ORACLE®

JavaOne™

ORACLE®

JavaFX Layout: Everything You Wanted to Know

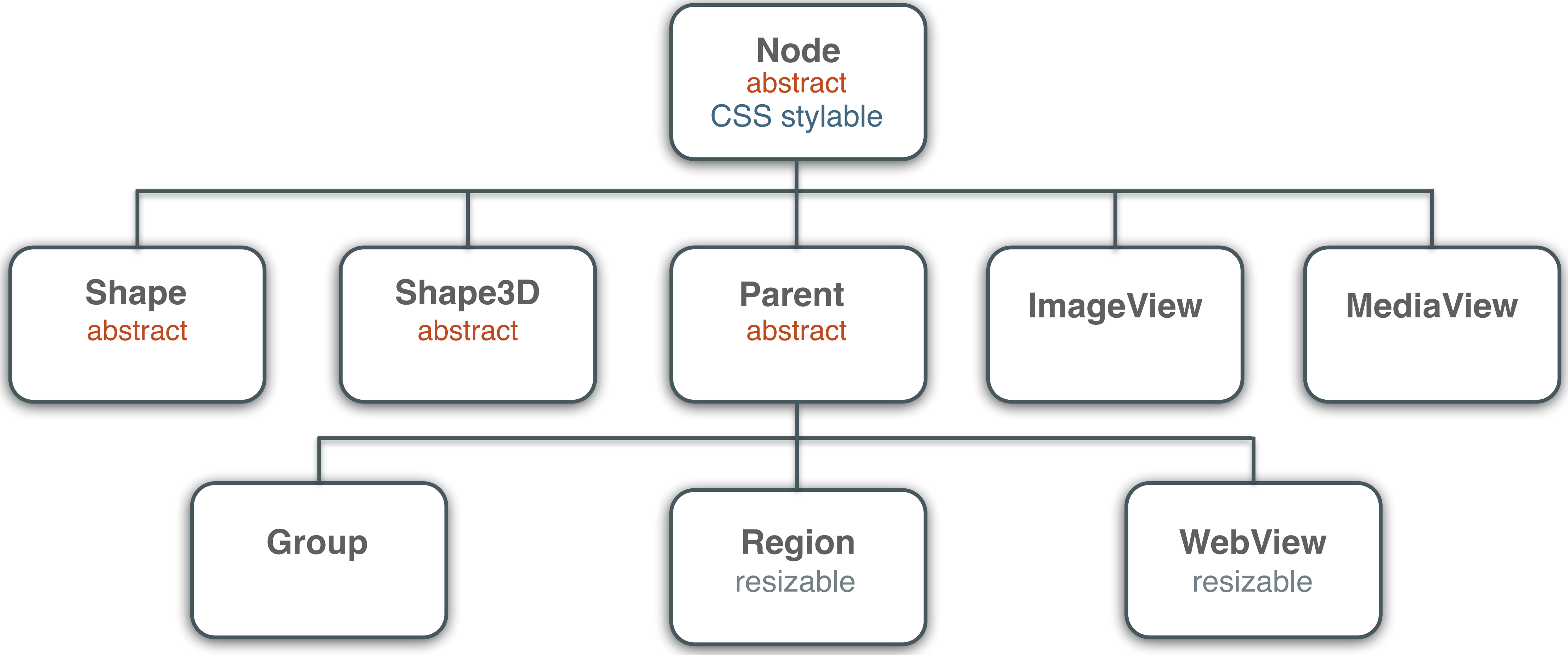
Kevin Rushforth
Chien Yang
Java Client Group, Oracle
October 27, 2015

Safe Harbor Statement

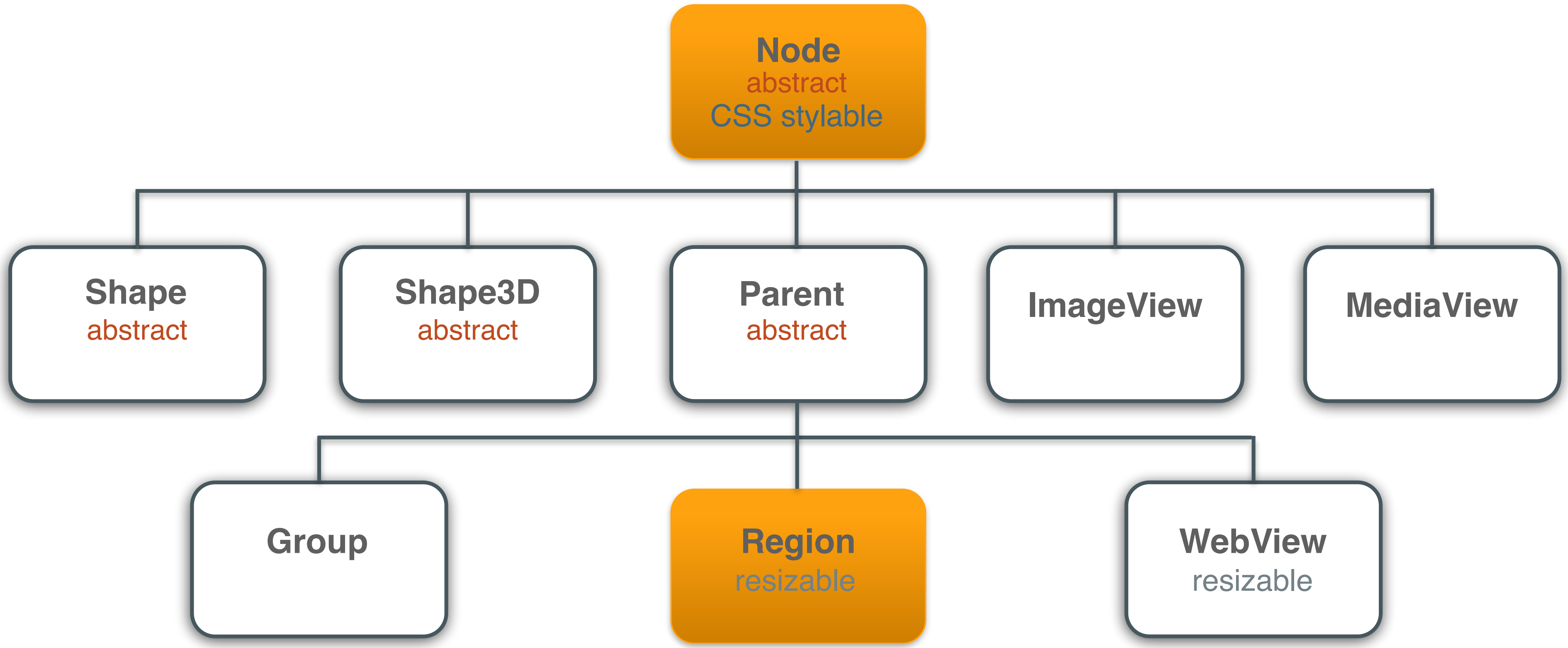
Program Agenda

1. JavaFX Core Scene Graph Classes
2. Common Layout Properties in Node
3. Layout Classes with Demos
4. Custom Layout Pane with Demo
5. Q & A

JavaFX Core Scene Graph Classes



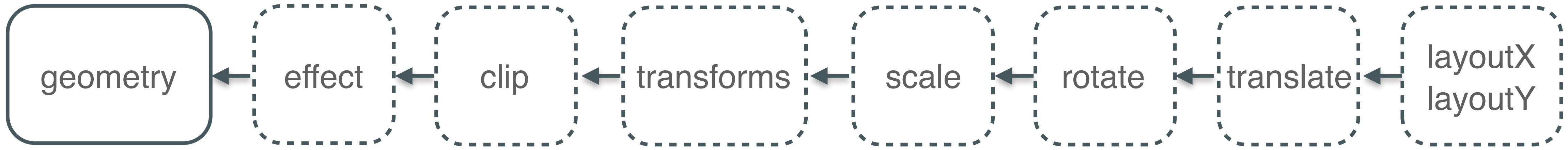
JavaFX Core Scene Graph Classes



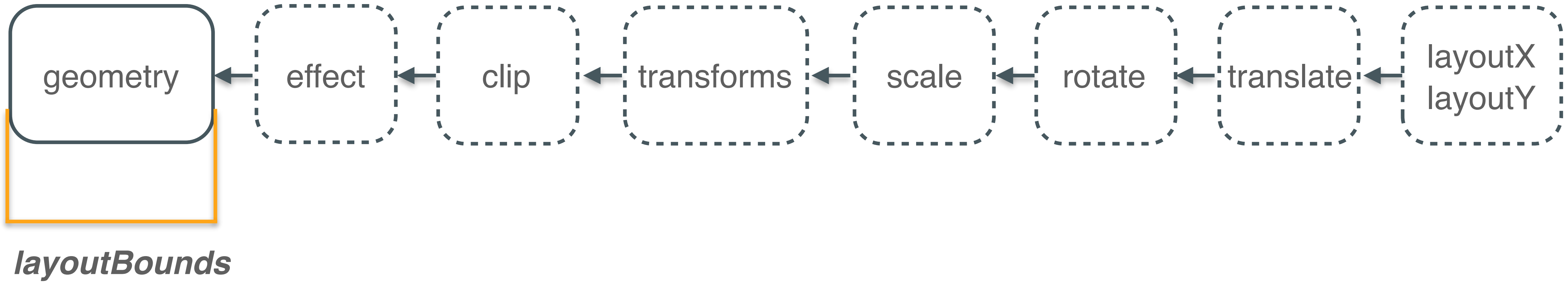
Node: Common Properties and Attributes

- ***layoutBounds***, ***boundInLocal*** and ***boundsInParent***
- resizable vs non-resizable
- ***visible***
- ***managed***
- CSS Stylable

