.button {
    -fx-text-fill: white;
    -fx-font-family: "Arial Narrow";
    -fx-font-weight: bold;
    -fx-background-color: linear-gradient(#61a2b1, #2A5058);
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );
}

.button:hover {
    -fx-background-color: linear-gradient(#2A5058, #61a2b1);
}
```

https://docs.oracle.com/javafx/2/get_started/css.htm

Original Class (no CSS)

```
public class UseNoCSS extends Application {
     // main here …
     @Override
     public void start(Stage primaryStage) throws Exception {

          GridPane grid = new GridPane();
          grid.setAlignment(Pos.CENTER);
          grid.setHgap(10);
          grid.setVgap(10);
          grid.setPadding(new Insets(25, 25, 25, 25));

          Label userName = new Label("User Name:");
          grid.add(userName, 0, 1);

          TextField userTextField = new TextField();
          grid.add(userTextField, 1, 1);

          Label pw = new Label("Password:");
          grid.add(pw, 0, 2);

          PasswordField pwBox = new PasswordField();
          grid.add(pwBox, 1, 2);

          Button okBtn = new Button("ok");
          grid.add(okBtn, 1,3);

          Scene scene = new Scene(grid, 300, 275);

          primaryStage.setScene(scene);
          primaryStage.setTitle("Trying without CSS window!");
          primaryStage.show();
     }
}
```

https://docs.oracle.com/javafx/2/get_started/css.htm

Original Class + CSS

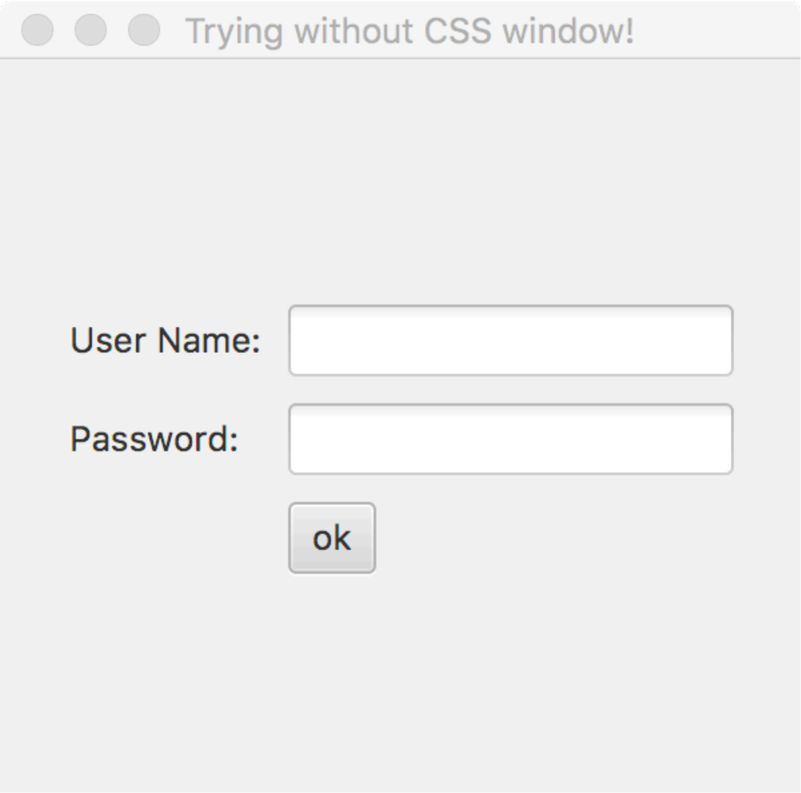
```
public class UseCSS extends Application {
      // main here …
      @Override
      public void start(Stage primaryStage) throws Exception {

           GridPane grid = new GridPane();
           grid.setAlignment(Pos.CENTER);
           grid.setHgap(10);
           grid.setVgap(10);
           grid.setPadding(new Insets(25, 25, 25, 25));

