

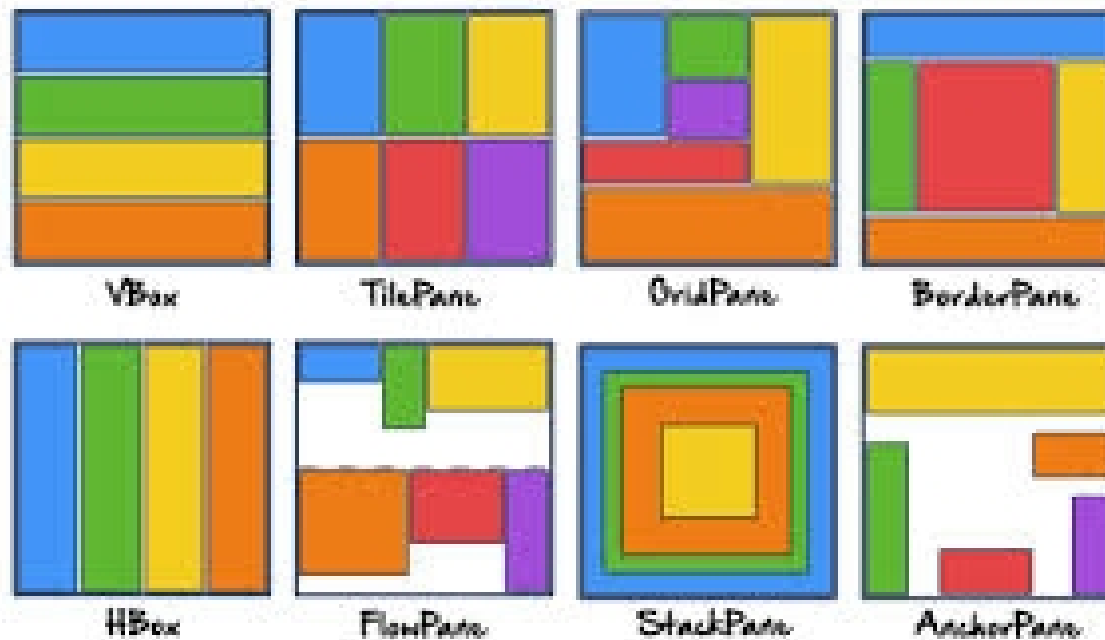
# Layout a User Interface

How would you create this user interface?



# Containers, Layouts, & Controls

Graphics frameworks use **containers** to divide the U.I. into **regions**, and to layout **components** in each region. In JavaFX, a **Pane** is a **container** with built-in **layout** :



*The color blocks show how components are layed out inside of different Panes (containers).*

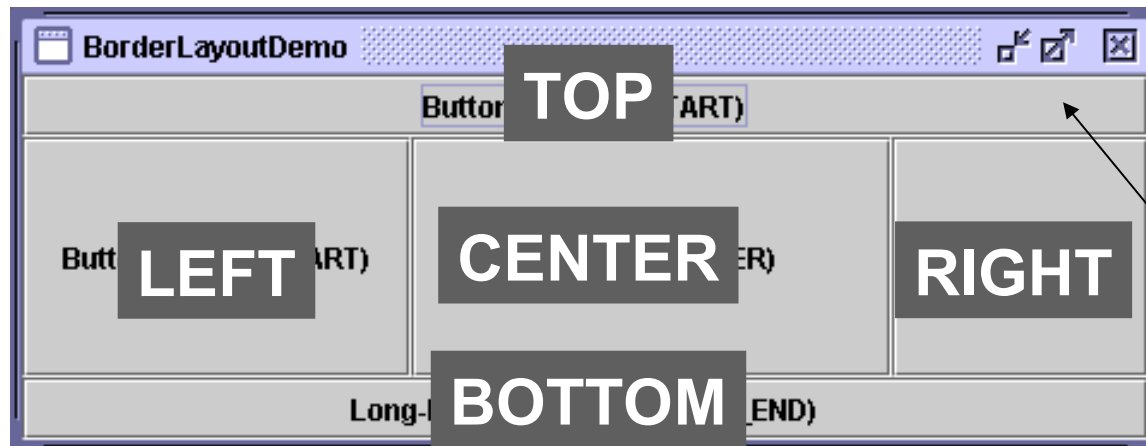
# Define Regions & Choose a Layout

Divide the UI into **Regions** using a container.