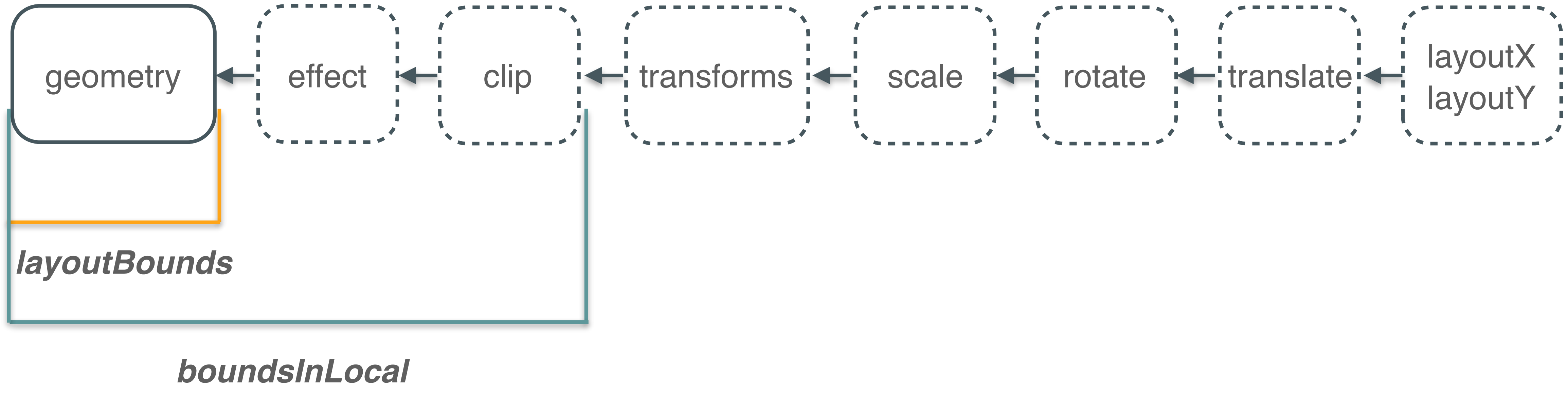
Node: Bounds Properties



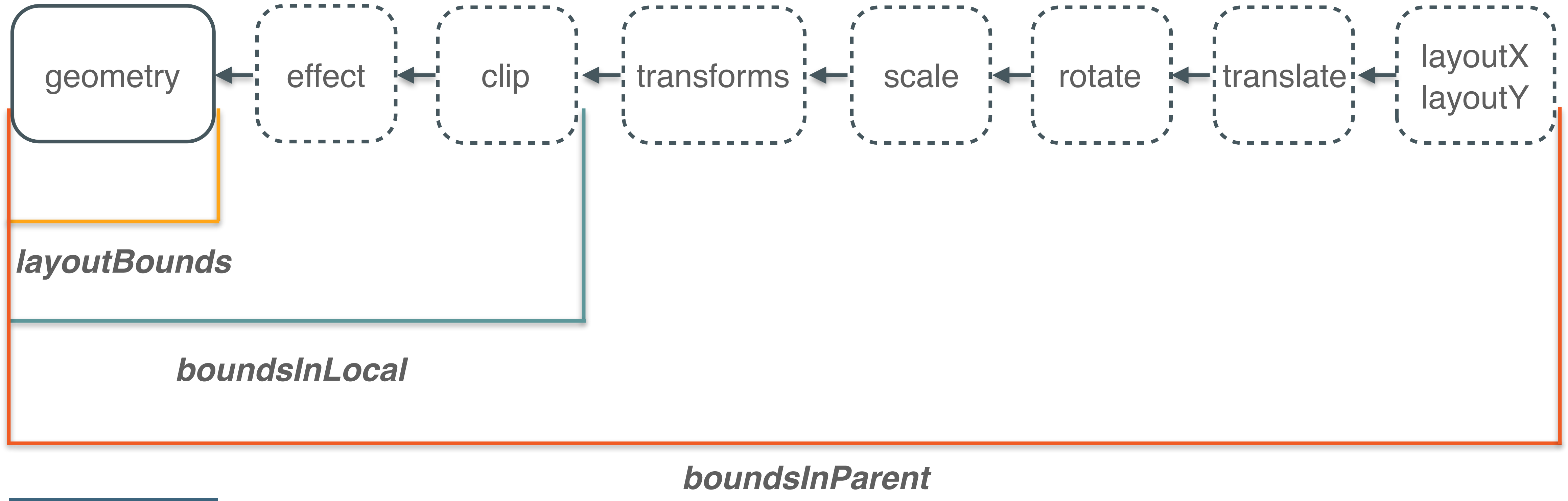
Node: Bounds Properties



Node: Bounds Properties

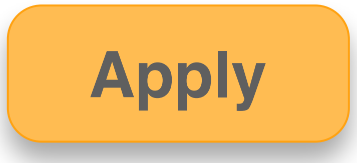


Node: Bounds Properties

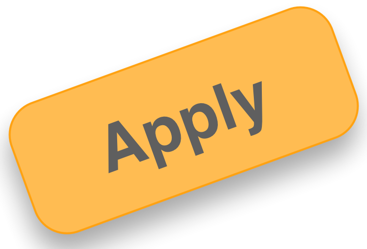


Node: Bounds Properties

Button

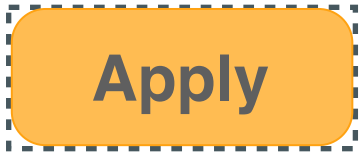


Button with drop shadow



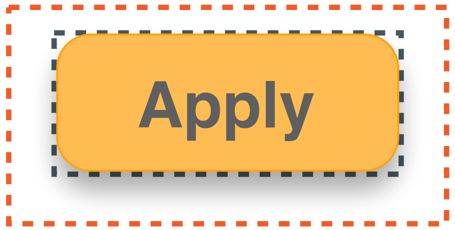
Button with drop shadow and rotation

Bounds Properties



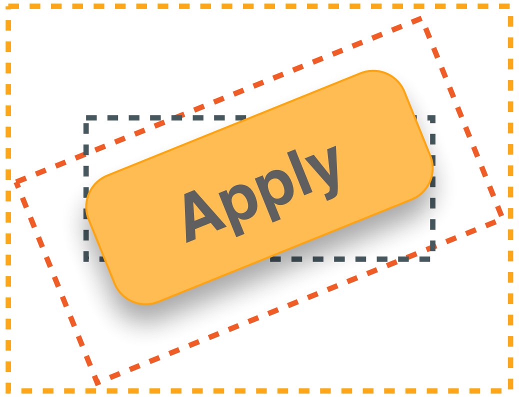
layoutBounds = boundsInLocal
= boundsInParent

Button



layoutBounds
boundsInLocal = boundsInParent

Button with drop shadow



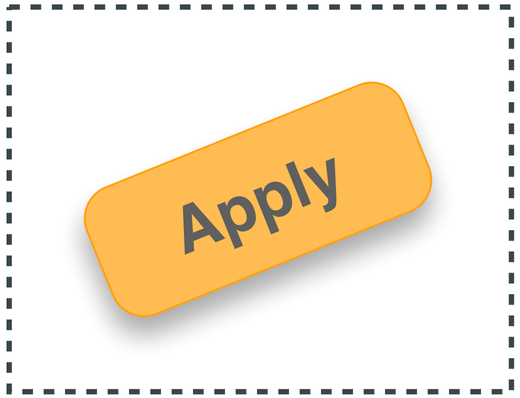
layoutBounds
boundsInLocal
boundsInParent

Button with drop shadow and rotation

Bounds Properties

- Use Group to include effect/clip/transforms in layout
- Group layoutBounds = union of children’s visual bounds

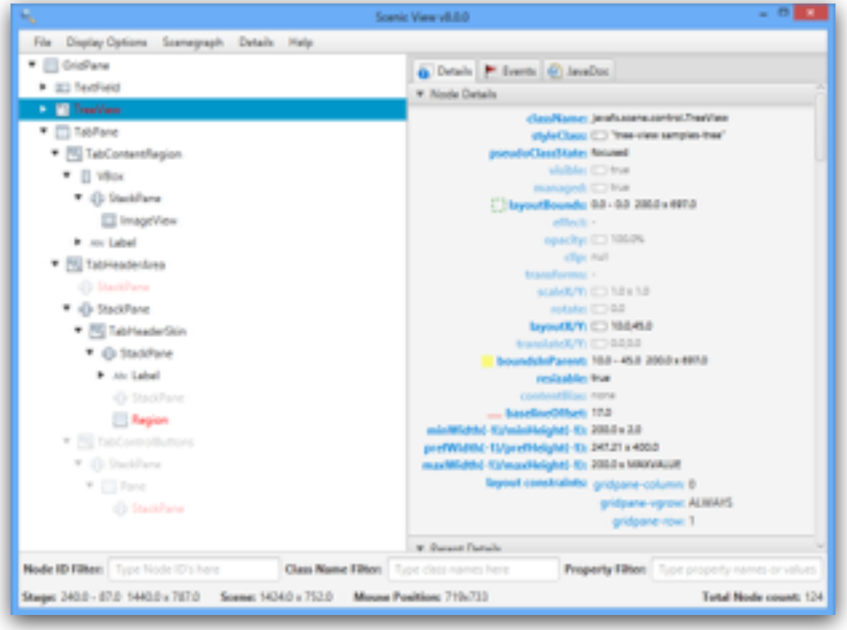
```
Button applyBtn = new Button(“Apply”);
applyBtn(new DropShadow());
applyBtn.setRotate(-22);
layoutRegion.getChildren().add(new Group(applyBtn));
```



Group’s layoutBounds

Scenic View

- Scenic View is a free JavaFX scenegraph analyzer
- It draws overlays of bounds in the application it is observing
- Download and find out more about Scenic View here:
  - http://www.scenic-view.org
- Great tool for debugging scenegraph, especially layout



```
▼ GridPane
  ▶ TextField
  ▶ TreeView
  ▼ TabPane
    ▼ TabContentRegion
      ▼ VBox
        ▼ StackPane
            ImageView
        ▶ Label
    ▼ TabHeaderArea
        StackPane
      ▼ StackPane
        ▼ TabHeaderSkin
          ▼ StackPane
            ▶ Label
              StackPane
              Region
      ▼ TabControlButtons
        ▼ StackPane
          ▼ Pane
              StackPane
```

```
                          className: javafx.scene.control.TreeView
                         styleClass: "tree-view samples-tree"
                   pseudoClassState: focused
                            visible: true
                            managed: true
                       layoutBounds: 0.0 - 0.0  200.0 x 697.0
                             effect: -
                            opacity: 100.0%
                               clip: null
                         transforms: -
                         scaleX/Y/: 1.0 x 1.0
                             rotate: 0.0
                          layoutX/Y: 10.0,45.0
                       translateX/Y: 0.0,0.0
                     boundsInParent: 10.0 - 45.0  200.0 x 697.0
                          resizable: true
                        contentBias: none
                     baselineOffset: 17.0
   minWidth(-1)/minHeight(-1): 200.0 x 2.0
 prefWidth(-1)/prefHeight(-1): 247.21 x 400.0
   maxWidth(-1)/maxHeight(-1): 200.0 x MAXVALUE
                 layout constraints: gridpane-column: 0
                                     gridpane-vgrow: ALWAYS
                                     gridpane-row: 1
```

Node ID Filter: Type Node ID's here   Class Name Filter: Type class names here   Property Filter: Type property names or values

Stage: 240.0 - 87.0 1440.0 x 787.0   Scene: 1424.0 x 752.0   Mouse Position: 710x733   Total Node count: 124

