Advanced Programming Methods

Lecture 10 - JavaFX

Our slides use examples from the followings:

JavaFX tutorials

- https://docs.oracle.com/javase/8/javase-clienttechnologies.htm
- https://wiki.eclipse.org/Efxclipse/Tutorials
- http://o7planning.org/en/11009/javafx
- http://code.makery.ch/library/javafx-8-tutorial/

JavaFX API

- https://docs.oracle.com/javase/8/javafx/api/toc.htm

Content

- What is JavaFX
- JavaFx Architecture
- Steps to install and set JavaFx (on Eclipse)
- Scene graph
- Simple JavaFx Application
- Layout management

What is JavaFX ?

- Classes and interfaces that provide support for creating Java applications that can be designed, implemented, tested on different platforms.
- Provides support for the use of Web components such as HTML5 code or JavaScript scripts
- Contains graphic UI components for creating graphical interfaces and manage their appearance through CSS files
- Provides support for interactive 3D graphics
- Provides support for handling multimedia content
- Supports RIAs (Rich Internet Application)
- Portability: desktop, browser, mobile, TV, game consoles, Blu-ray, etc.
- Ensures interoperability to Swing

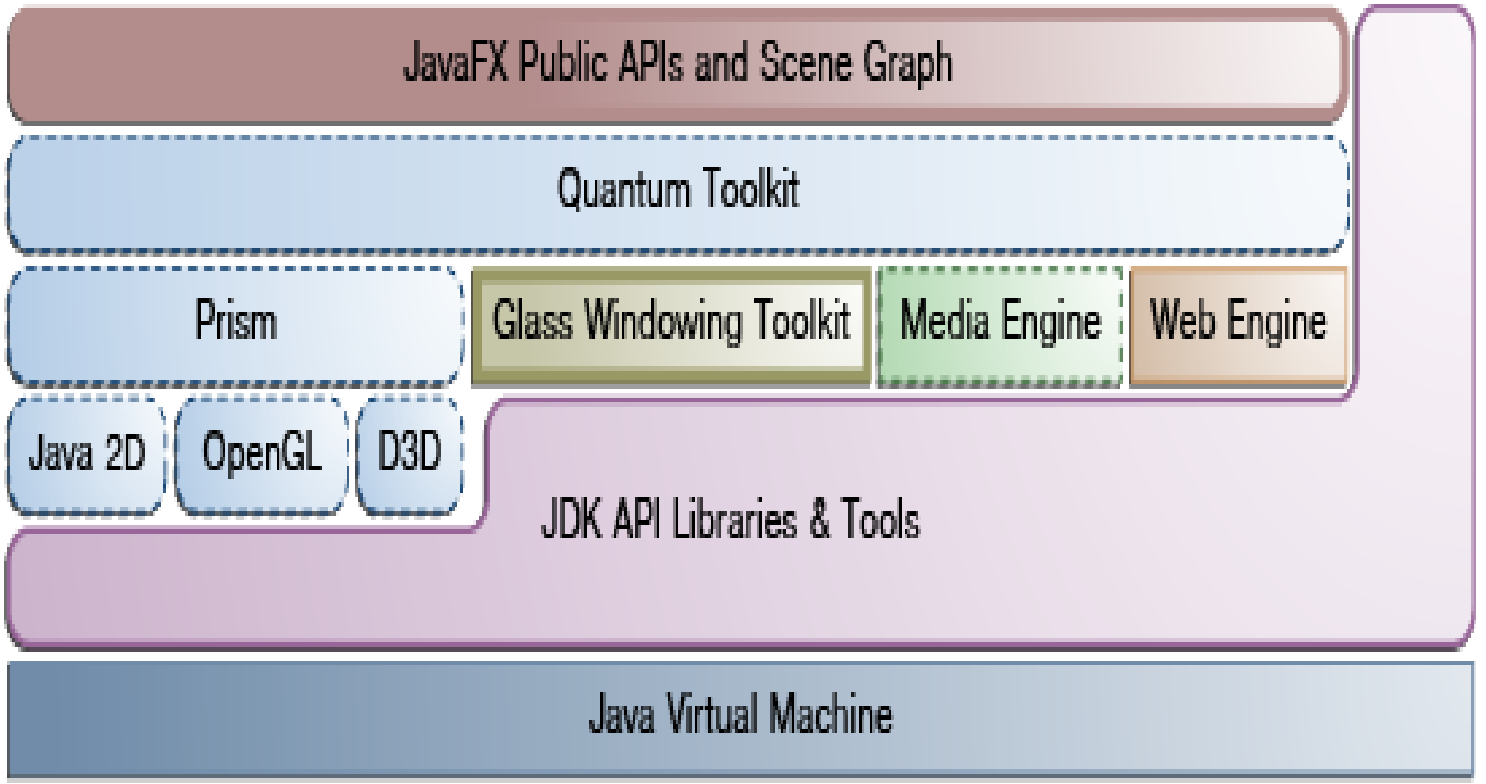
What is JavaFX ?

- Java 8 JavaFX is bundled with the Java platform, so JavaFX is available everywhere Java is
- From Java 8 you can also create standalone install packages for Windows, Mac and Linux with Java, which includes the JRE needed to run them.
  - This means that you can distribute JavaFX applications to these platforms even if the platform does not have Java installed already

JavaFX vs. Swing

- JavaFX is intended to replace Swing as the default GUI toolkit in Java.
- JavaFX is more consistent in its design than Swing, and has more features:
  - It is more modern too, enabling you to design GUI using layout files (XML) and style them with CSS, just like we are used to with web applications.
  - JavaFX also integrates 2D + 3D graphics, charts, audio, video, and embedded web applications into one coherent GUI toolkit.

JavaFX Architecture



JavaFX Architecture

- Scene Graph:
  - is the starting point for constructing a JavaFX application.
  - is a hierarchical tree of nodes that represents all of the visual elements of the application's user interface.
  - can handle input and can be rendered.

- Java Public API:
  - provides a complete set of Java public APIs that support rich client application development.