# Choose a Layout

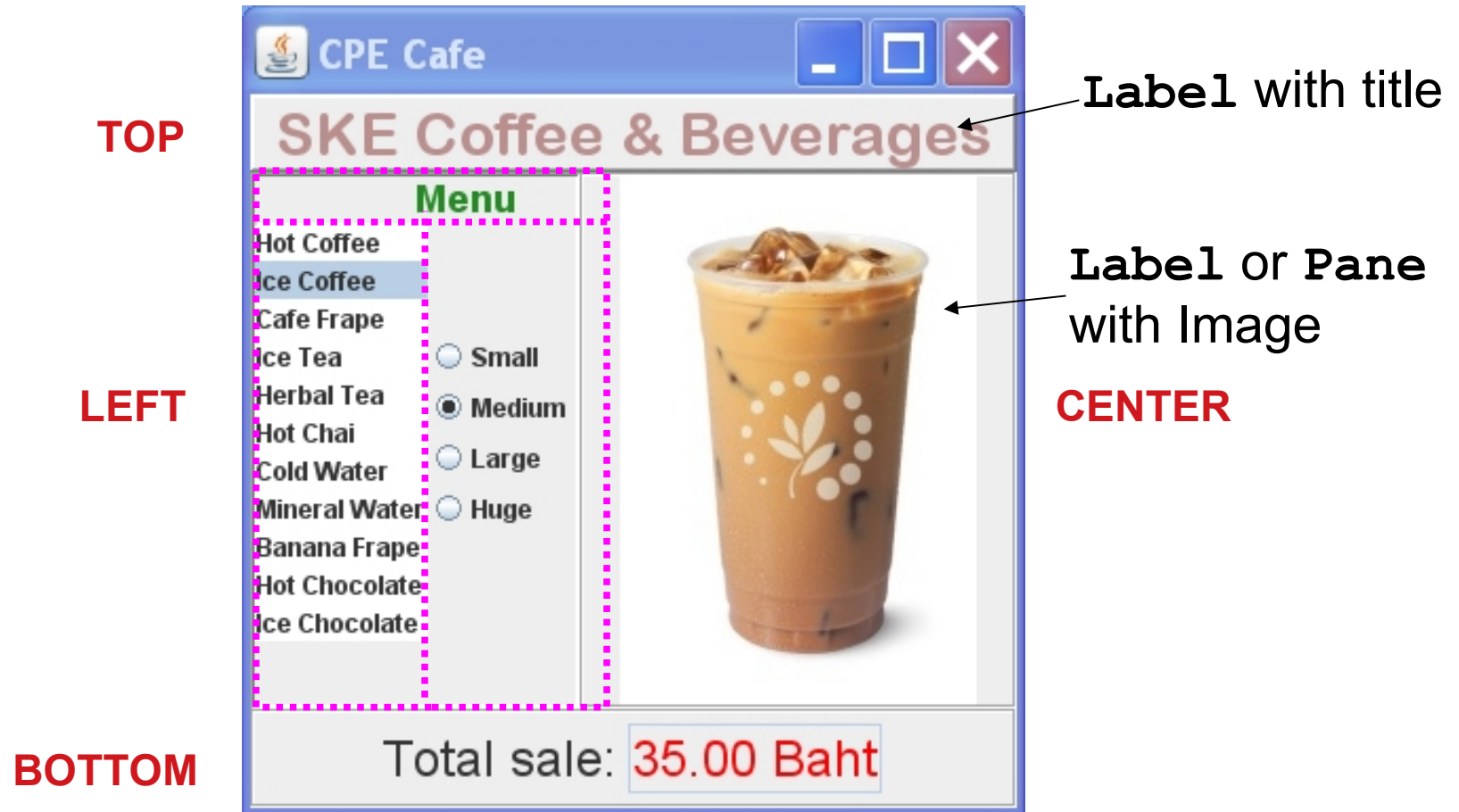
- **BorderPane** divides a region into 5 sub-regions.
- If a sub-region is empty, it is not shown.
- Each sub-region grows to fit its contents.
- Center gets preference for space.
- use: `borderpane.setTop(node)` or `.setCenter(node)` ...



```
Label title = new Label("SKE Coffee & ..");  
borderpane.setTop(title);
```

# Layout the Left Region (Menu)

We need a separate container to layout the left side.



# Layout & Controls for Left Region

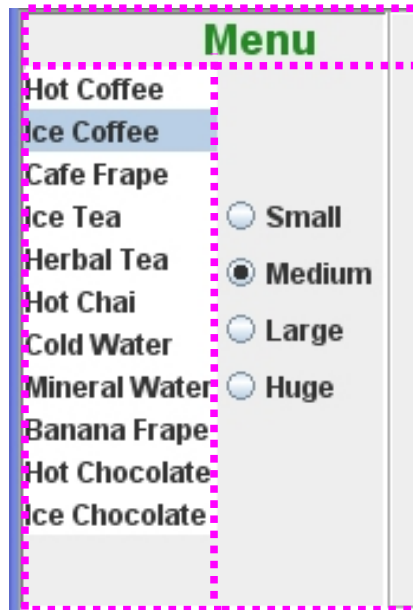
BorderPane or GridPane will work.