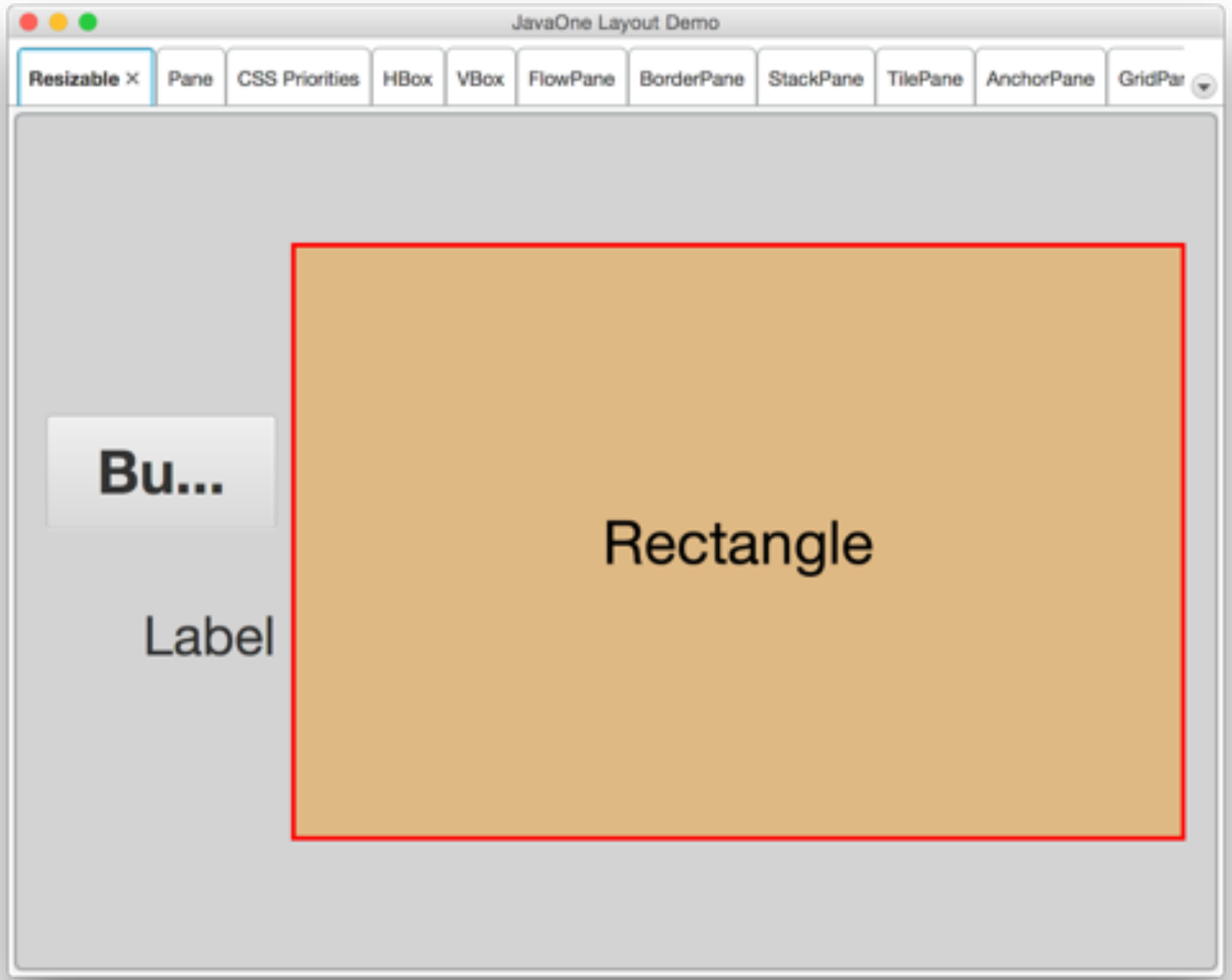
Resizable vs Non-Resizable

- 2 types of nodes: resizable nodes and non-resizable nodes
- **Resizable**: resized by its parent during layout
    - Applications do NOT set size directly
    - Parent makes sizing decision based on own layout policy
- **Non-resizable:** NOT resized by parent during layout
    - Applications set properties to establish size
    - Doesn’t mean node size can’t change!

Resizable vs Non-Resizable

Resizable	Non-Resizable
Region	Group
Pane	Shape
Control	Shape3D
WebView	Text
	ImageView
	MediaView

Demo


JavaOne Layout Demo
Resizable
Pane
CSS Priorities
HBox
VBox
FlowPane
BorderPane
StackPane
TilePane
AnchorPane
GridPar
Bu...
Label
Rectangle

Resizable Nodes

- A resizable node has 3 attributes that define the range of its size:
  - Preferred size
  - Minimum size
  - Maximum size
- Parents query preferred size during layout
- Applications may set range directly

Maximum Size

- Desired max for layout
- All layout regions (except AnchorPane) honor max size
- Max == Double.MAX_VALUE => unbounded
  - Indicates hunger for space if available
- Max == preferred => clamped
  - Indicates desire to be preferred size

Maximum Size

- Examples of default Max Size on common controls

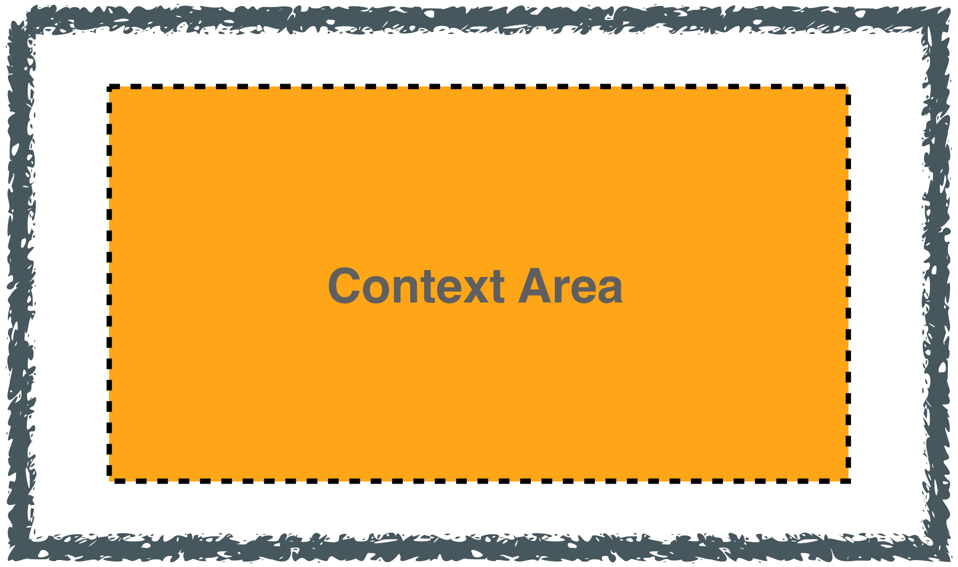
clamped	unbounded
Button	ListView
Label	TreeView
ChoiceBox	TableView
Hyperlink	TabPane
ProgressBar	SplitPane
Slider	ScrollPane
	TextArea

Visible and Managed Properties

- Children layed out regardless of visibility
  - Parent still leaves space for it
- Unmanaged children ignored for layout
  - Child will still be visible, but not resized or relocated by parent
  - Not included in min/pref/max size computations
- If child needs to disappear from layout, must set both ***managed*** and ***visible*** to false

CSS Styleable

- Node is CSS styleable
- CSS is primarily used to style Region nodes
  - https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html



Backgrounds (fills and images)

Borders (strokes and images)