JavaFX Architecture

- Graphics System:
  - **Prism:**
    - processes render jobs.
    - can run on both hardware and software renderers, including 3-D.
    - is responsible for rasterization and rendering of JavaFX scenes.
  - **Quantum Toolkit**
    - ties Prism and Glass Windowing Toolkit together and makes them available to the JavaFX layer above them in the stack.
    - also manages the threading rules related to rendering versus events handling.

JavaFX Architecture

- Glass Windowing Toolkit
  - provides native operating services, such as managing the windows, timers, and surfaces.
  - serves as the platform-dependent layer that connects the JavaFX platform to the native operating system.
  - is also responsible for managing the event queue.
  - runs on the same thread as the JavaFX application

JavaFX Architecture

- Running Threads
  - **JavaFX application thread:**
    - primary thread used by JavaFX application developers
    - Any ”live” scene, which is a scene that is part of a window, must be accessed from this thread
    - A scene graph can be created and manipulated in a background thread, but when its root node is attached to any live object in the scene, that scene graph must be accessed from the JavaFX application thread
  - **Prism render thread:**
    - handles the rendering separately from the event dispatcher.
  - **Media thread:**
    - runs in the background and synchronizes the latest frames through the scene graph by using the JavaFX application thread.

JavaFX Architecture

- Pulse:
  - is an event that indicates to the JavaFX scene graph that it is time to synchronize the state of the elements on the scene graph with Prism.
  - The Glass Windowing Toolkit is responsible for executing the pulse events
- Media and Images:
  - JavaFX supports both visual and audio media
- Web Component:
  - is a JavaFX UI control, based on Webkit, that provides a Web viewer and full browsing functionality through its API.

JavaFX Architecture

- JavaFX Cascading Style Sheets (CSS)
  - provides the ability to apply customized styling to the user interface of a JavaFX application without changing any of that application's source code.
  - can be applied to any node in the JavaFX scene graph and are applied to the nodes asynchronously.
  - can also be easily assigned to the scene at runtime, allowing an application's appearance to dynamically change.

JavaFX Architecture

- JavaFX UI Controls
  - available through the JavaFX API
  - are built by using nodes in the scene graph.
  - are portable across different platforms

JavaFX Architecture

- Layout:
  - Layout containers (panes) can be used to allow for flexible and dynamic arrangements of the UI controls within a scene graph

- Transformations 2-D and 3-D
  - Each node in the JavaFX scene graph can be transformed in the x-y coordinate

- Visual Effects:
  - to enhance the look of JavaFX applications in real time.
  - are primarily image pixel-based

Steps to install and set JavaFx (on Eclipse)

- **Step 1:** install e(fx)clipse into Eclipse (JavaFx tooling)
  - You can follow the steps from one of the followings:
    - http://o7planning.org/en/10619/install-efxclipse-into-eclipse
    - https://wiki.eclipse.org/Efxclipse/Tutorials/AddingE(fx)clipse_to_eclipse

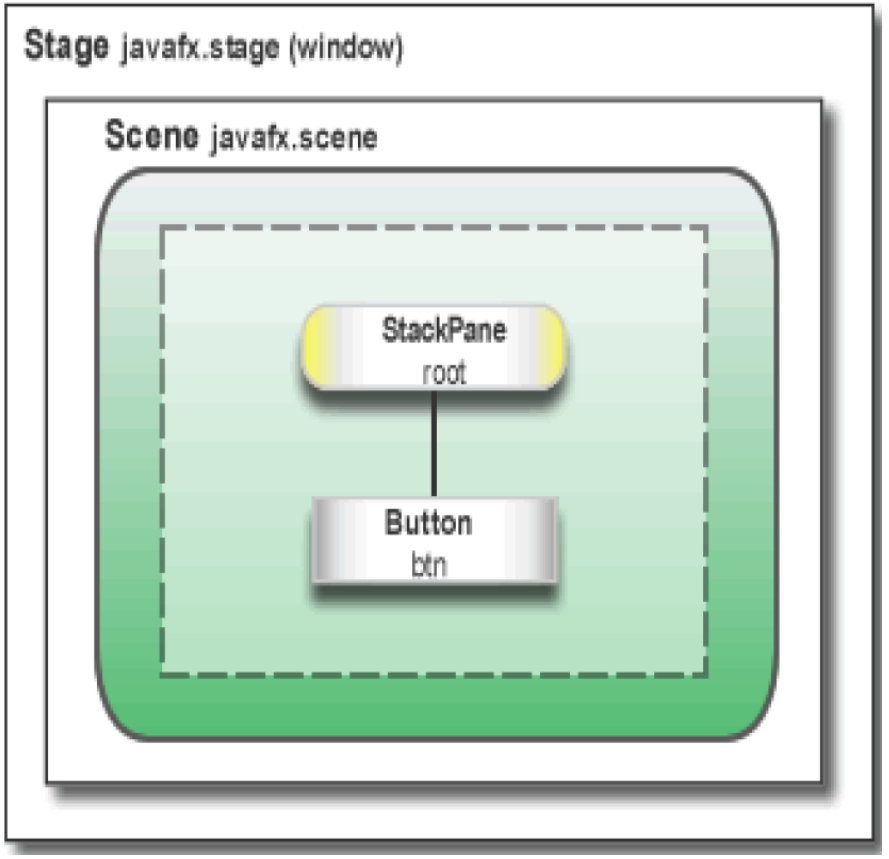
- **Step 2:** download and set JavaFx SceneBuilder
  - You can follow the steps from http://o7planning.org/en/10621/install-javafx-scene-builder-into-eclipse
  - You can download the SceneBuilder
    - Either from Oracle
      http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-1x-archive-2199384.html
    - Or from http://gluonhq.com/products/scene-builder/

Scene Graph

scene-graph-based programming model

A JavaFX application contains:

- An object **Stage (window)**
- One or more objects **Scene**



Scene Graph

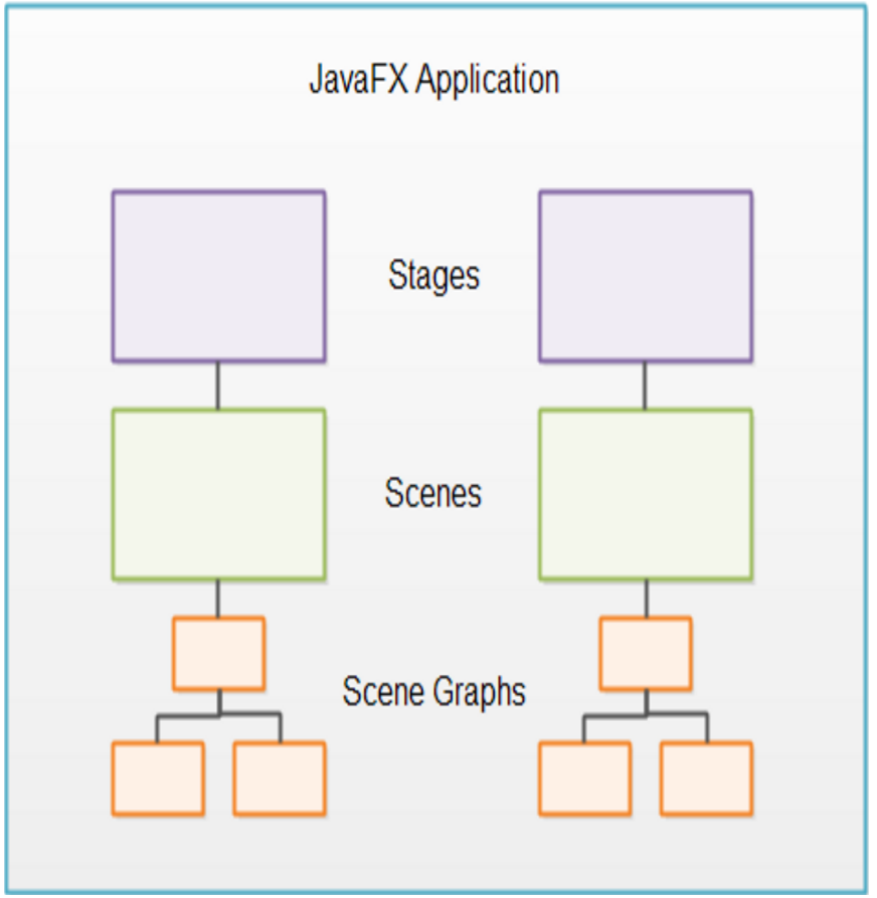
- A Scene Graph is a tree of graphical user interface components.

- A scene graph element is a node.
- Each node has an ID, a class style, a bounding volume, etc.

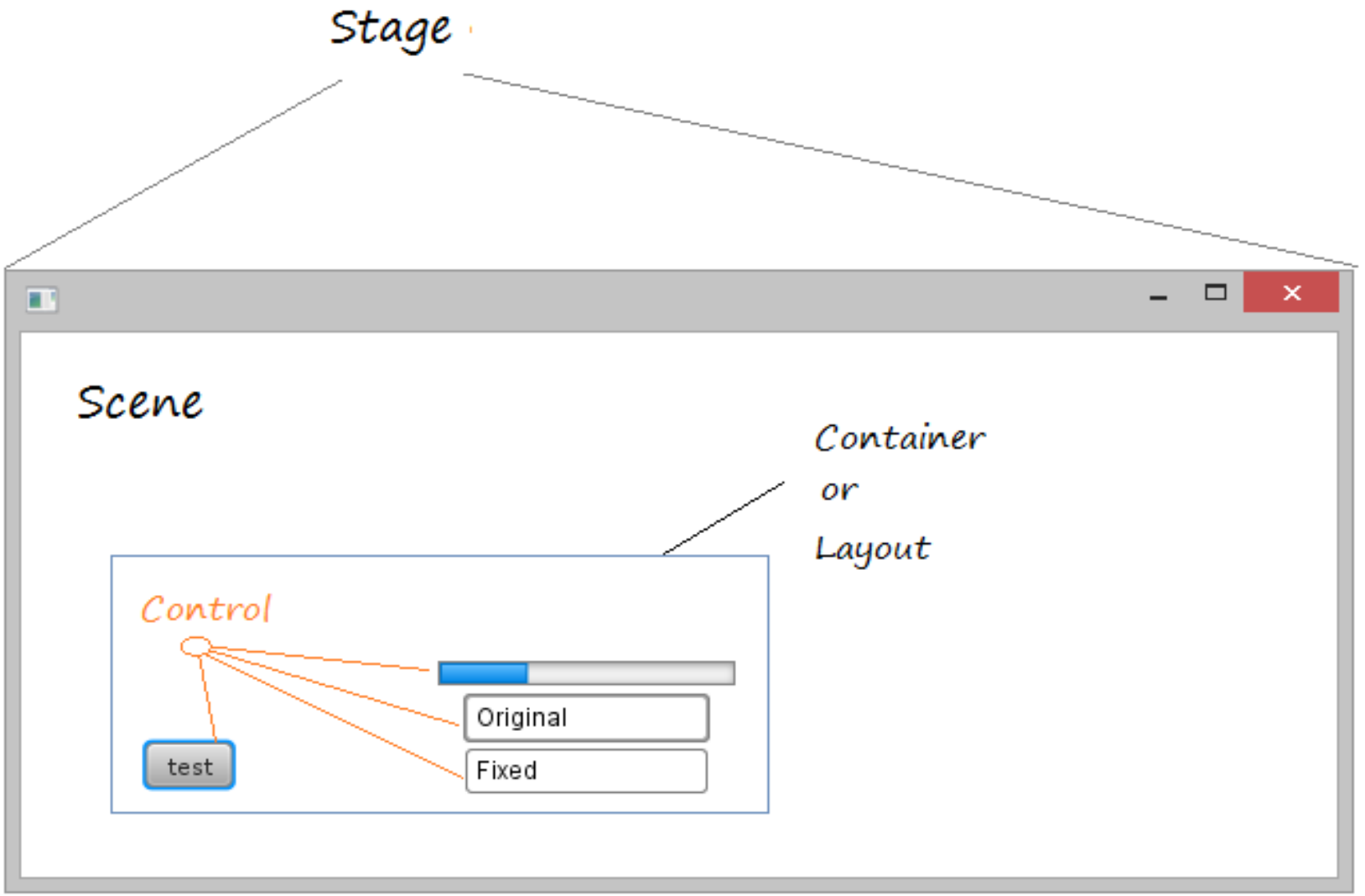
- Except the root node, each node has a single parent and 0 or more children.
- Nodes can be internal (Parent) or leaf

