JavaFX

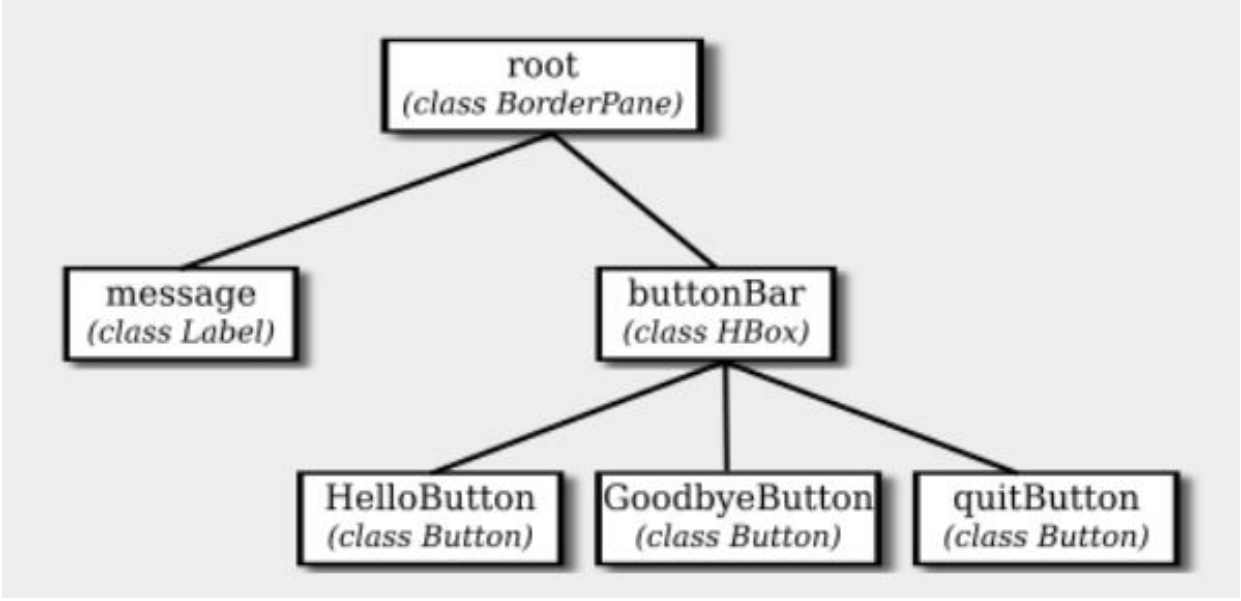
Layouts

Dalton State College

T. Gonzalez

Scene Graph

A **scene graph** is a diagram that shows the containment relationships among all components in a scene.



Nodes

The components that can be in a scene (and hence a scene graph) are referred to as **nodes**.

Nodes are child classes of `javafx.scene.Node`.

Nodes that can act as containers must be child classes of `javafx.scene.Parent` which is a child class of `Node`.

The nodes contained in a container are called **children** of that node.

Scene Graph Implementation

See SceneGraphImplementation.java.

Layout

A layout is a positioning of GUI components in a scene.

Positions and sizes for each component must be determined.

Containers and Nodes

The layout of the child nodes is usually done automatically by the container.

Different containers have different ways for laying out their child nodes.

Every node has a minimum width and height, a maximum width and height, and a preferred width and height.

Containers usually consults these values when laying out its children.

Containers will compute its own preferred size based on the nodes it contains, allowing each child node to have at least its preferred size, and similarly for minimum and maximum sizes.

Resizable nodes, such as controls and most containers, have methods for changing preferred, minimum, and maximum sizes.

Layouts

In JavaFX, containers that do layouts are `javafx.scene.layout.Pane` and its child classes.

Pane

Add nodes to a `Pane` by calling `getChildren().add()` or `addAll()`.

Use the `relocate()` method to set the position of a node by specifying an `x` and `y` coordinate.