Content Area

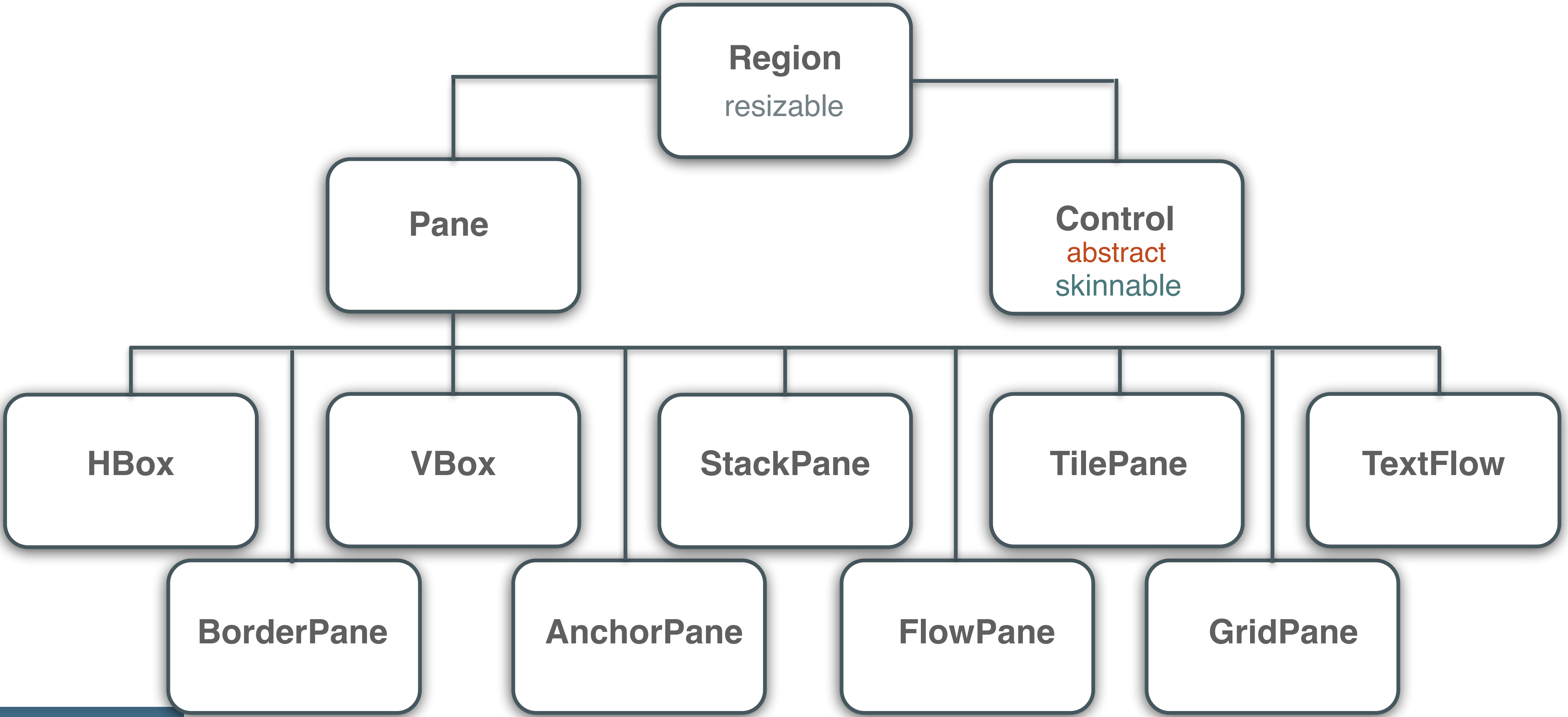
Padding

Priorities of Styles for a Node

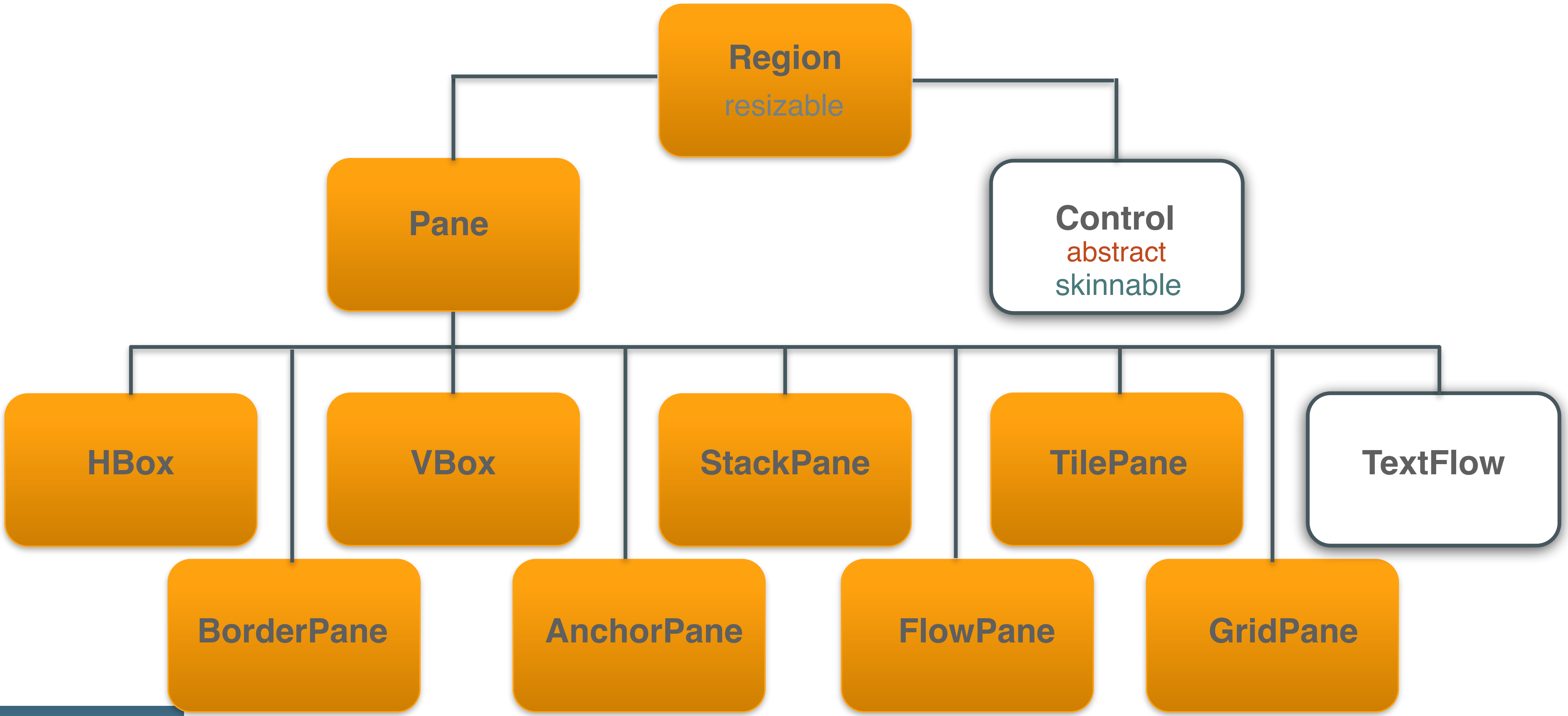
The following priority rules are used to set the visual properties of a node (listed is the order of highest to lowest priority):

- Inline style
- Parent style sheets
- Scene style sheets
- Values set in the code using JavaFX API
- User agent style sheets

Region Extended Classes



Region Extended Classes



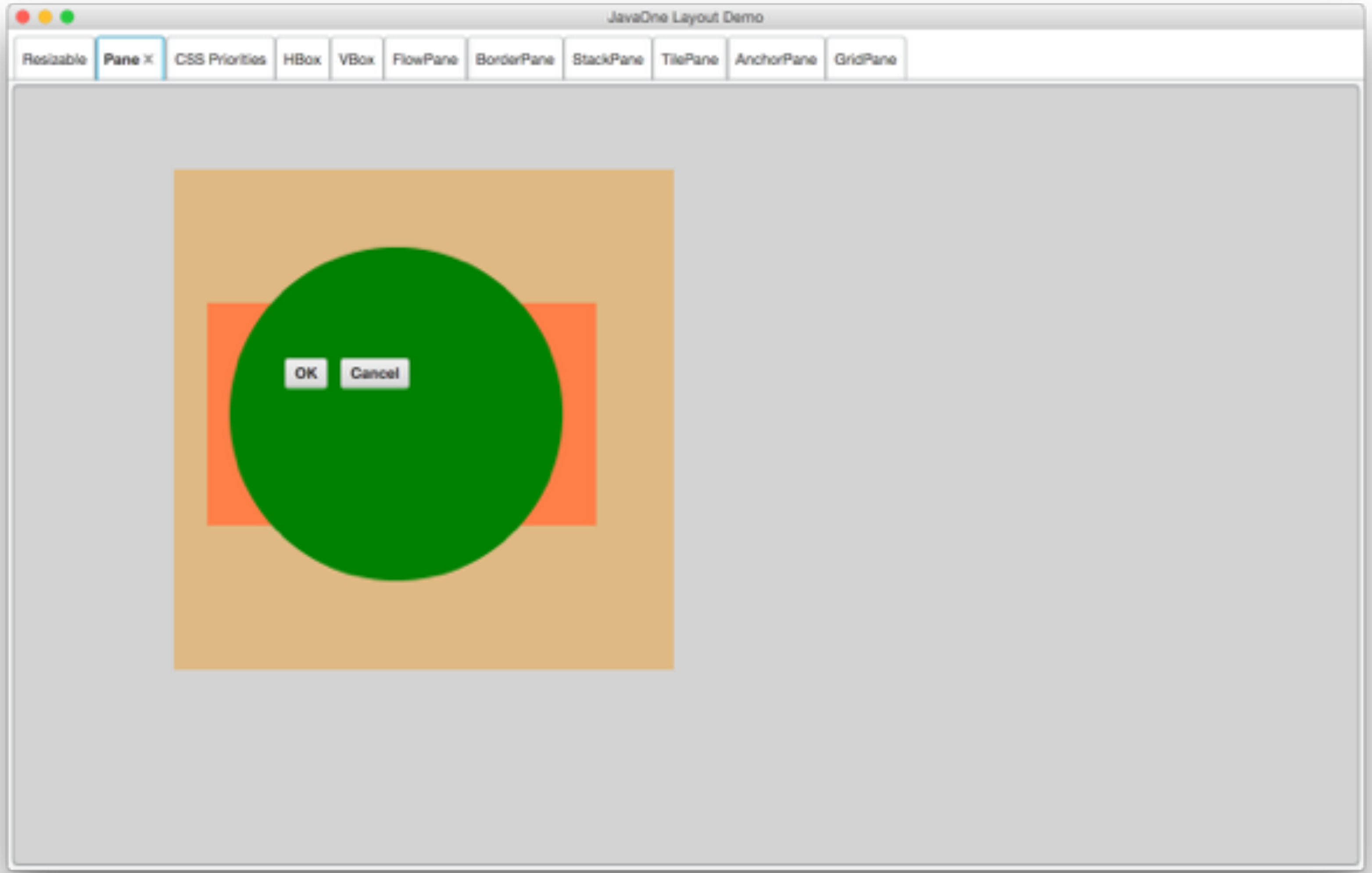
Region

- It is the base class for all layout panes and controls
- It is the highest-level class that is fully CSS-styleable
- List of children is not publicly writable
  - Inherits protected getChildren() method from Parent

Pane

- Pane is a subclass of the Region class
- Exposes the getChildren() method of the Parent class
- Can be used when absolute positioning is needed
  - Positions its children at their (layoutX, layoutY)
- Resizes all resizable children to their preferred sizes
- Pane does not clip its content
  - Its children may be displayed outside its bounds

Demo


JavaOne Layout Demo
Resizable
Pane ×
CSS Priorities
HBox
VBox
FlowPane
BorderPane
StackPane
TilePane
AnchorPane
GridPane
OK
Cancel

HBox

- Lays out children in a single horizontal row
  - Pref width: large enough to display all children at their pref widths
  - Pref height: largest of the pref heights of all its children
- Use properties and constraints to control locations and sizes of children
  - ***alignment***, ***fillHeight***, ***spacing***, setHgrow(), setMargin()

HBox

- Supports 2 types of constraints using static methods
  - hgrow — specifies whether child expands horizontally when there is additional space

```
HBox.setHgrow(node, Priority.ALWAYS);  // expands horizontally

HBox.setHgrow(node, Priority.NEVER);   // don’t expand horizontally
```

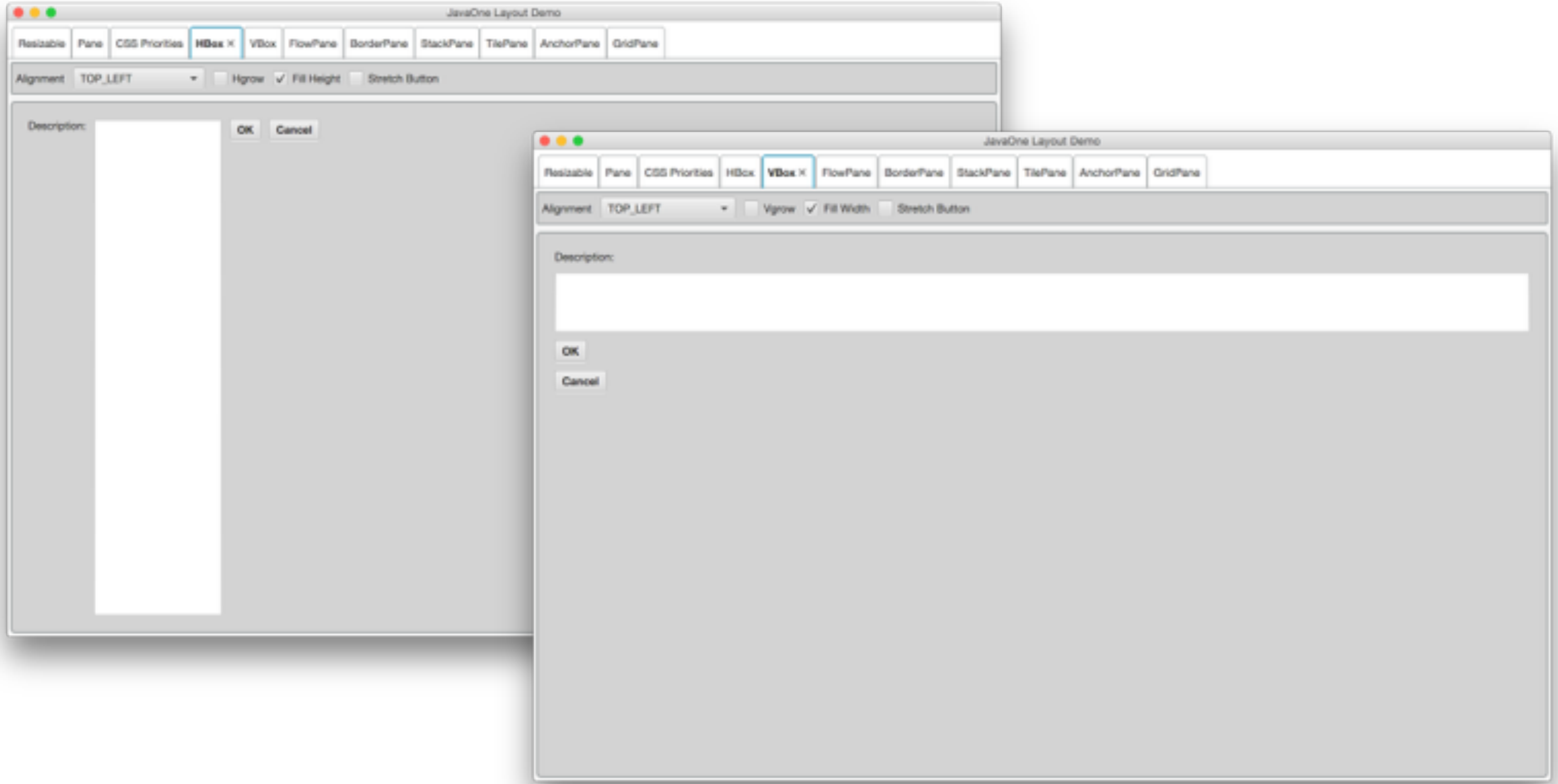
  - margin - specifies space outside the edges of a child node

```
// 6px top, 2px right, 6px bottom, and 2px left
Insets margin = new Insets(6, 2, 6, 2);
HBox.setMargin(okBtn, margin);
```