Label with title

TOP

LEFT

**ListView**  
for menu  
items



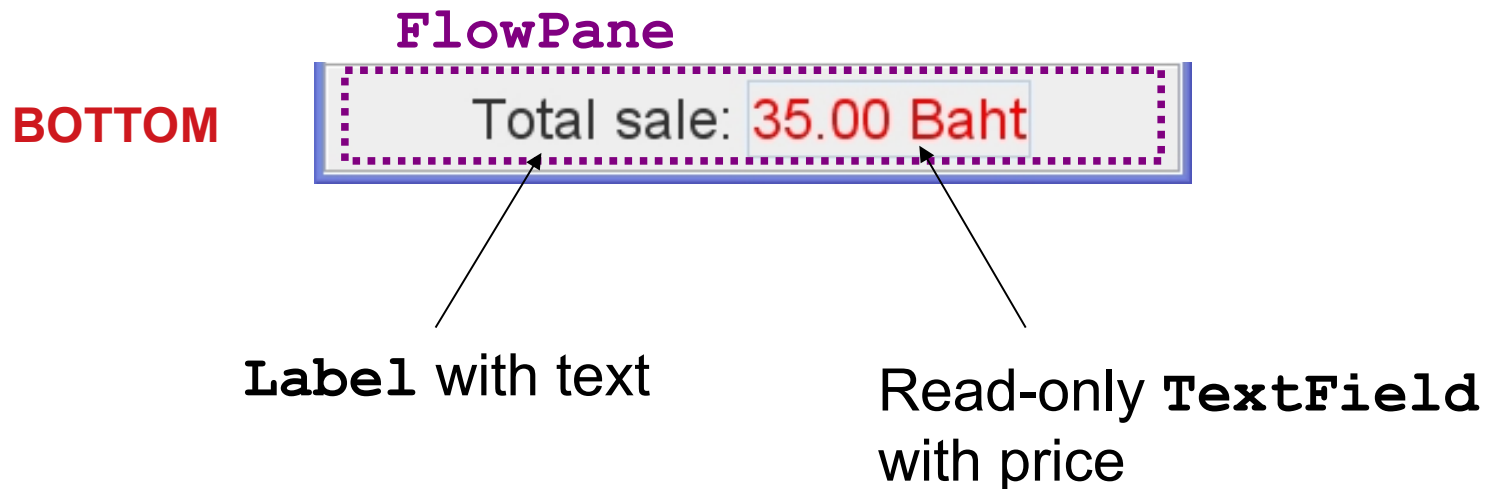
RIGHT

**VBox**  
containing  
**RadioButtons**

# Refine the Bottom Region

We can use a **FlowPane** for the bottom region.

Use `pane.setAlignment()` so the contents are centered.