- A node can be associated with various properties (effects (blur, shadow), opacity, transformations) and events (event handlers (Mouse, Keyboard))

Scene Graph



Scene Graph


Stage
Scene
Container
or
Layout
Control
Original
Fixed
test

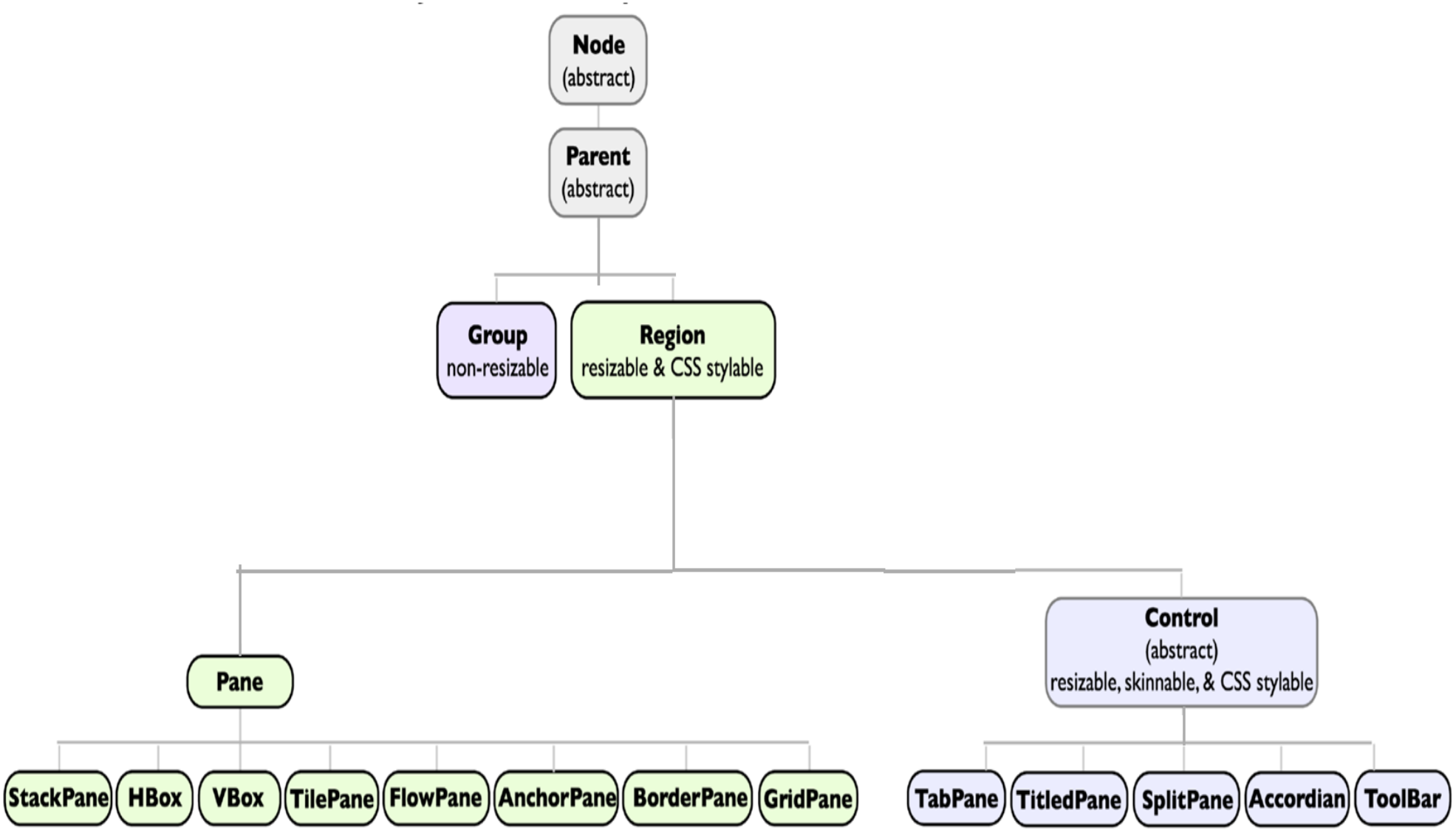
Scene Graph

- Node: The abstract base class for all scene graph nodes.

- Parent: The abstract base class for all branch nodes. (This class directly extends Node).

- Scene: The base container class for all content in the scene graph.

Scene Graph


Node
(abstract)
Parent
(abstract)
Group
non-resizable
Region
resizable & CSS stylable
Pane
Control
(abstract)
resizable, skinnable, & CSS stylable
StackPane
HBox
VBox
TilePane
FlowPane
AnchorPane
BorderPane
GridPane
TabPane
TitledPane
SplitPane
Accordian
ToolBar

JavaFX Application

- A JavaFX application is an instance of class **Application**

```
public abstract class Application extends Object;
```

- Creation of a new object of class Application is done by executing the static method ***launch()*** from the class ***Application***:

```
public static void launch(String... args);
```

  - `args` **aplication parameters**(parameters of the method *main*).