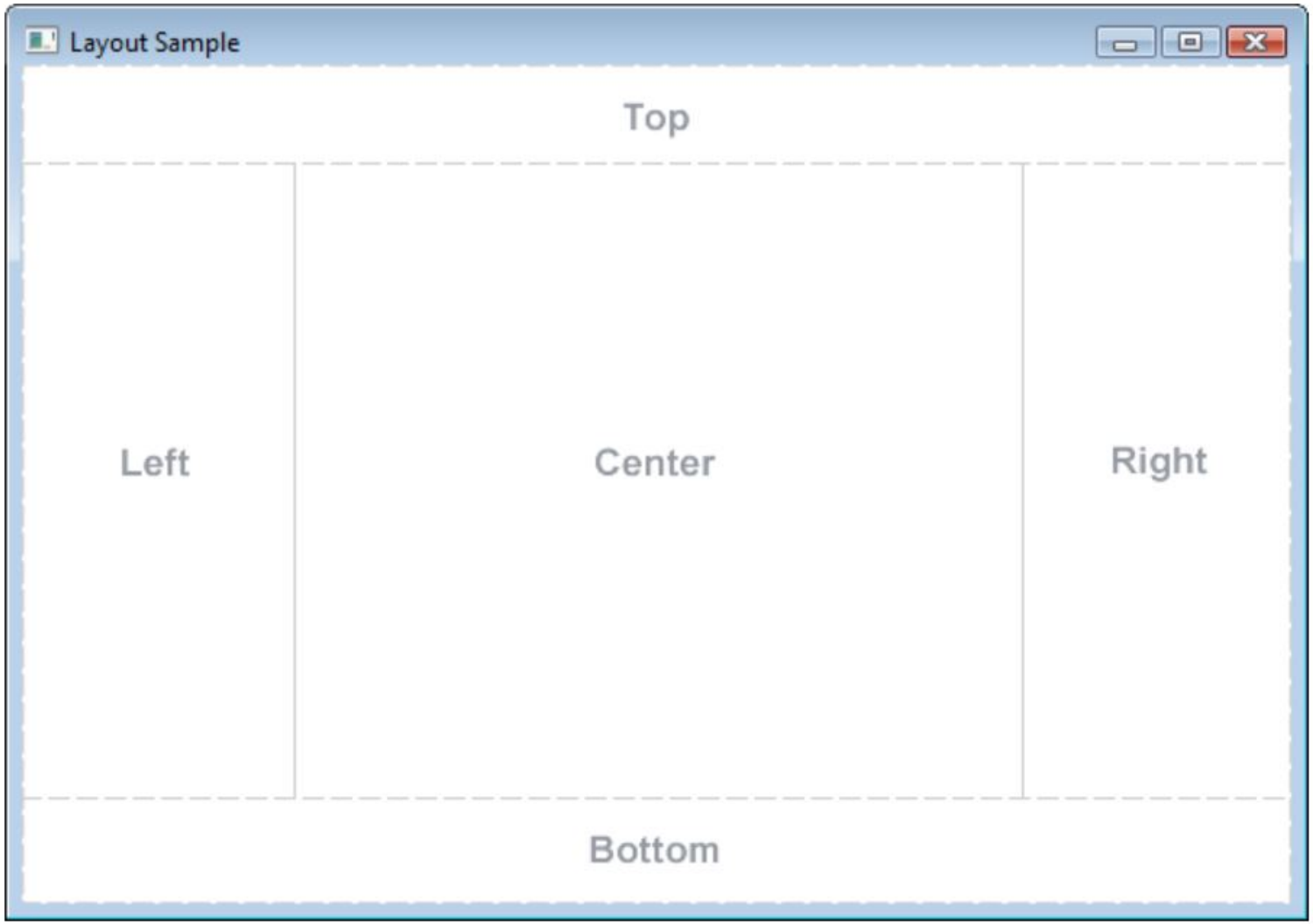
Use the `resize()` method to set the width and height of a resizable node.

Before the `resize()` method will work, the node must call `setManaged(false)`.

See PaneDemo.java.

BorderPane

`BorderPane` lays out its children in top, left, right, bottom, and center positions.



Adding Children to `BorderPane` Compartments

Use the following methods to add child nodes to the `BorderPane` compartments:

- `setTop()`
- `setBottom()`
- `setLeft()`
- `setRight()`
- `setCenter()`

Use these methods with `null` as the argument to remove the child node.

Alignments

Nodes can be aligned within `BorderPane` compartments by using the `static` method call

```
BorderPane.setAlignment( child, position);
```

The position can be any of the values in the `javafx.geometry.Pos` class.

See the documentation for `javafx.geometry.Pos`.

See BorderPaneDemo.java.

HBox

`HBox` lays out it nodes in a horizontal row.

Use the `setSpacing()` method to increase the spacing between nodes.

Use the `getChildren().add()` or `addAll()` method to add child nodes.