# Put Container inside Container

Build the overall GUI from the parts

**TOP**


**LEFT**

**BorderPane**  
for menu

**BOTTOM**

**FlowPane**  
for total

**RIGHT**  
or  
**CENTER**

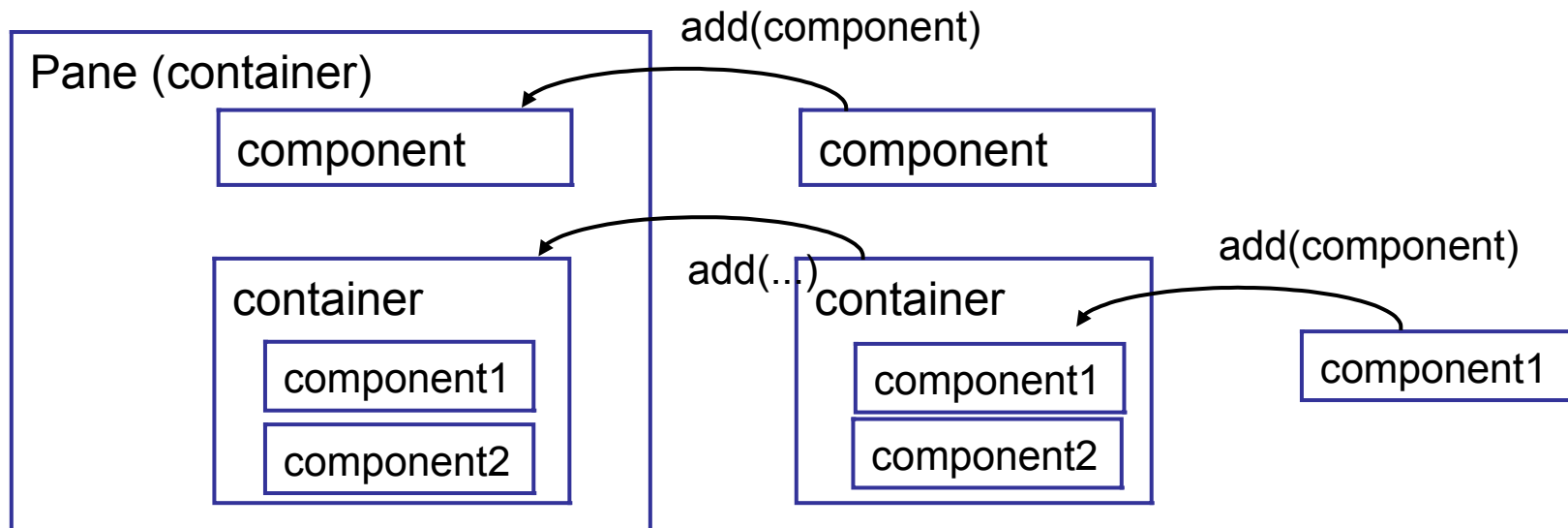


The screenshot shows a Java Swing window titled "CPE Cafe" with standard Windows-style window controls (minimize, maximize, close). The window contains a GUI for "SKE Coffee & Beverages". The GUI is organized into three main sections: a top title bar, a left menu area, and a right area for the coffee image. The menu area is a **BorderPane** containing a list of items: Hot Coffee, Ice Coffee (selected), Cafe Frappe, Ice Tea, Herbal Tea, Hot Chai, Cold Water, Mineral Water, Banana Frappe, Hot Chocolate, and Ice Chocolate. To the right of the menu are radio buttons for size: Small, Medium (selected), Large, and Huge. The right area displays a large image of an iced coffee drink. At the bottom, a **FlowPane** displays the text "Total sale: 35.00 Baht".



# Controls inside Container

- A GUI consists of **components** in **containers**.
- A **container** contains other components.
- JavaFX calls them **Nodes**, **Pane**, and **Group**