VBox

- Lays out children in a single vertical column
  - Pref height: large enough to display all children at their pref heights
  - Pref width: largest of the pref widths of all its children
- Use properties and constraints to control the locations and sizes of children
  - ***alignment***, ***fillWidth***, ***spacing***, setVgrow(), setMargin()
- VBox is similar to HBox except in opposite direction

Demo



Alignment TOP_LEFT ☐ Hgrow ✓ Fill Height ☐ Stretch Button

Alignment TOP_LEFT ☐ Vgrow ✓ Fill Width ☐ Stretch Button

FlowPane

- Lays out children in rows or columns wrapping at a specified width or height
- Flow alignment, order and spacing are configurable
  - ***alignment***, ***columnHalignment***, ***rowValignment***
  - ***hgap***, ***vgap***, ***prefWrapLength***, ***nodeOrientation***
- Renders all children at their preferred sizes

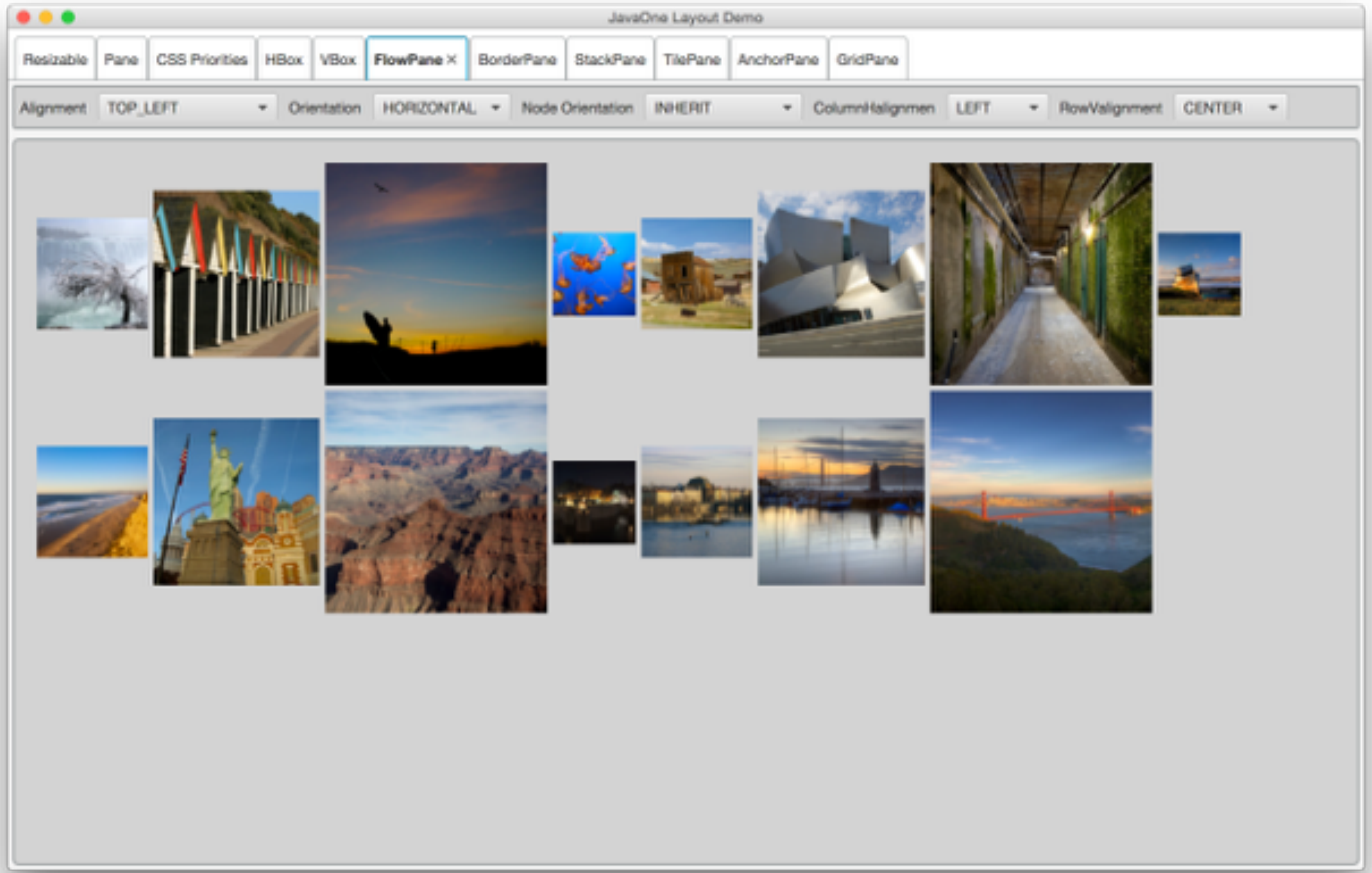
FlowPane

```
// Add photos to the flow pane
for (int i = 1; i < 16; i++) {
    String imageStr = "resources/images/squares" + i + ".jpg";
    int index = i % sizeArr.length;

    flowPane.getChildren().add(new ImageView(new Image(imageStr,
            sizeArr[index], sizeArr[index], true, true)));
}
flowPane.getStyleClass().add(“layout");

Label nodeOrientationLabel = new Label("Node Orientation");
ChoiceBox<NodeOrientation> nodeOrientationCBox = new ChoiceBox<>();
nodeOrientationCBox.getItems().addAll(NodeOrientation.INHERIT,
        NodeOrientation.LEFT_TO_RIGHT, NodeOrientation.RIGHT_TO_LEFT);
nodeOrientationCBox.getSelectionModel().select(flowPane.getNodeOrientation());
nodeOrientationCBox.getSelectionModel().selectedItemProperty().addListener(
        this::nodeOrientationChanged);
```

Demo


JavaOne Layout Demo
Resizable
Pane
CSS Priorities
HBox
VBox
FlowPane ×
BorderPane
StackPane
TilePane
AnchorPane
GridPane
Alignment
TOP_LEFT
Orientation
HORIZONTAL
Node Orientation
INHERIT
ColumnHalignmen
LEFT
RowValignment
CENTER

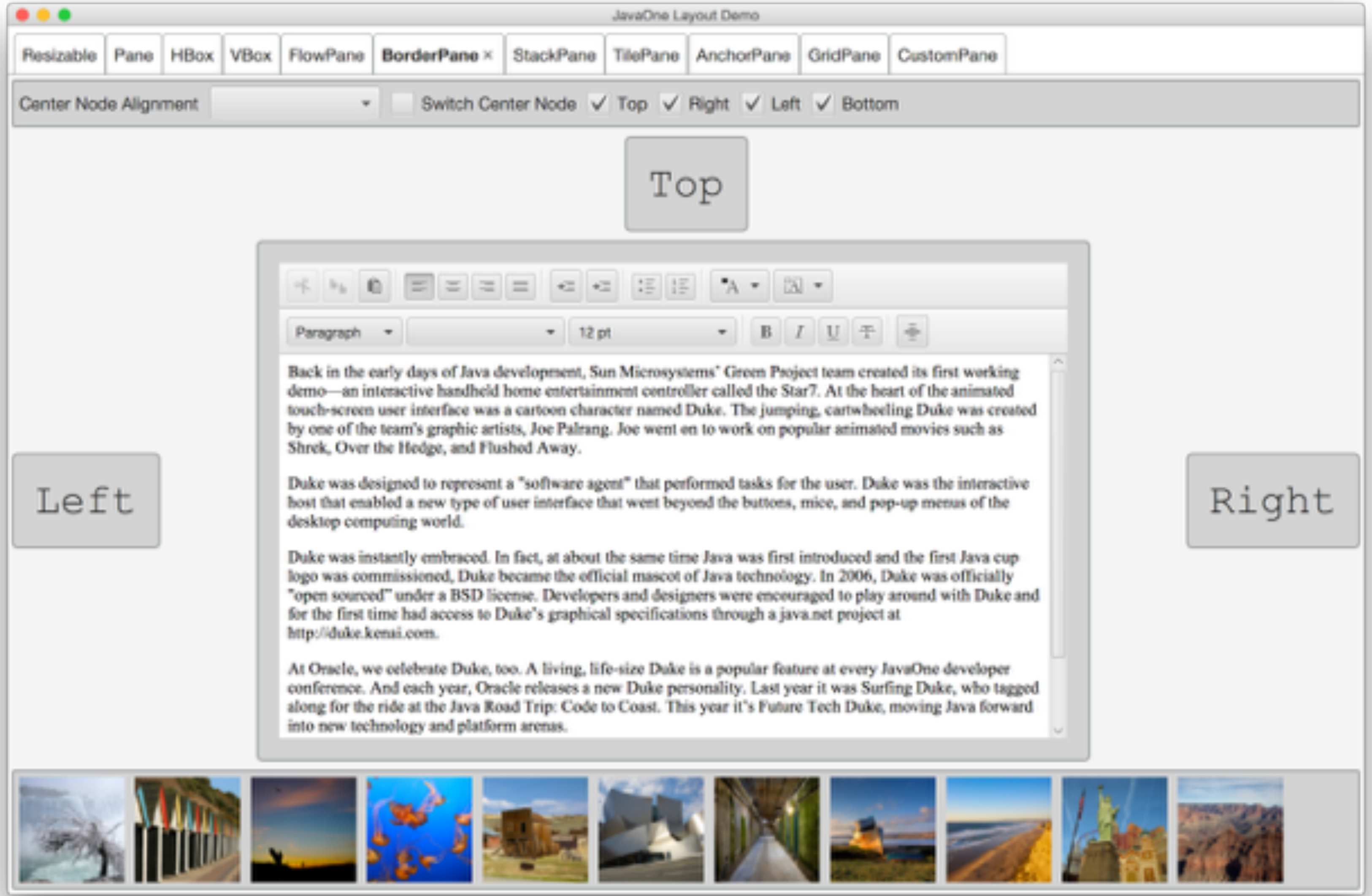
BorderPane

- Divides layout area into five regions:
  - ***top***, ***right***, ***bottom***, ***left***, and ***center***
- Resizing policies for children:
  - top and bottom - preferred height, width extended
  - left and right - preferred width, height extended
  - center - fill the rest of the available space

BorderPane

- Use ***top***, ***right***, ***bottom***, ***left***, and ***center*** properties to set children
  - Do not add children via the getChildren() method
- Not all of the five positions need to have nodes
- Set to null to remove a child node from a position

Demo



Center Node Alignment
Switch Center Node
✓ Top ✓ Right ✓ Left ✓ Bottom

Back in the early days of Java development, Sun Microsystems' Green Project team created its first working demo—an interactive handheld home entertainment controller called the Star7. At the heart of the animated touch-screen user interface was a cartoon character named Duke. The jumping, cartwheeling Duke was created by one of the team's graphic artists, Joe Palrang. Joe went on to work on popular animated movies such as Shrek, Over the Hedge, and Flushed Away.