VBox

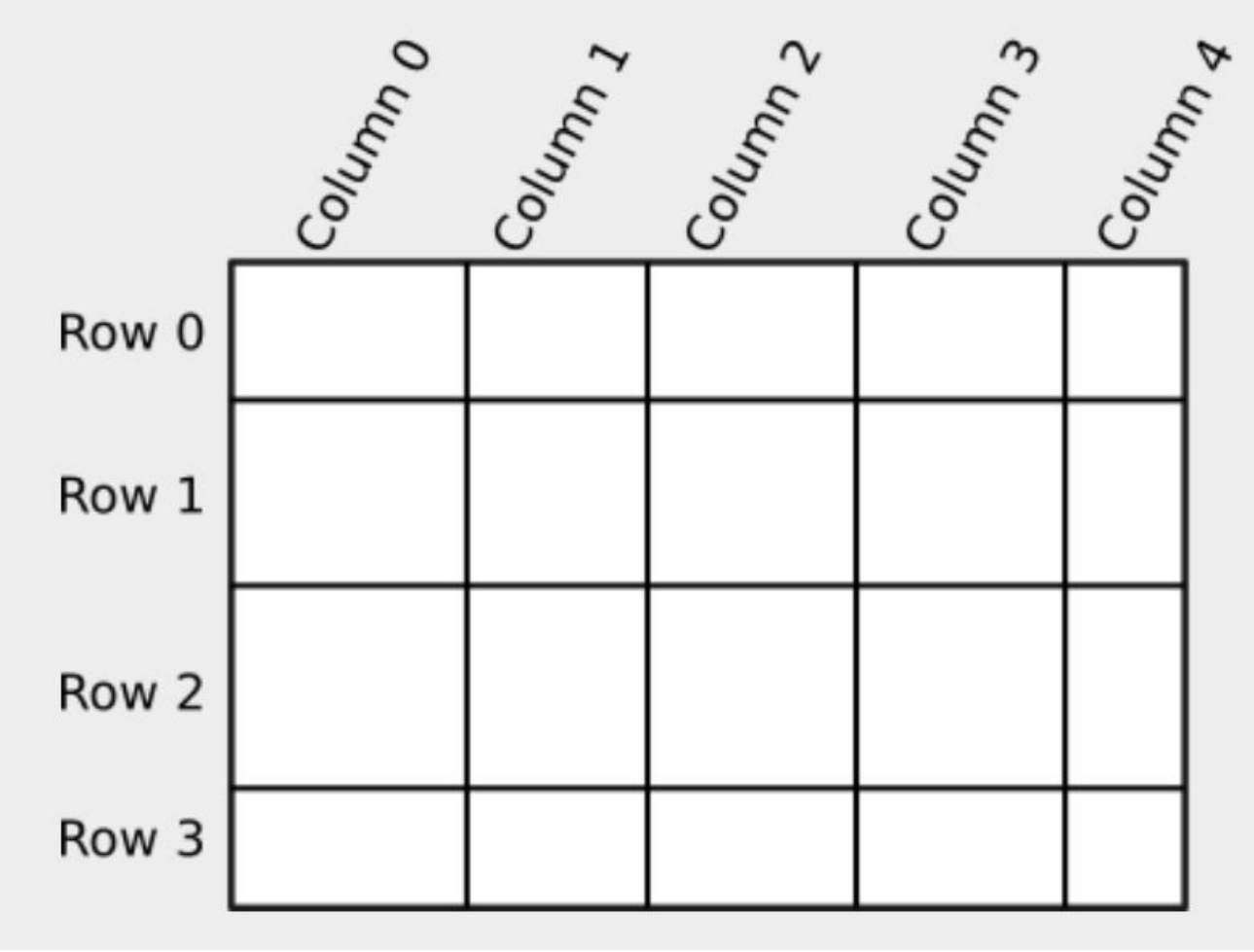
`VBox` lays out it nodes in a vertical column.

Use the `setSpacing()` method to increase the spacing between nodes.

Use the `getChildren().add()` or `addAll()` method to add child nodes.

GridPane

GridPane uses a grid to layout its child nodes in rows and columns.



Adding Nodes to GridPane

```
root.add( child, column, row );

root.add( child, column, row, colspan, rowspan);
```

See GridPaneDemo.java

TilePane

`TilePane` lays out its children in a grid of uniformly sized tiles.

See TilePaneDemo.java.

FlowPane

`FlowPane` lays out its children in a grid of tiles. The size of the tiles does not have to be uniform.

See FlowPaneDemo.java.

In-Class Problem