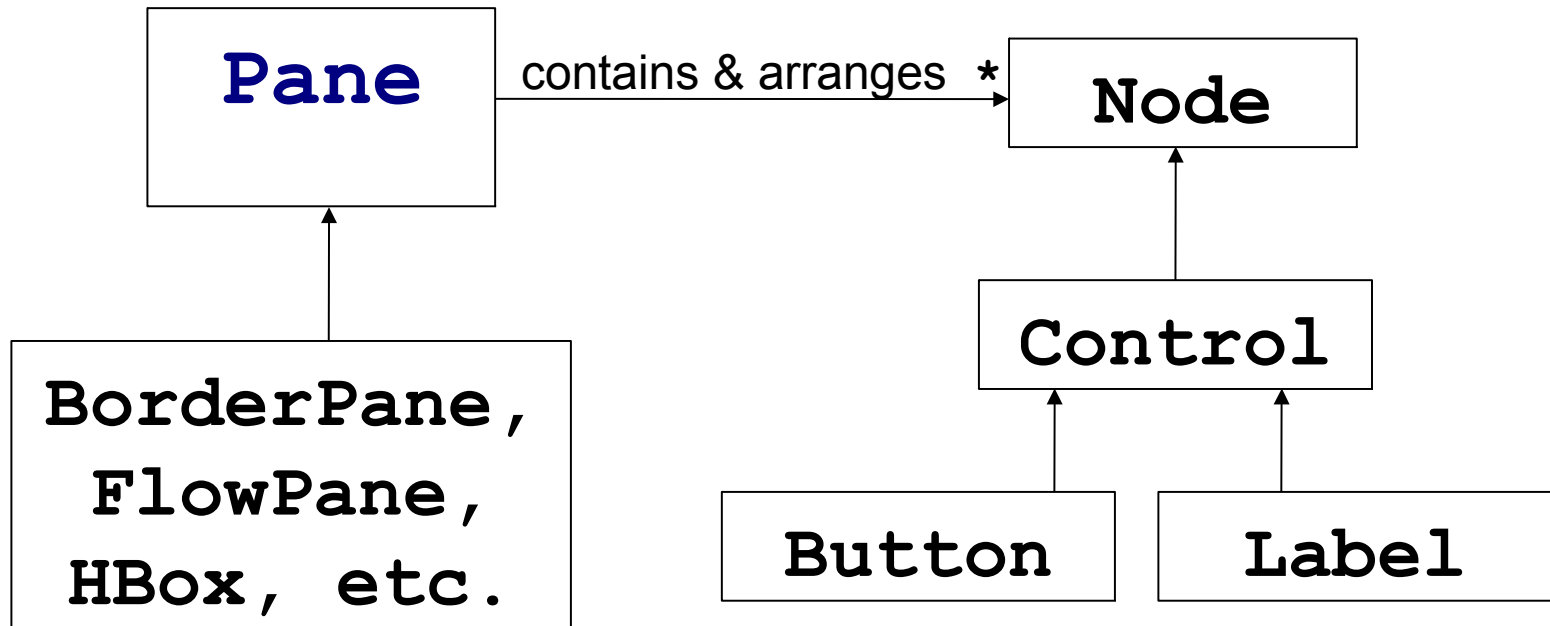


# How Does this Work?

A **Pane** or **Group** contains one or more **Nodes**.

Every control is a subclass of **Node**.

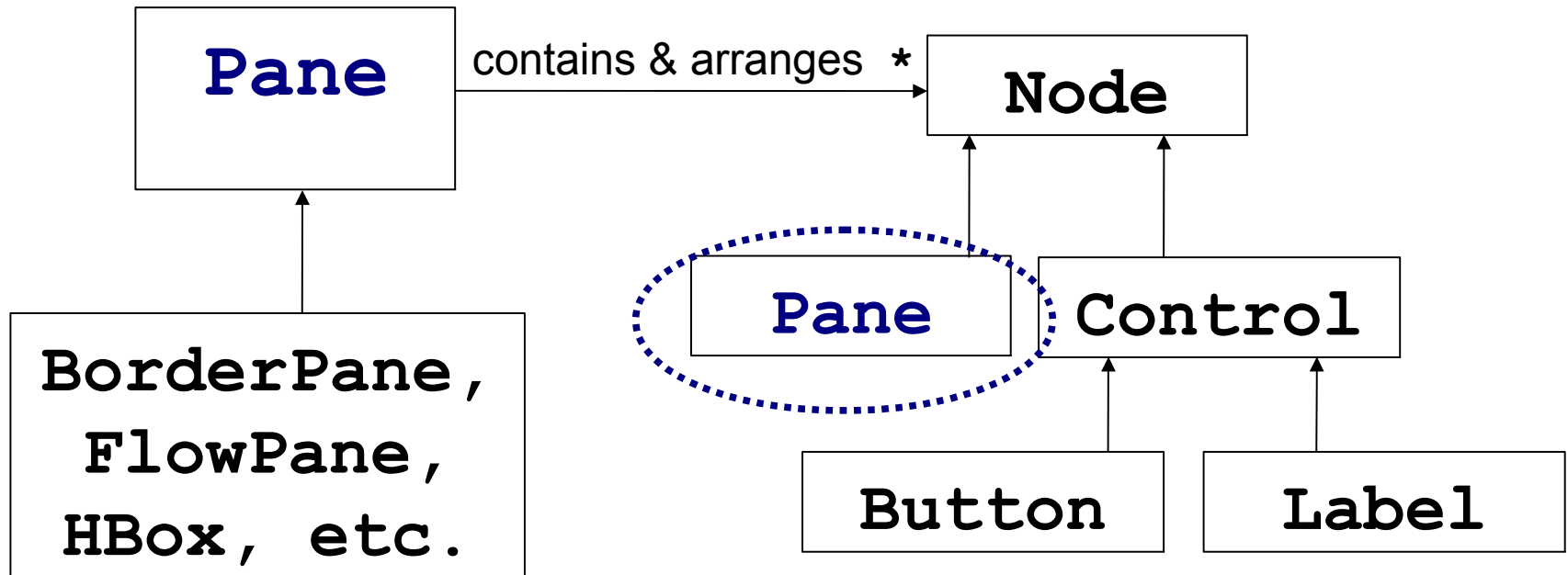
Subclasses of **Pane** provide special layouts.