           Label userName = new Label("User Name:");
           grid.add(userName, 0, 1);

           TextField userTextField = new TextField();
           grid.add(userTextField, 1, 1);

           Label pw = new Label("Password:");
           grid.add(pw, 0, 2);

           PasswordField pwBox = new PasswordField();
           grid.add(pwBox, 1, 2);

           Button okBtn = new Button("ok");
           grid.add(okBtn, 1,3);

           Scene scene = new Scene(grid, 300, 275);

           scene.getStylesheets().add
            (UseCSS.class.getResource(“mycss.css").toExternalForm());
           // System looks for css in a location relative to where your main is

           primaryStage.setTitle("Trying with CSS");
           primaryStage.setScene(scene);
           primaryStage.show();
      }
}
```

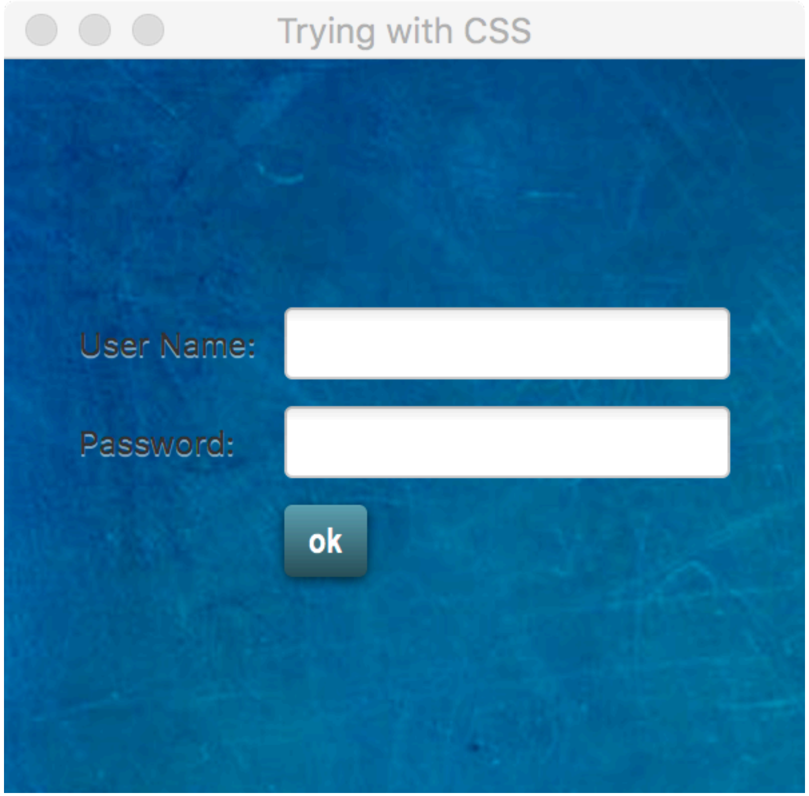
Simple CSS use
to apply basic styling:
just load CSS file

https://docs.oracle.com/javafx/2/get_started/css.htm

Consistent design using CSS

Simple way to apply style to all windows

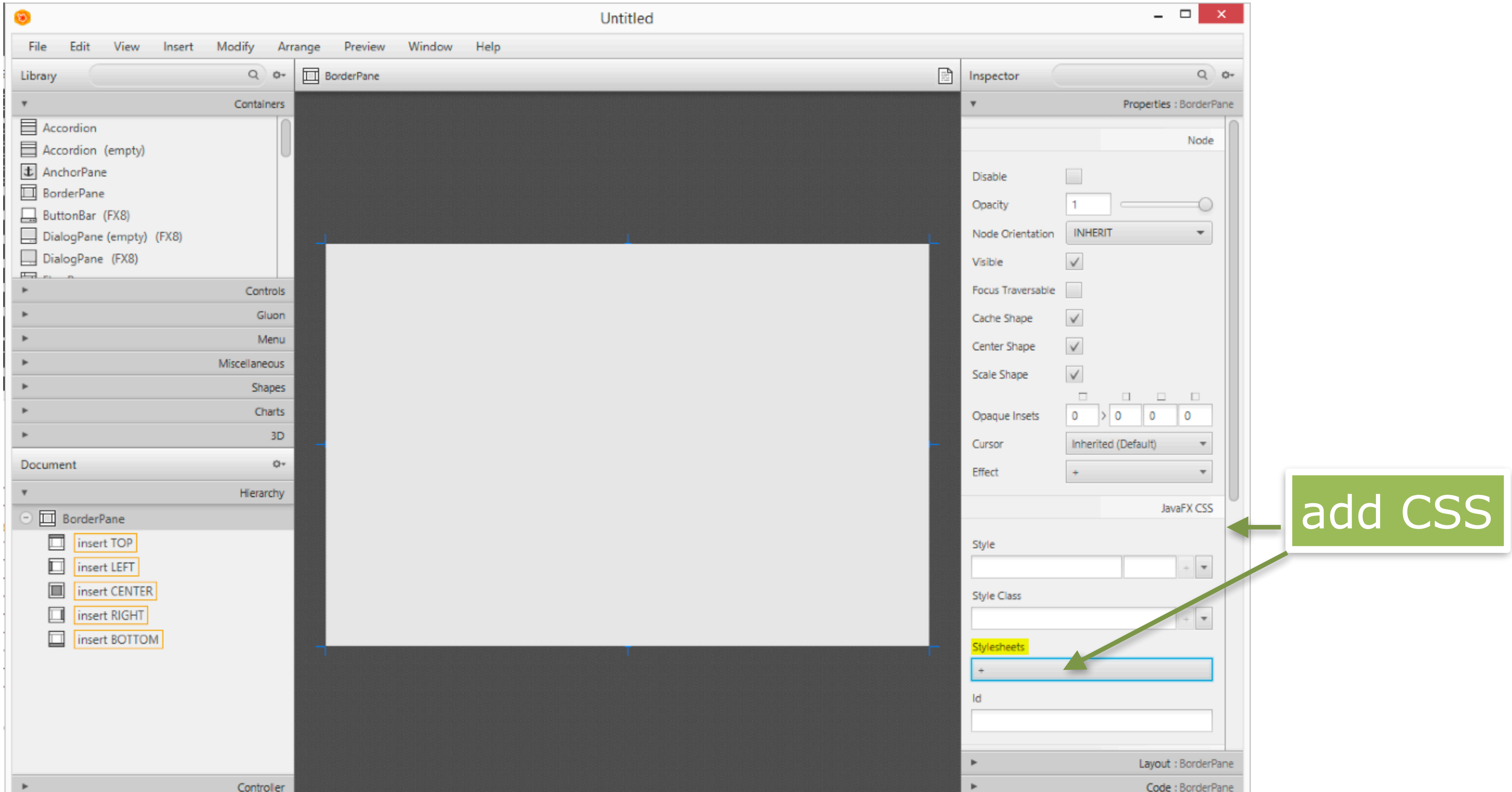




https://docs.oracle.com/javafx/2/get_started/css.htm

CSS and SceneBuilder

You can add CSS in SceneBuilder too on the Top hierarchy Node (here a BorderPane). But adding CSS in your Scene is more flexible (applicable across everything in the Scene).



Disable ☐
Opacity 1
Node Orientation INHERIT
Visible ☑
Focus Traversable ☐
Cache Shape ☑
Center Shape ☑
Scale Shape ☑
Opaque Insets 0 0 0 0
Cursor Inherited (Default)
Effect +

add CSS

Resources

https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm

https://docs.oracle.com/javase/8/javafx/layout-tutorial/size_align.htm#JFXLY133

https://o7planning.org/en/11009/javafx (lots on layouts)

https://docs.oracle.com/javafx/2/get_started/css.htm