Duke was designed to represent a "software agent" that performed tasks for the user. Duke was the interactive host that enabled a new type of user interface that went beyond the buttons, mice, and pop-up menus of the desktop computing world.

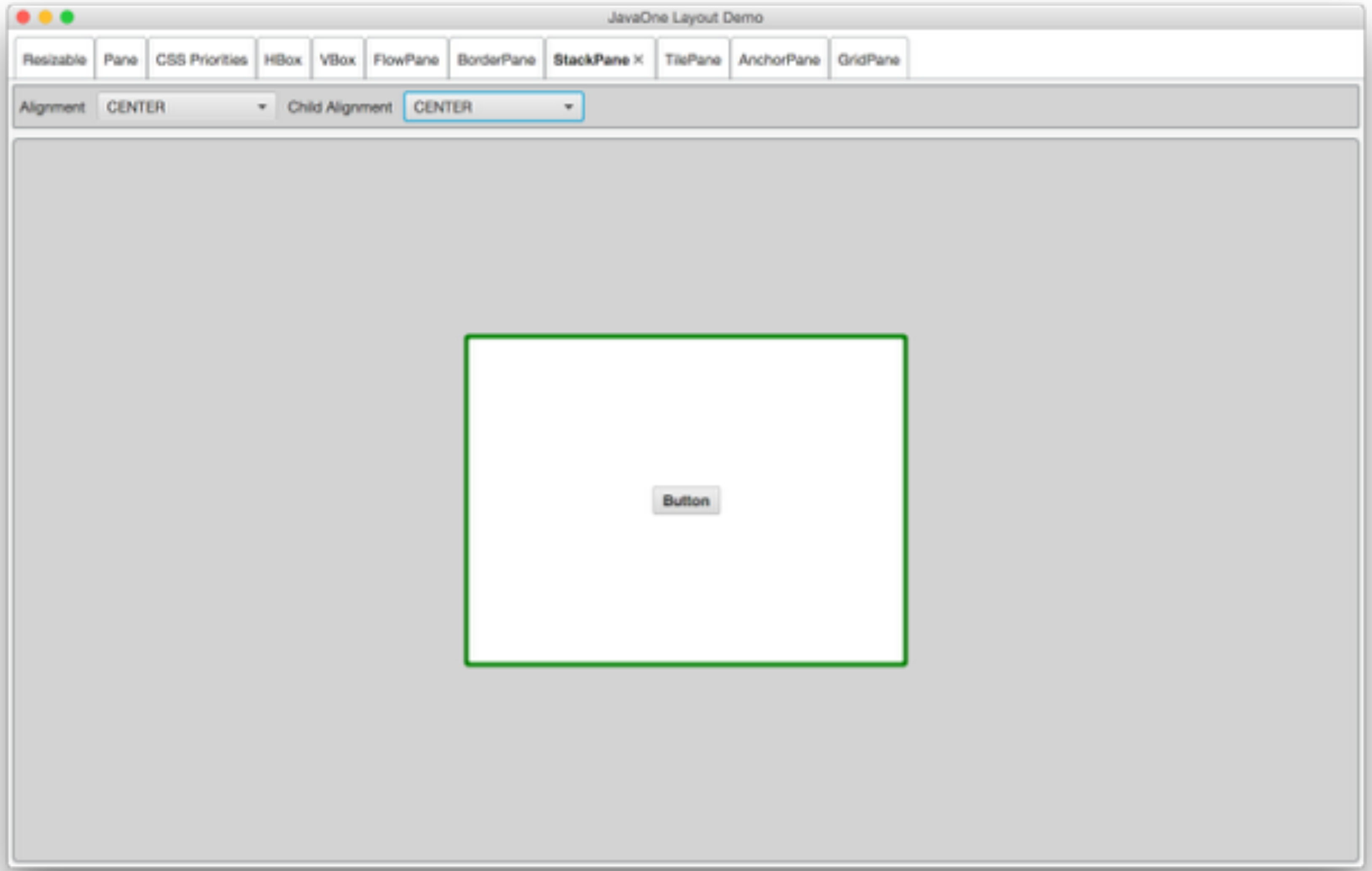
Duke was instantly embraced. In fact, at about the same time Java was first introduced and the first Java cup logo was commissioned, Duke became the official mascot of Java technology. In 2006, Duke was officially "open sourced" under a BSD license. Developers and designers were encouraged to play around with Duke and for the first time had access to Duke's graphical specifications through a java.net project at http://duke.kenai.com.

At Oracle, we celebrate Duke, too. A living, life-size Duke is a popular feature at every JavaOne developer conference. And each year, Oracle releases a new Duke personality. Last year it was Surfing Duke, who tagged along for the ride at the Java Road Trip: Code to Coast. This year it's Future Tech Duke, moving Java forward into new technology and platform arenas.

StackPane

- Lays out children in a stack of nodes
- Preferred width is the width of its widest child
- Preferred height is the height of its tallest child
- Alignment is configurable
- Resizes children to fill (up to their max limit)

Demo



Alignment CENTER Child Alignment CENTER

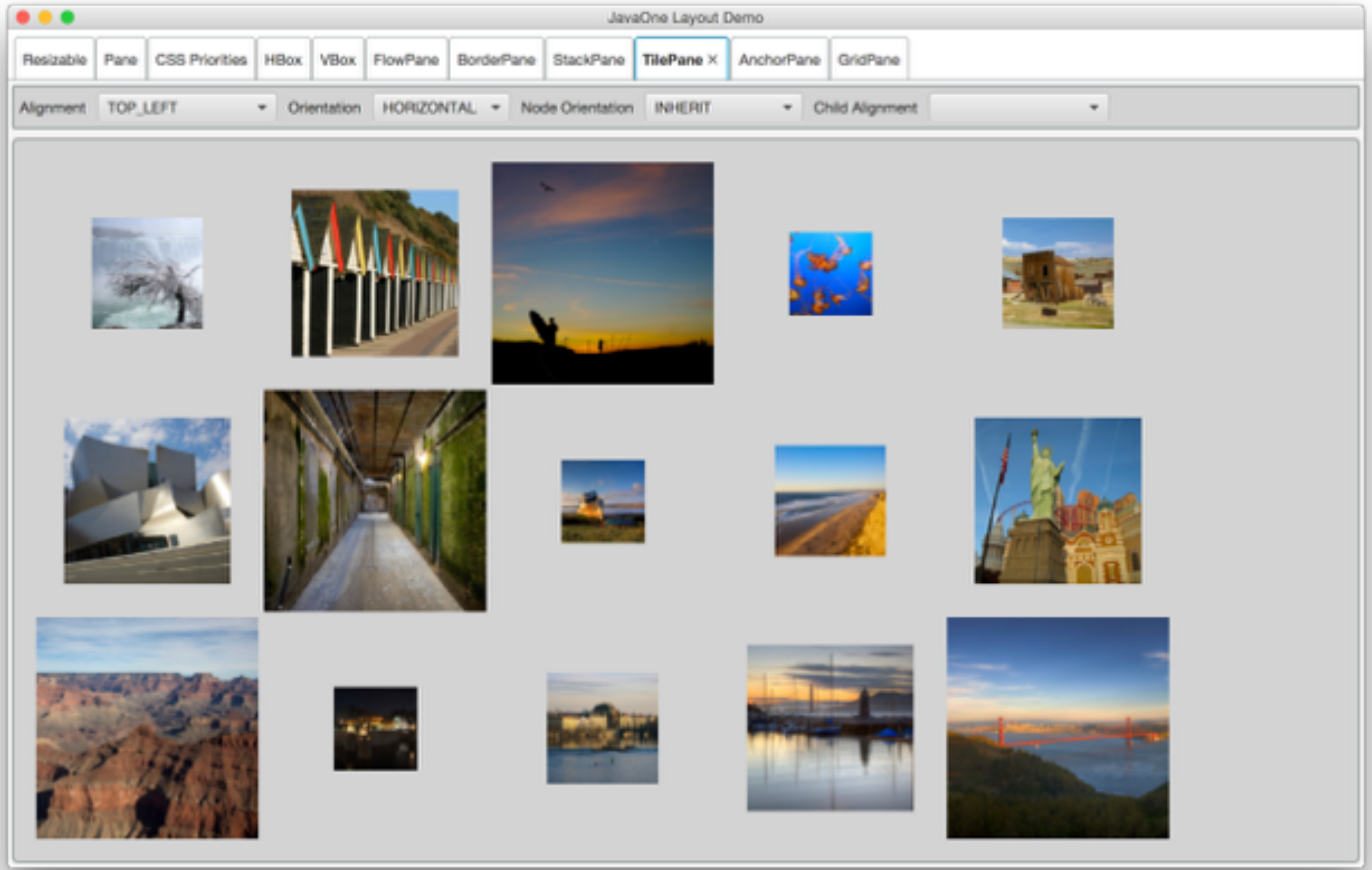
TilePane

- Lays out children in a grid of uniformly sized cells known as tiles
- Works similarly to FlowPane except all rows have same height and columns have same width
- Alignment of flows and within tiles are configurable

TilePane

- 3 types of alignment attributes
  - ***alignment*** property affects the content of TilePane as a whole
  - ***tileAlignment*** property affects the alignment of all children within their tiles
  - TilePane.setAlignment(Node, Pos) affects the alignment of the child node within its tile