# A Pane is also a Node!

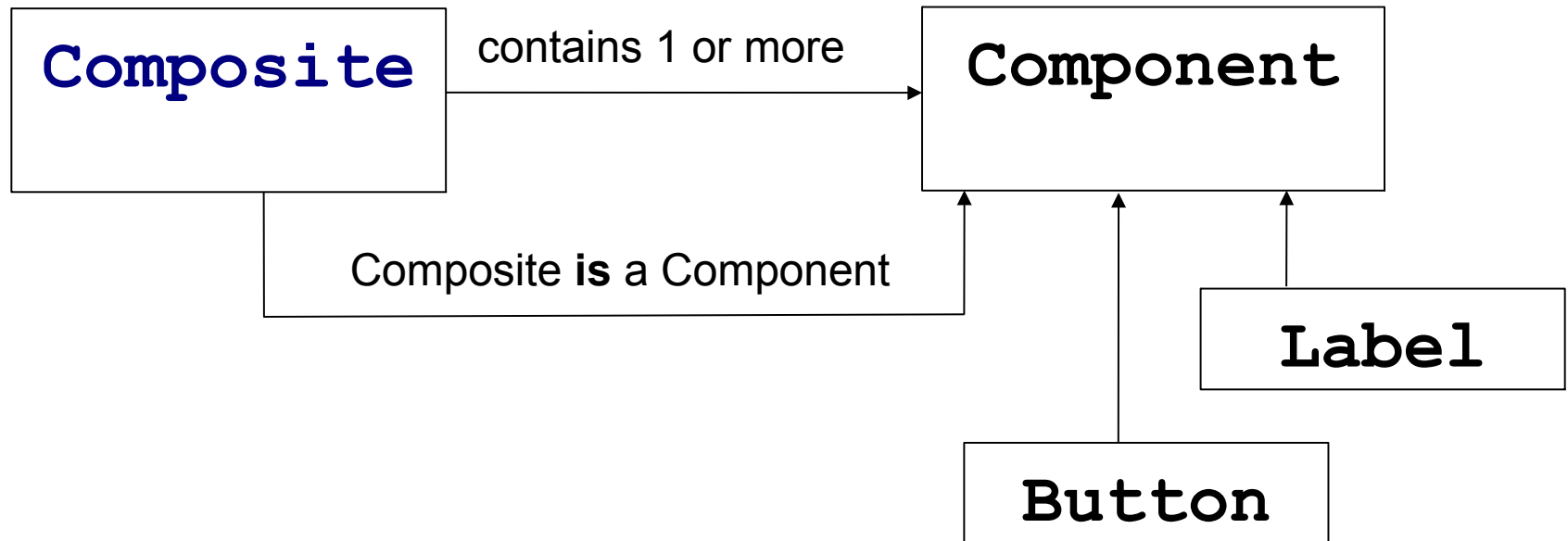
A **Pane** is also a subclass of **Node**.

So a **Pane** can contain other **Panes** (composition).



# Composite Design Pattern

A **Composite** contains components,  
and the **Composite** itself is also a kind of **Component**.