- JavaFX runtime executes the following steps:

  1. Create a new object Application
  2. Call the method init of the new object Application
  3. Call the method start of the new object Application
  4. Wait for the application to terminate

Application parameters can be obtained by calling the method *getParameters*()

JavaFX Application

```
public class Main extends Application {

    @Override
    public void start(Stage stage) {
        Group root = new Group();
        Scene scene = new Scene(root, 500, 500, Color.PINK);
        stage.setTitle("Welcome to JavaFX!");

        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args); //an object Application is created
    }
}
```

Adding nodes

```
// A node of type Group is created
Group group = new Group();


// A node of type Rectangle is created
Rectangle r = new Rectangle(25,25,50,50);
r.setFill(Color.BLUE);
group.getChildren().add(r);

// A node of type Circle is created
Circle c = new Circle(200,200,50, Color.web("blue", 0.5f));
group.getChildren().add(c);
```

JavaFX Application

DEMO

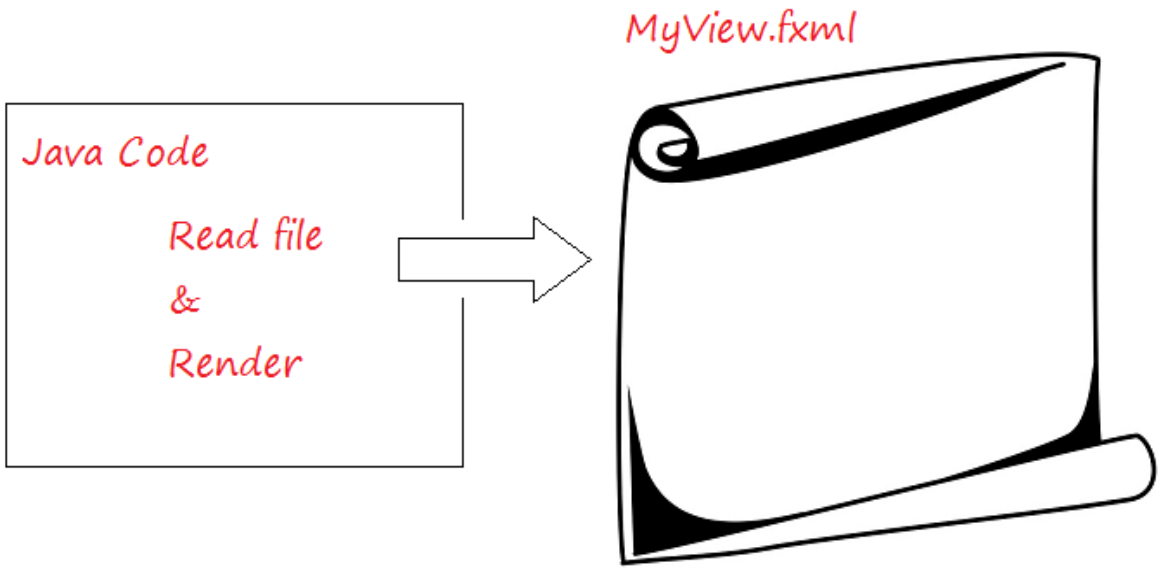
JavaFx Scene Builder

- to create a JavaFX application interface, you can write code in Java entirely.
  - it takes so much time to do this,

- JavaFX Scene Builder is a visual tool allowing you to design the interface of Scene.
  - The code which is generated, is XML code saved on *.fxml file.

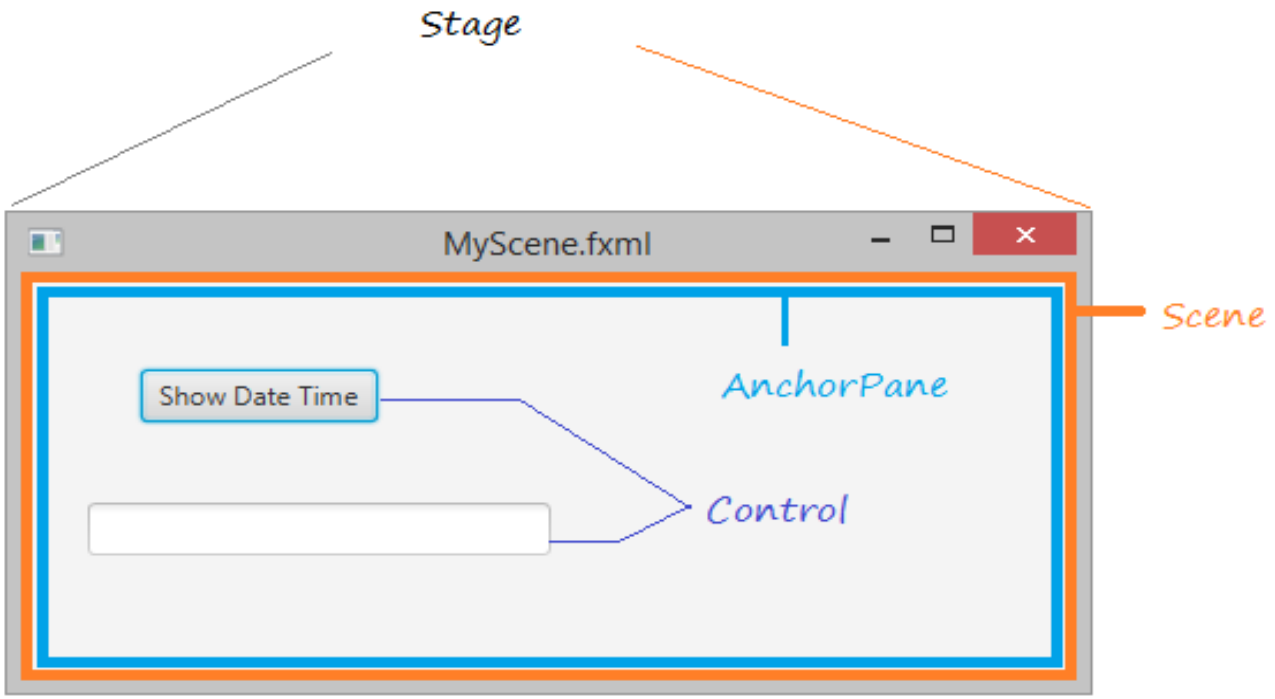
JavaFx Scene Builder

- to create a JavaFX application interface, you can write code in Java entirely.
  - it takes so much time to do this,
- JavaFX Scene Builder is a visual tool allowing you to design the interface of Scene.
  - The code which is generated, is XML code saved on *.fxml file.