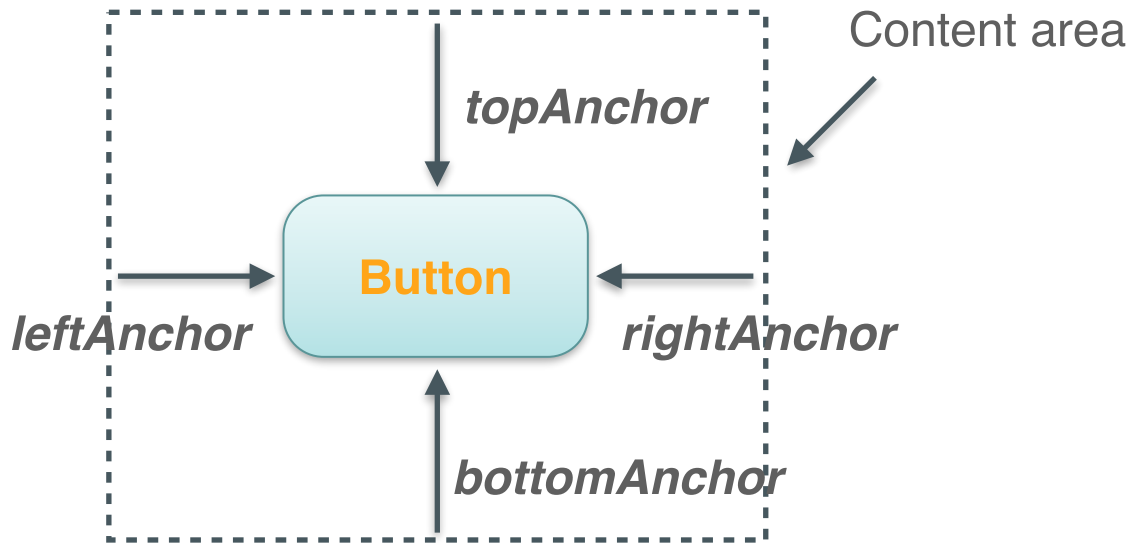
Demo



Alignment TOP_LEFT Orientation HORIZONTAL Node Orientation INHERIT Child Alignment

AnchorPane

- Lays out children by allowing their edges to be anchored to parent's
- Anchor distance is measured from the edges of the content area of the AnchorPane and the edges of the children



AnchorPane

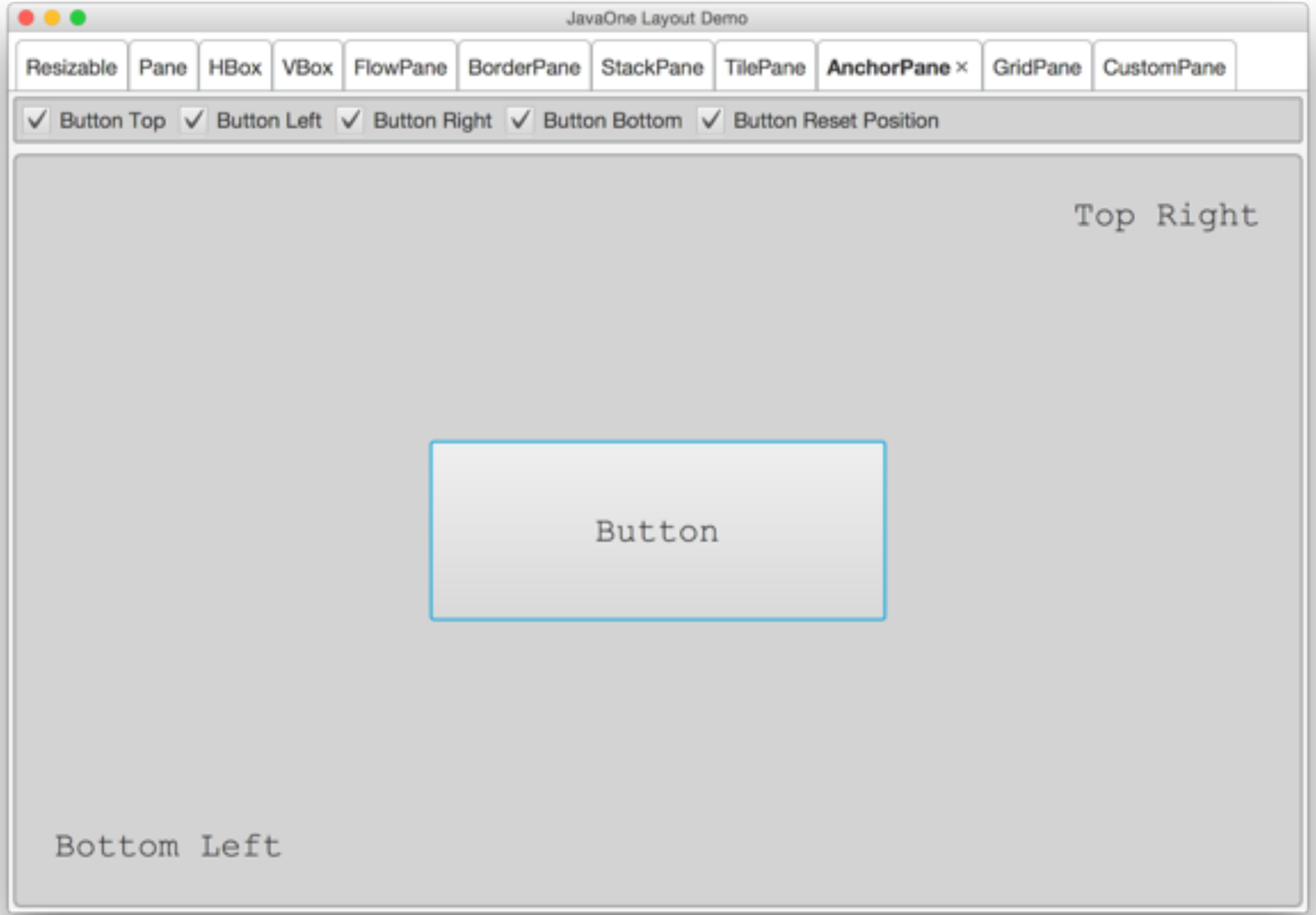
- AnchorPane may be used for two purposes:
  - Aligning children along one or more edges of the layout

```
AnchorPane.setTopAnchor(topRight, 10.0);
AnchorPane.setRightAnchor(topRight, 10.0);

AnchorPane.setBottomAnchor(bottomLeft, 10.0);
AnchorPane.setLeftAnchor(bottomLeft, 10.0);
```

  - Stretching children when the layout is resized
    - Opposing edges are anchored

Demo


JavaOne Layout Demo
Resizable
Pane
HBox
VBox
FlowPane
BorderPane
StackPane
TilePane
AnchorPane ×
GridPane
CustomPane
✓ Button Top
✓ Button Left
✓ Button Right
✓ Button Bottom
✓ Button Reset Position
Top Right
Button
Bottom Left

GridPane

- Lays out children within a flexible grid of rows and columns
- Best suited for creating forms or any layout that is organized in rows and columns
- A child may be placed anywhere within the grid and may span multiple rows/columns. Its placement within the grid is defined by it's layout constraints:
  - columnIndex, rowIndex, columnSpan, rowSpan
- Total number of rows/columns does not need to be specified up front It will automatically expand/contract to accommodate its content.

GridPane: convenience methods

- Use convenience methods that combine setting of constraints and adding the children
  - add, addRow, and addColumn

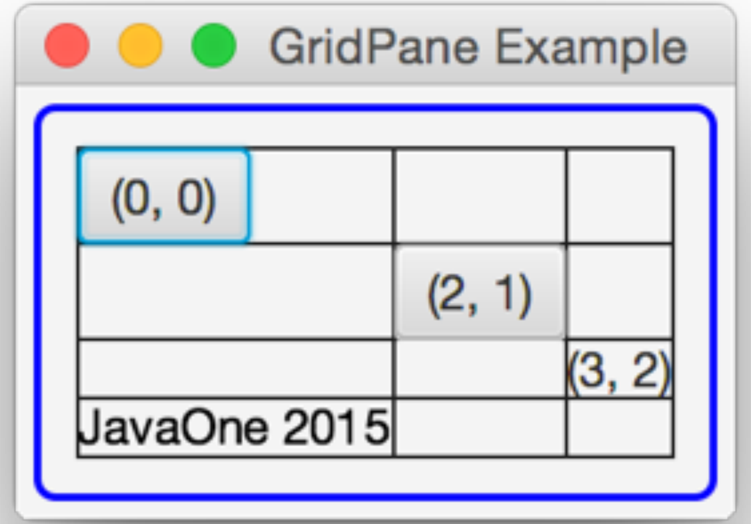
```
GridPane gridpane = new GridPane();

gridpane.add(new Button("(0, 0)"), 0, 0); // column=0 row=0

gridpane.add(new Button(), 2, 1); // column=2 row=1

gridpane.add(new Label(), 3, 2);  // column=3 row=2

gridpane.addRow(3, new Text(“JavaOne 2015”)); // column=0, row=3
```



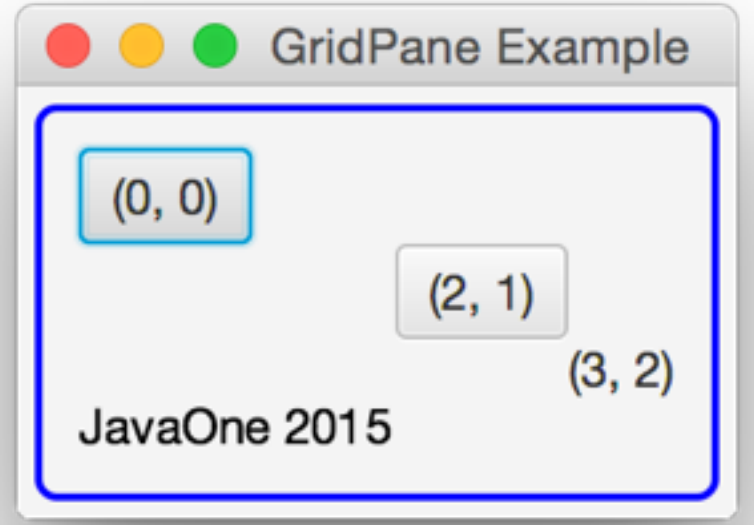
GridPane: Sizing

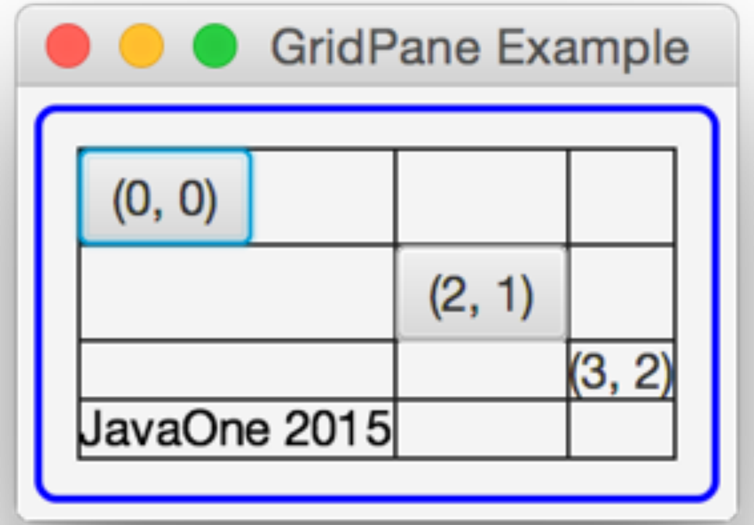
- Supports 3 types of row/column sizing
  - Sized to content: Rows and columns sized to fit their content
  - Fixed: columnConstraints, rowConstraints
  - Percentage: setPercentWidth, setPercentHeight
- GridPane.REMAINING means that the child node spans the remaining columns or remaining rows

```
GridPane.setColumnSpan(descText, GridPane.REMAINING);
```