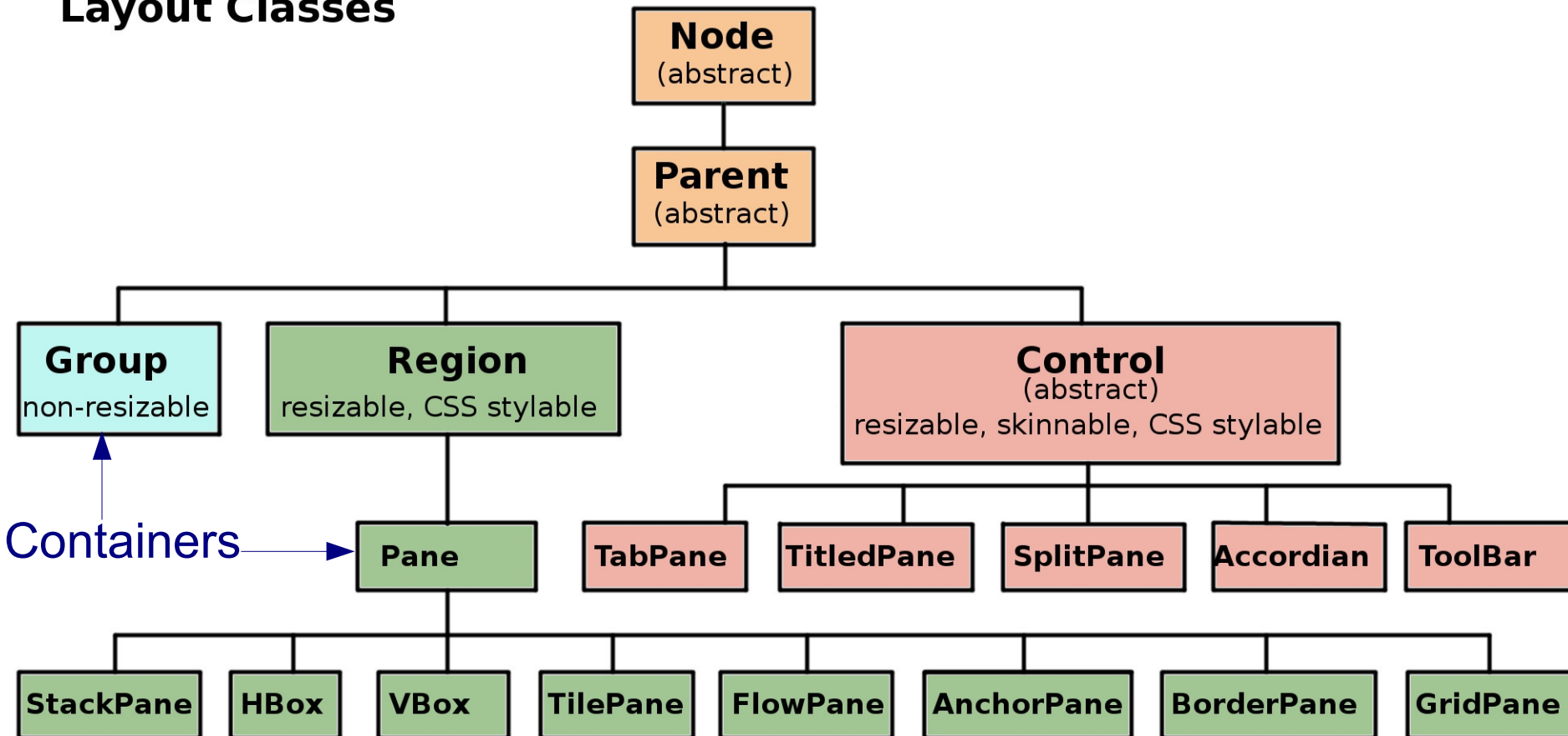


# JavaFX Classes class hierarchy

**Pane** and **Group** are containers for other Nodes.

Button, TextField, etc. are subclasses of Control.

## Layout Classes



# What You Need to Know

---

What are the Containers? How to they Look?

**FlowPane** - components "flow" to available space

**BorderPane** - 5 regions

**GridPane** - a flexible grid of components. Node can span multiple columns or rows.

**VBox** - vertical boxes of different sizes

# How To Customize the Layout?

You need to know the properties you can set.

This is easier using SceneBuilder

```
setAlignment( Pos.CENTER )
```

```
setVGap( 2.0 )           // space between components
```

```
setHGap( 5.0 )
```

```
setPadding( new Insets(10.0) ) // space around edges
```

```
setPrefWidth( 50.0 )      // try to avoid this
```

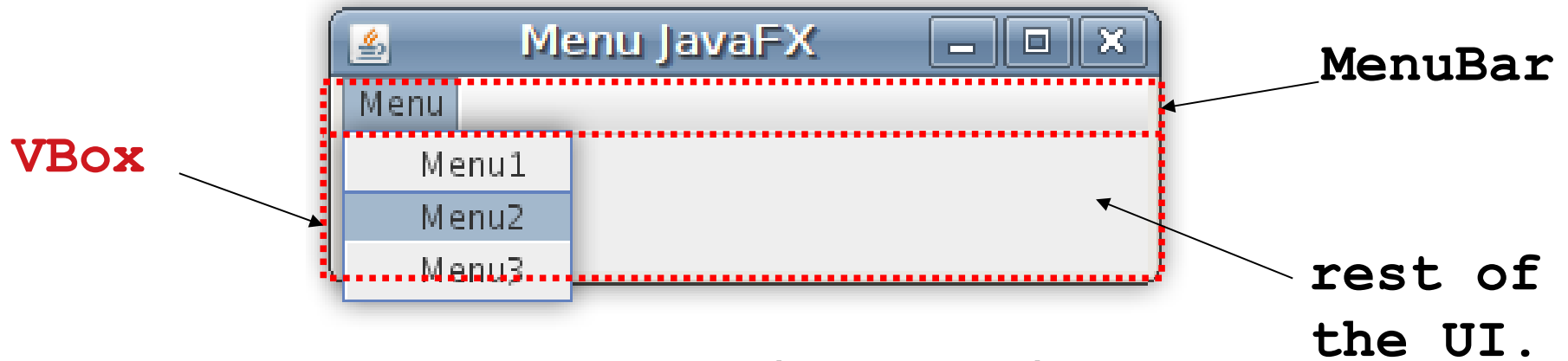
```
prefWidthProperty().bind( scene.getWidthProperty() )
```

```
// make width match the size of the scene or parent
```

# Adding a MenuBar

A JavaFX **MenuBar** is a Control and also a Region.