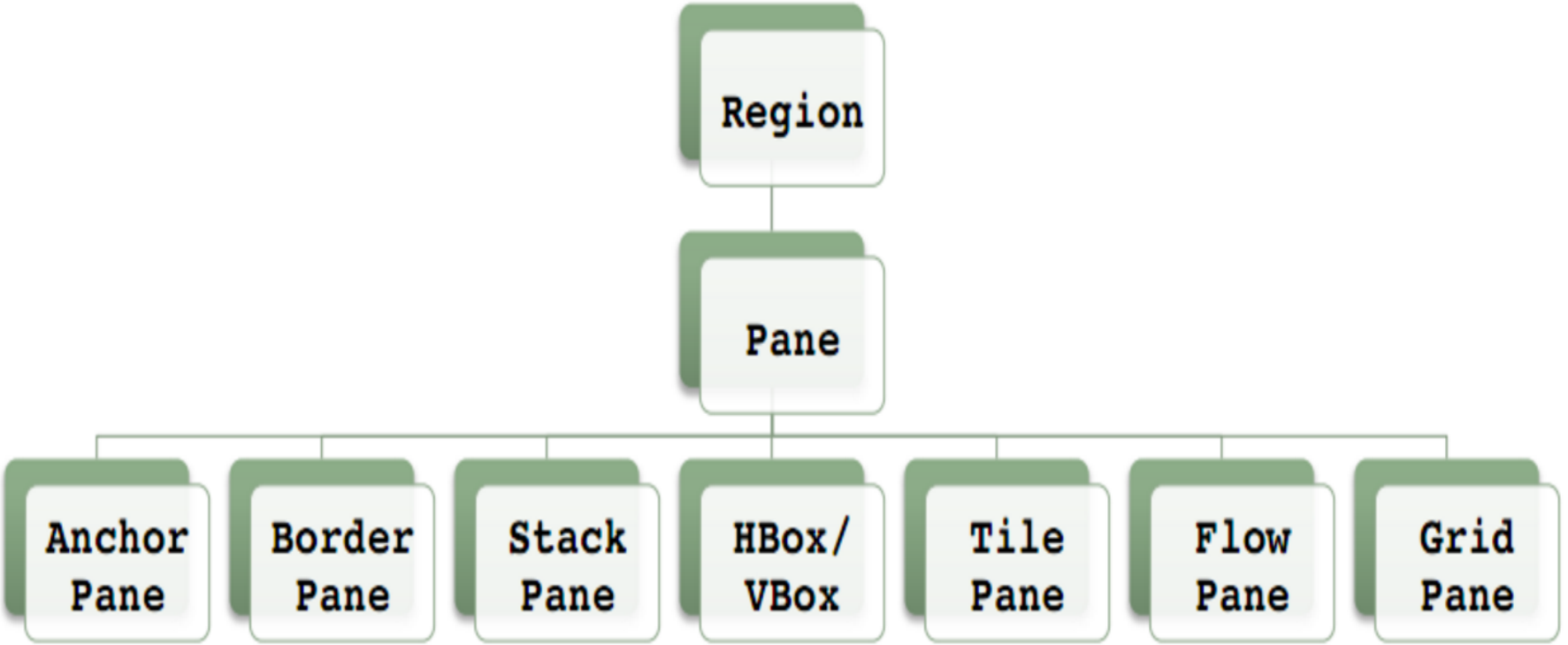
JavaFx Scene Builder

- DEMO

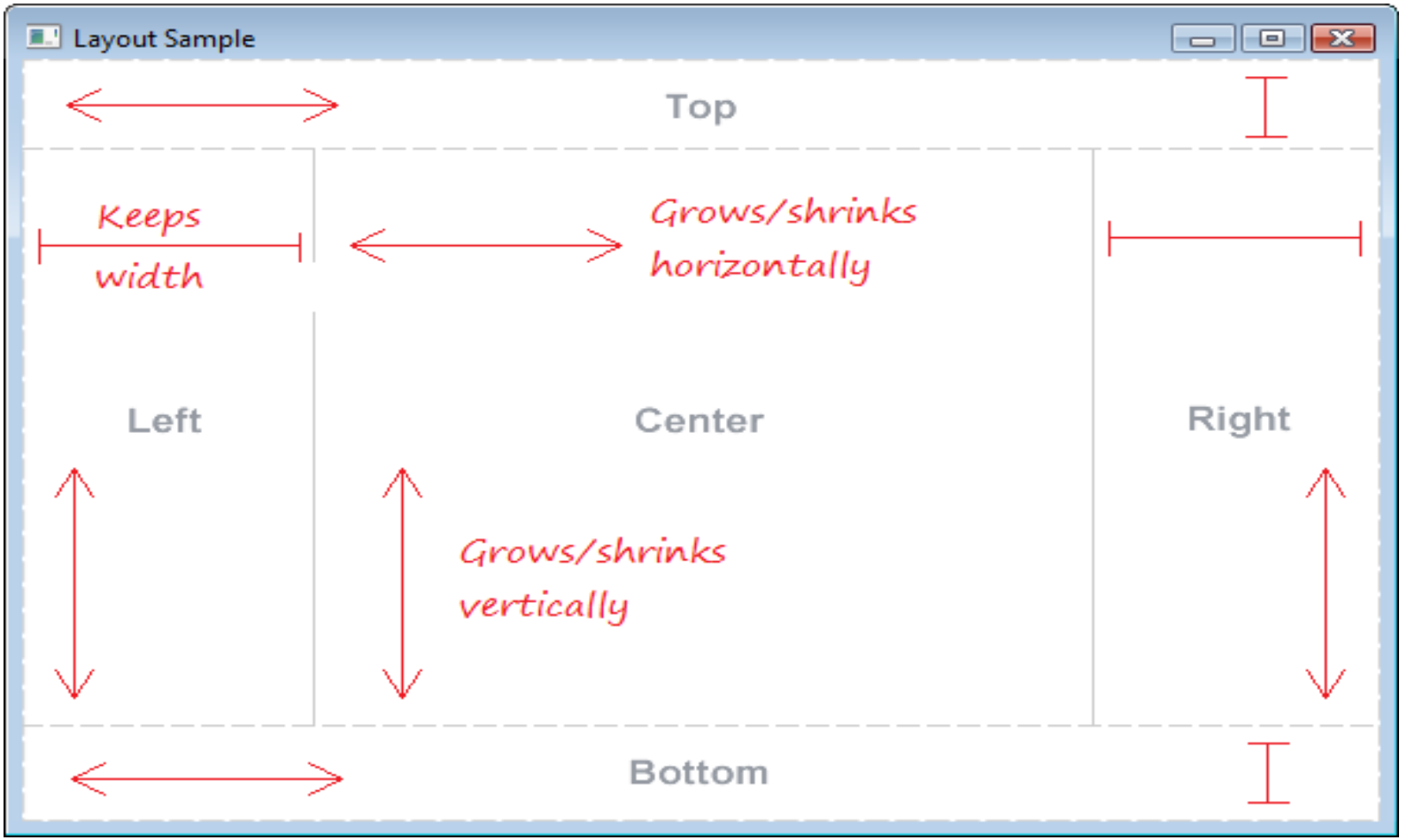


Layout management

- layouts are components which contains other components inside them and manage the nested components

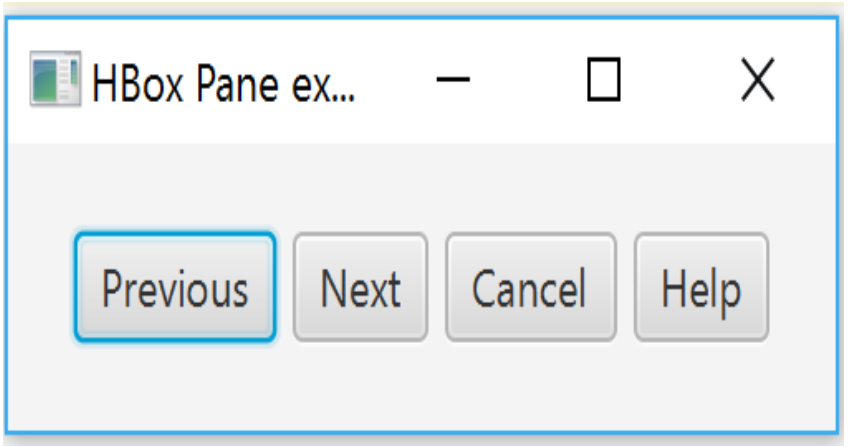


Layout Management - BorderPane



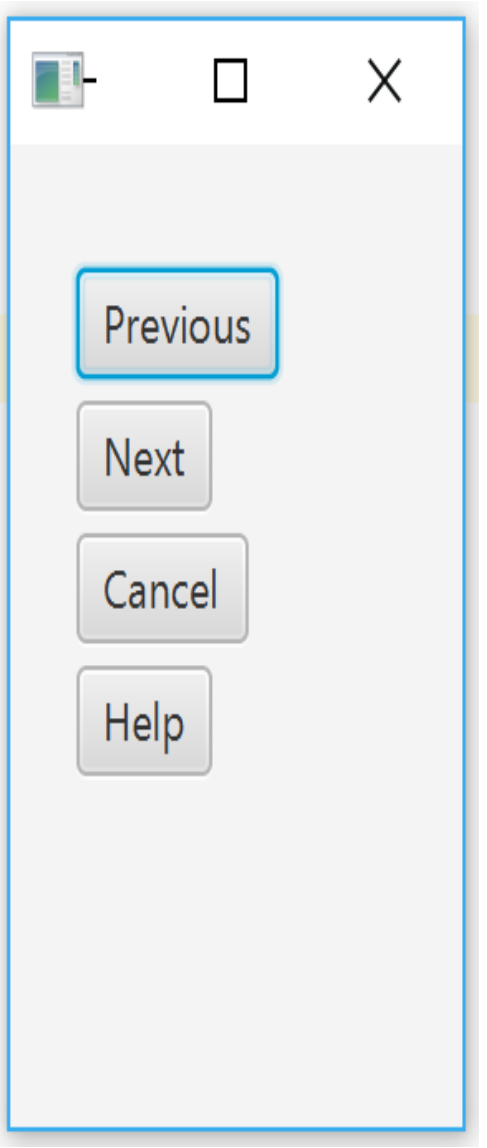
Layout Management - HBOX

```
HBox root = new HBox(5);
root.setPadding(new Insets(100));
root.setAlignment(Pos.BASELINE_RIGHT);

Button prevBtn = new Button("Previous");
Button nextBtn = new Button("Next");
Button cancBtn = new Button("Cancel");
Button helpBtn = new Button("Help");

root.getChildren().addAll(prevBtn,
nextBtn, cancBtn, helpBtn);
```