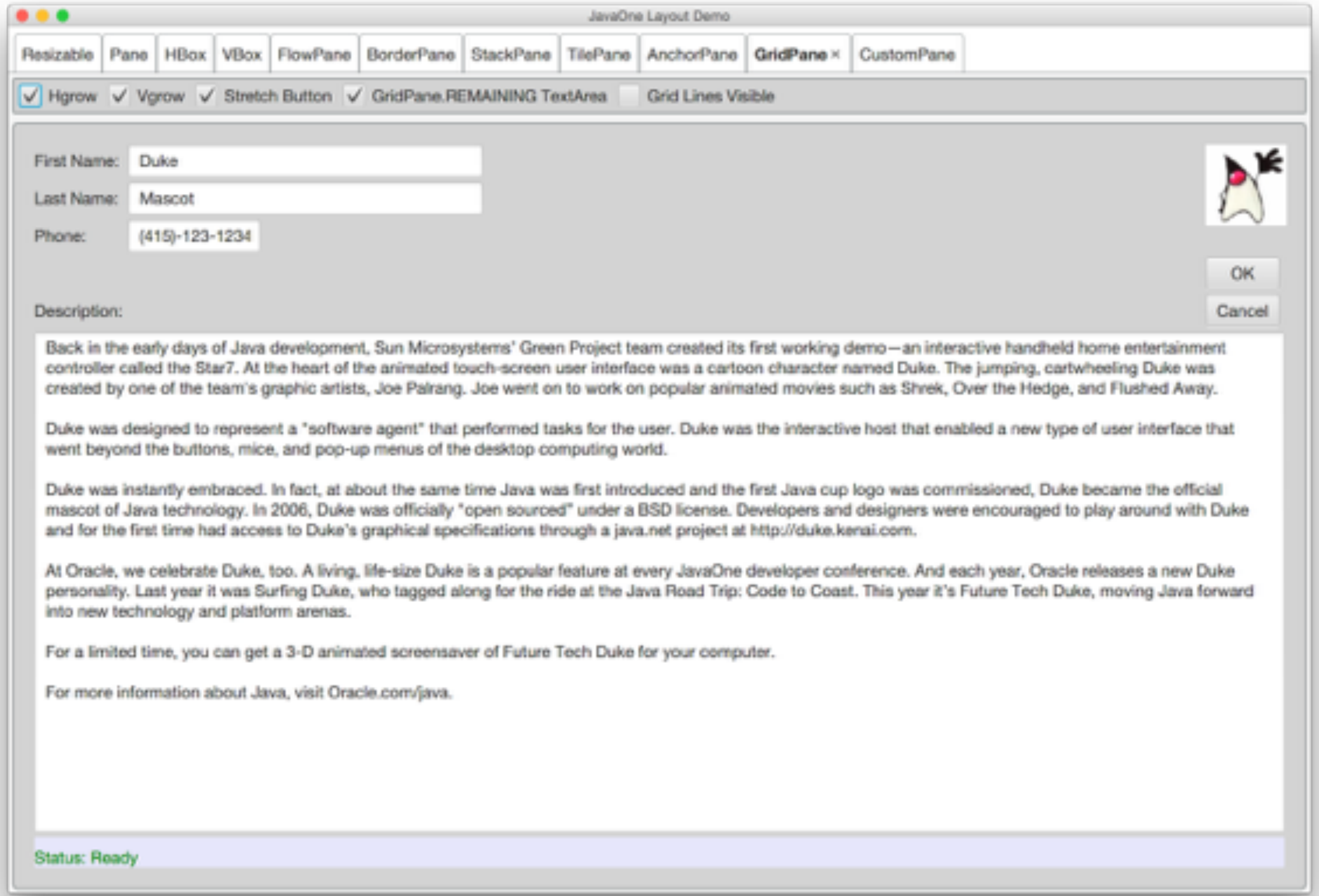
GridPane: Debugging Feature

- This feature is primarily for debug purposes
  - gridPane.setGridLinesVisible(true); // Make grid lines visible





Demo



✓ Hgrow ✓ Vgrow ✓ Stretch Button ✓ GridPane.REMAINING TextArea ☐ Grid Lines Visible

First Name: Duke

Last Name: Mascot

Phone: (415)-123-1234

Description:

Back in the early days of Java development, Sun Microsystems' Green Project team created its first working demo—an interactive handheld home entertainment controller called the Star7. At the heart of the animated touch-screen user interface was a cartoon character named Duke. The jumping, cartwheeling Duke was created by one of the team's graphic artists, Joe Palrang. Joe went on to work on popular animated movies such as Shrek, Over the Hedge, and Flushed Away.

Duke was designed to represent a "software agent" that performed tasks for the user. Duke was the interactive host that enabled a new type of user interface that went beyond the buttons, mice, and pop-up menus of the desktop computing world.

Duke was instantly embraced. In fact, at about the same time Java was first introduced and the first Java cup logo was commissioned, Duke became the official mascot of Java technology. In 2006, Duke was officially "open sourced" under a BSD license. Developers and designers were encouraged to play around with Duke and for the first time had access to Duke's graphical specifications through a java.net project at http://duke.kenai.com.

At Oracle, we celebrate Duke, too. A living, life-size Duke is a popular feature at every JavaOne developer conference. And each year, Oracle releases a new Duke personality. Last year it was Surfing Duke, who tagged along for the ride at the Java Road Trip: Code to Coast. This year it's Future Tech Duke, moving Java forward into new technology and platform arenas.

For a limited time, you can get a 3-D animated screensaver of Future Tech Duke for your computer.

For more information about Java, visit Oracle.com/java.

Status: Ready

Custom Layout

- Make sure no existing layout pane meets your requirements
- Create a subclass of Region or Pane and override layoutChildren()

```
@Override protected void layoutChildren() { …. }
```

- May need to override computeMin/Pref/Max size methods

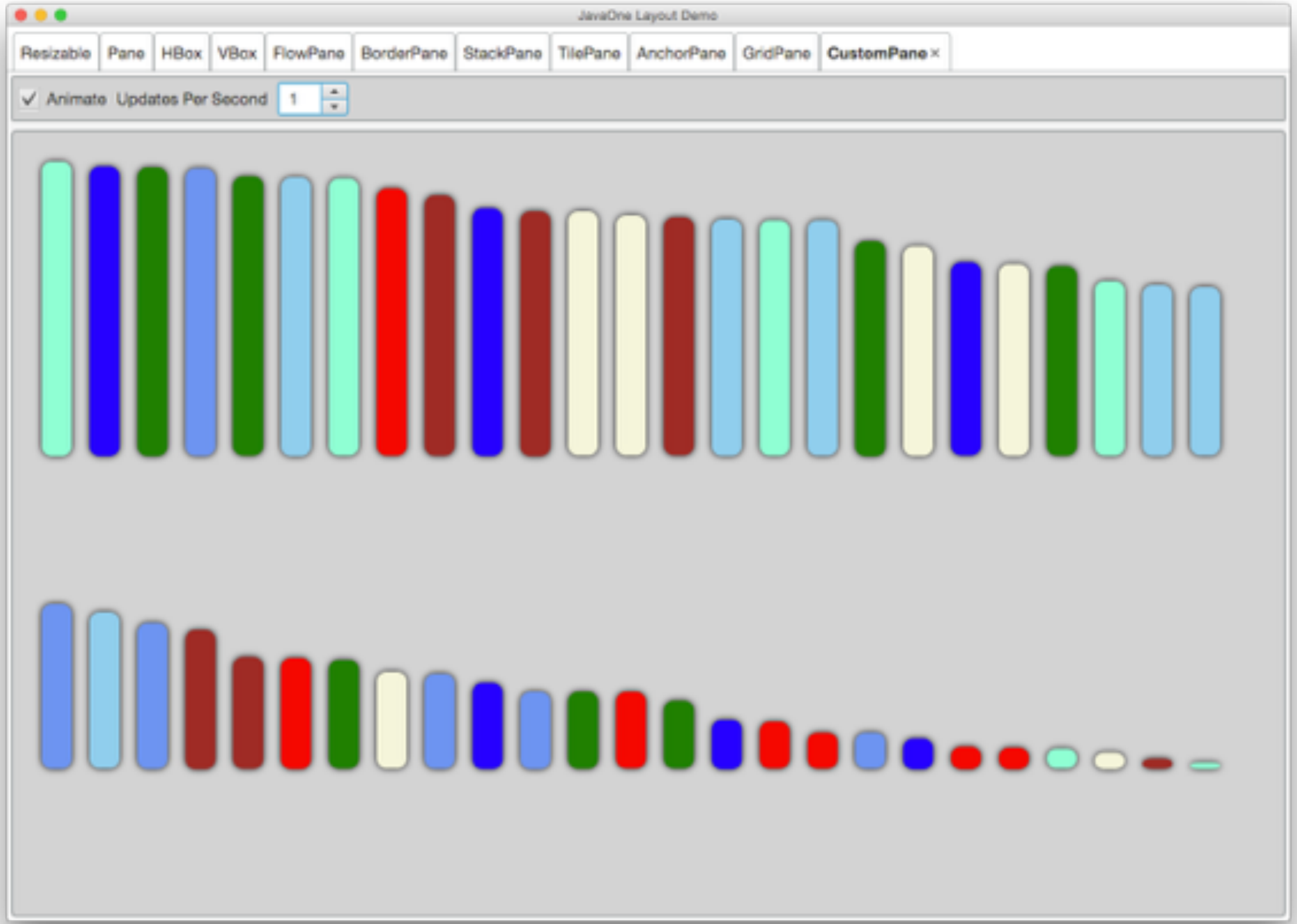
```
@Override protected double computeMinWidth(double height) { … }
@Override protected double computePrefWidth(double height) { … }
@Override protected double computeMaxWidth(double height) { … }
```

Custom Layout

- Simple example of layout of children by sorted height

```
@Override protected void layoutChildren() {
    List<Node> sortedManagedChidlren =
            new ArrayList<>(getManagedChildren());
    Collections.sort(sortedManagedChidlren, (c1, c2)
            -> new Double(c2.prefHeight(-1)).compareTo(
                    new Double(c1.prefHeight(-1))));
    double currentX = pad;
    for (Node c : sortedManagedChidlren) {
        double width = c.prefWidth(-1);
        double height = c.prefHeight(-1);
        layoutInArea(c, currentX, maxHeight - height, width,
                height, 0, HPos.CENTER, VPos.CENTER);
        currentX = currentX + width + pad;
    }
}
```

Demo


JavaOne Layout Demo
Resizable
Pane
HBox
VBox
FlowPane
BorderPane
StackPane
TilePane
AnchorPane
GridPane
CustomPane
Animate
Updates Per Second
1

Summary

- JavaFX has a complete set of layouts
  - Easy to use customizable API
  - Will meet the needs of most applications
- Create your own custom layout for specialized needs
  - Override existing layout to add new capabilities
  - Override Region or Pane for fully custom behavior

Q & A

Session Surveys

Help us help you!!

- Oracle would like to invite you to take a moment to give us your session feedback. Your feedback will help us to improve your conference.
- Please be sure to add your feedback for your attended sessions by using the Mobile Survey or in Schedule Builder.

ORACLE®

JavaOne™

ORACLE®