- Use a Pane (container) as root node of the Scene.
- Add MenuBar to a sub-region of the Pane
- Example using VBox:



**GridPane** or **BorderPane** will also work.



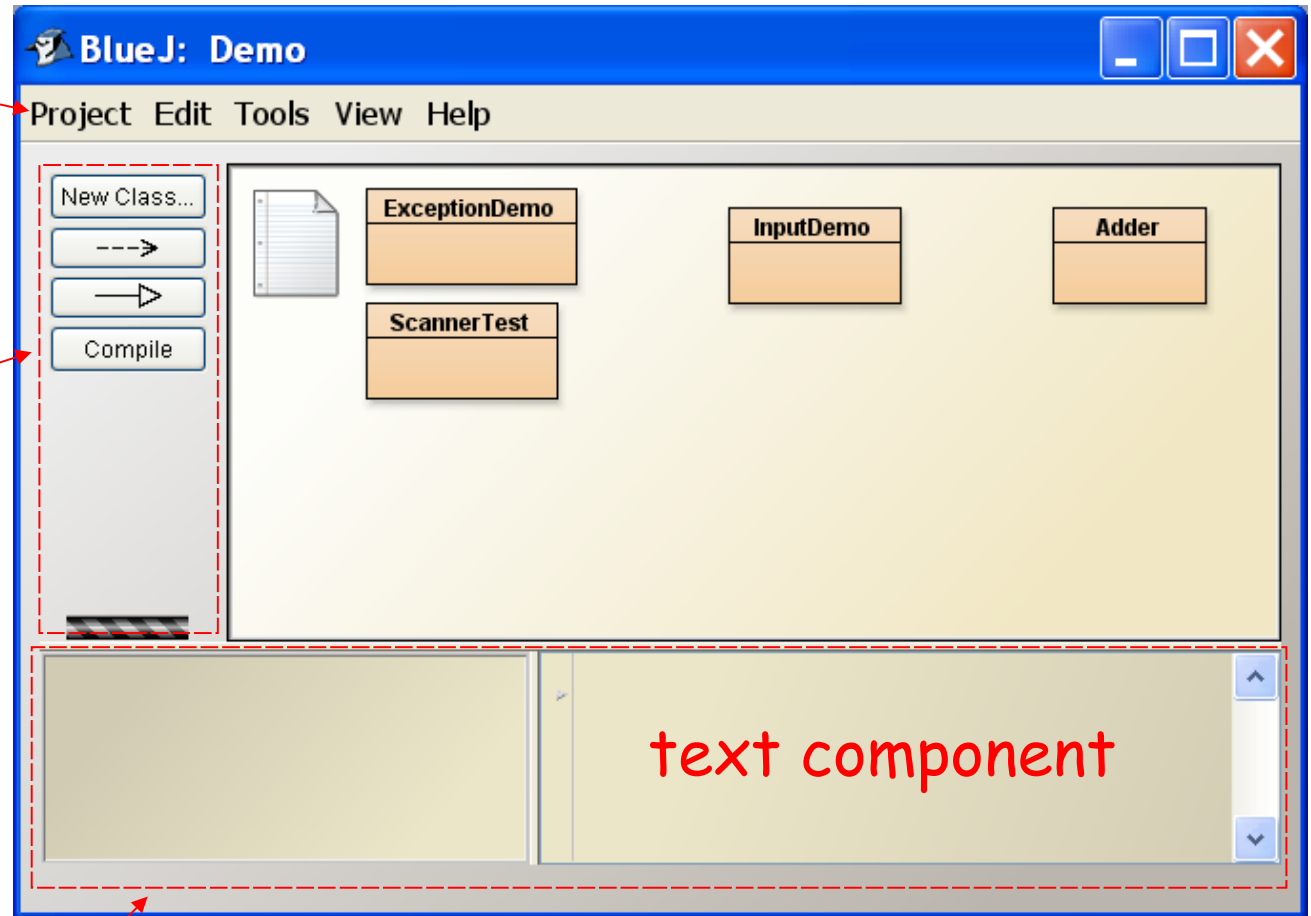
# GridPane with MenuBar & Scene graph

```
public void start(Stage primaryStage) {  
    // container for all the UI controls  
    Pane ui = initComponents();  
    // create MenuBar & add Event Handlers  
    MenuBar menubar = makeMenuBar();  
    // A Layout for MenuBar & UI  
    VBox root = new VBox();  
    root.getChildren().addAll(menubar, ui);  
    // the rest you already know  
    primaryStage.setScene(new Scene(root));  
    // TODO customize scene & stage?  
    primaryStage.show();  
}
```

# BlueJ uses nested containers

MenuBar

Container  
with a row  
of buttons  
and other  
controls



a SplitPane with 2 adjustable regions

# Learn More

---

- **Using Built-in Layouts** (Oracle JavaFX tutorial)  
[https://docs.oracle.com/javafx/8/layout/builtin\\_layouts.htm](https://docs.oracle.com/javafx/8/layout/builtin_layouts.htm)
- **JavaFX Tutorial on Java2s.com** -  
<http://www.java2s.com/Tutorials/Java/JavaFX/index.htm>
- **SceneBuilder**  
visual layout tool - use Panes and Controls, and experiment with setting properties