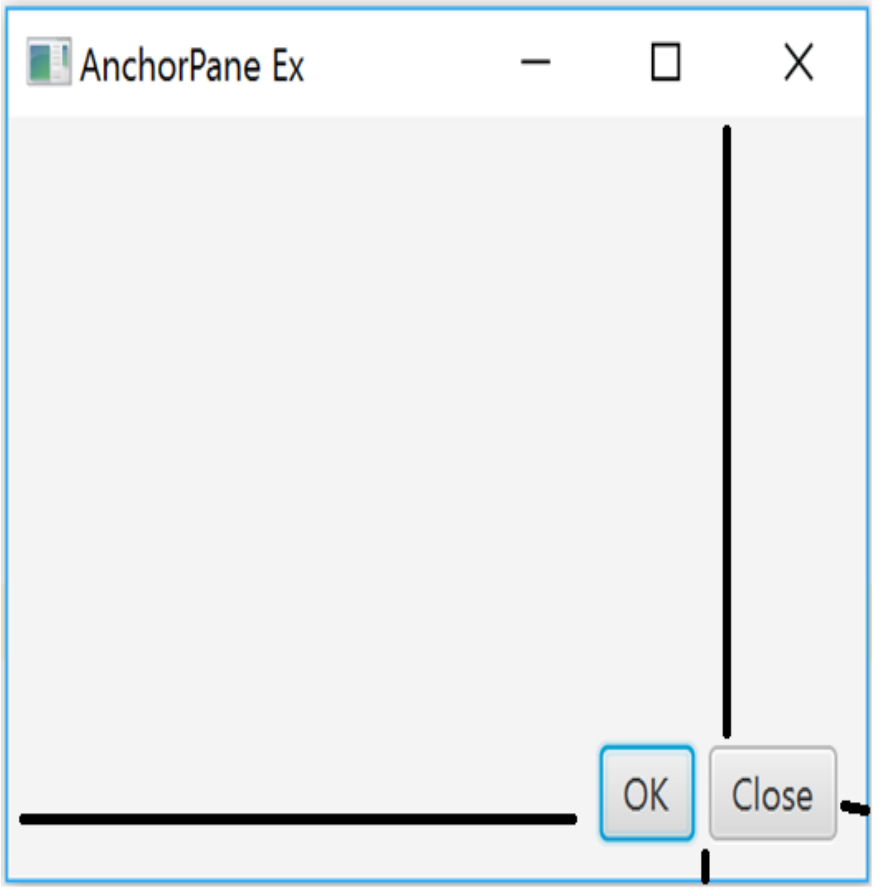
Layout Management - VBOX

```
VBox root = new VBox(5);
root.setPadding(new Insets(20));
root.setAlignment(Pos.BASELINE_LEFT);

Button prevBtn = new Button("Previous");
Button nextBtn = new Button("Next");
Button cancBtn = new Button("Cancel");
Button helpBtn = new Button("Help");

root.getChildren().addAll(prevBtn,
nextBtn, cancBtn, helpBtn);
Scene scene = new Scene(root, 150, 200);
```



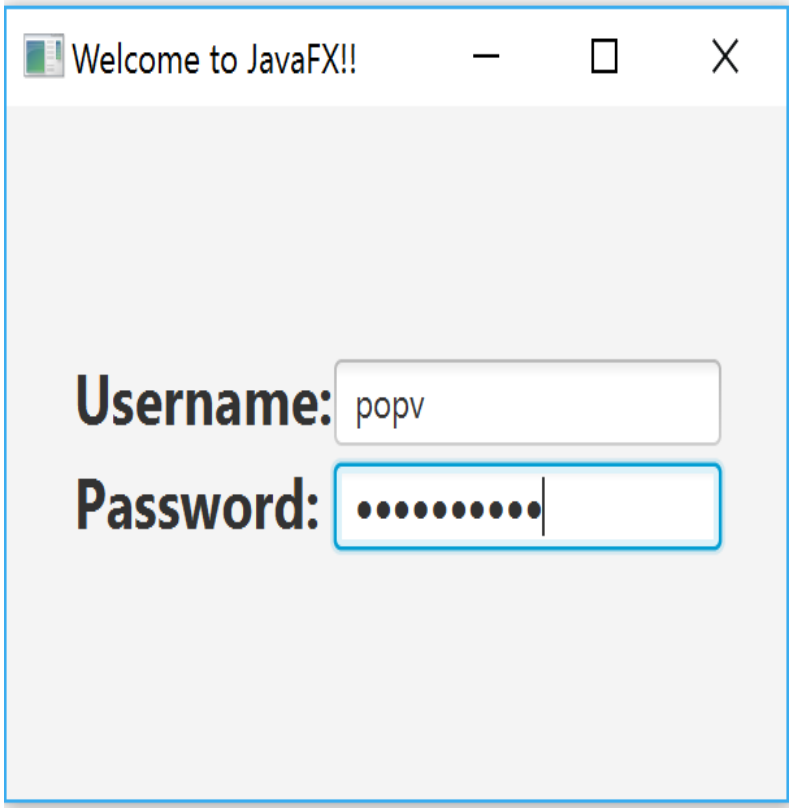
Layout Management - AnchorPane

```
AnchorPane root = new AnchorPane();

Button okBtn = new Button("OK");
Button closeBtn = new
Button("Close");
HBox hbox = new HBox(5, okBtn,
closeBtn);

root.getChildren().addAll(hbox);

AnchorPane.setRightAnchor(hbox, 10d);
AnchorPane.setBottomAnchor(hbox,
10d);

Scene scene = new Scene(root, 300,
200);

stage.setTitle("AnchorPane Ex");
stage.setScene(scene);
stage.show();
```



Layout Management - GridPane

```
GridPane gr=new GridPane();
gr.setPadding(new Insets(20));
gr.setAlignment(Pos.CENTER);

gr.add(createLabel("Username:"),0
,0);
gr.add(createLabel("Password:"),0
,1);

gr.add(new TextField(),1,0);
gr.add(new PasswordField(),1,1);

Scene scene = new Scene(gr, 300,
200);
stage.setTitle("Welcome to
JavaFX!!");
stage.setScene(scene);
stage.show();
```



JavaFx Controls

- Are the main components of the GUI
- A control is a node in the scene graph
- Can be manipulated by the user
- Java FX Controls reference:

https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/ui_controls.htm#JFXUI336

Button
Radio Button
Toggle Button
Checkbox
Choice Box
Text FieldLabel
Password Field
Scroll Bar
Scroll Pane

List View
Table View
Tree View
Combo Box
Separator
Slider
Progress Bar and Progress I
Hyperlink

Tooltip
HTML Editor
Titled Pane and Accordion
Menu
Color Picker
Pagination Control
File Chooser
Customization of UI Controls