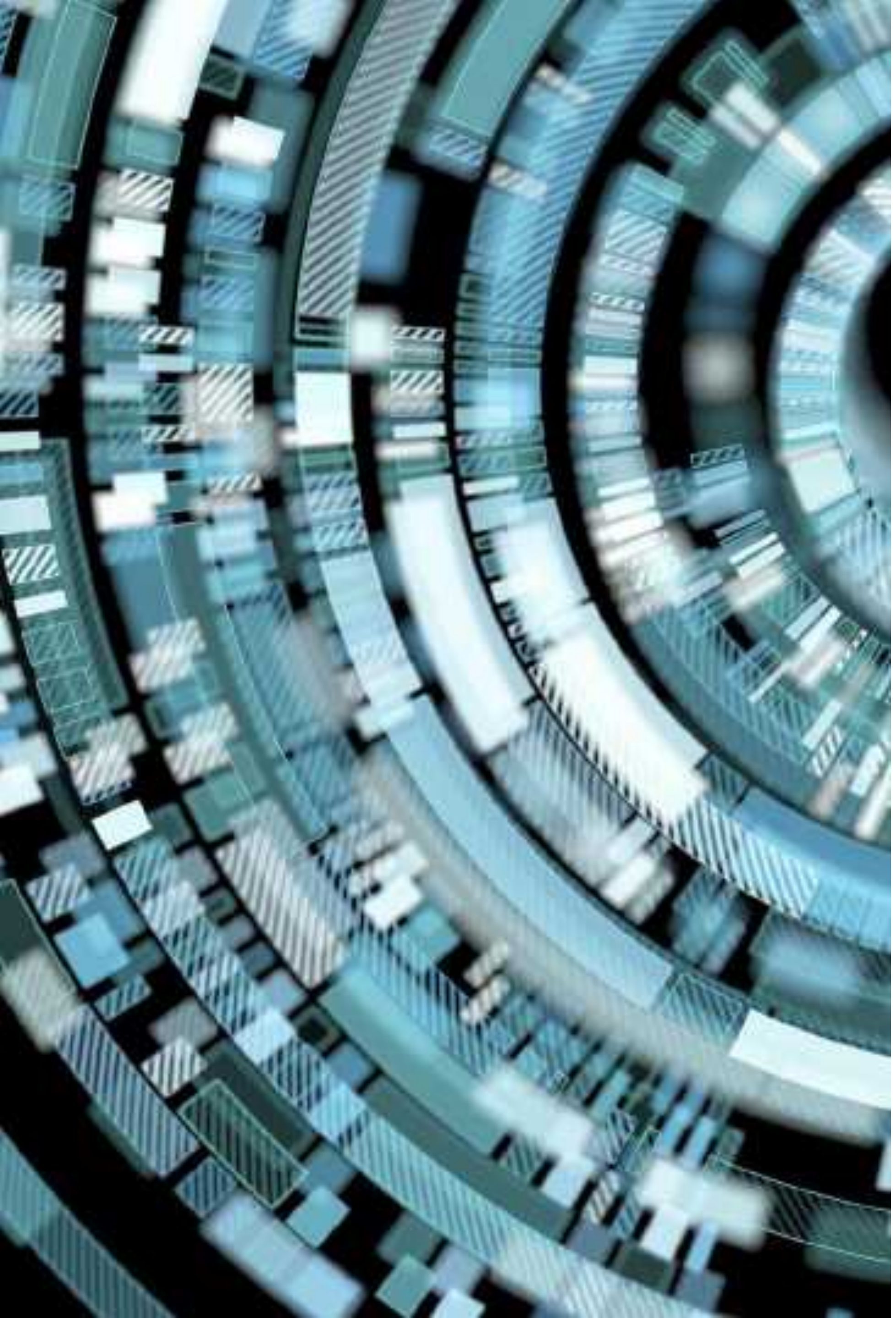
Computer System Design & Application
计算机系统设计与应用A

陶伊达 （TAO Yida）
taoyd@sustech.edu.cn

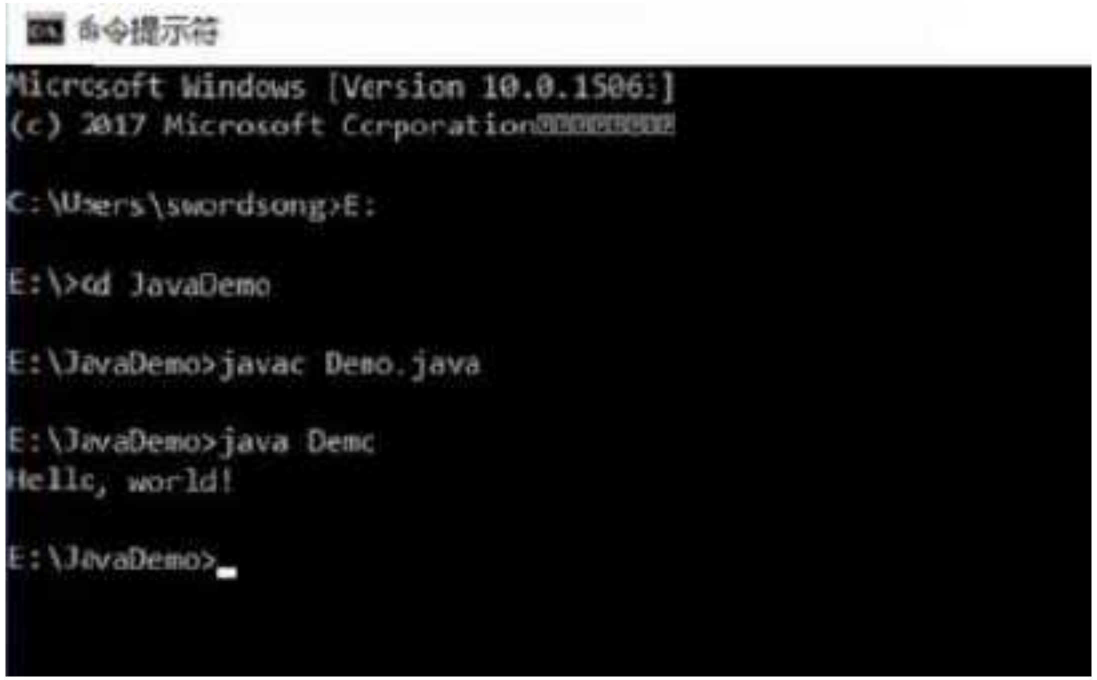


Lecture 5

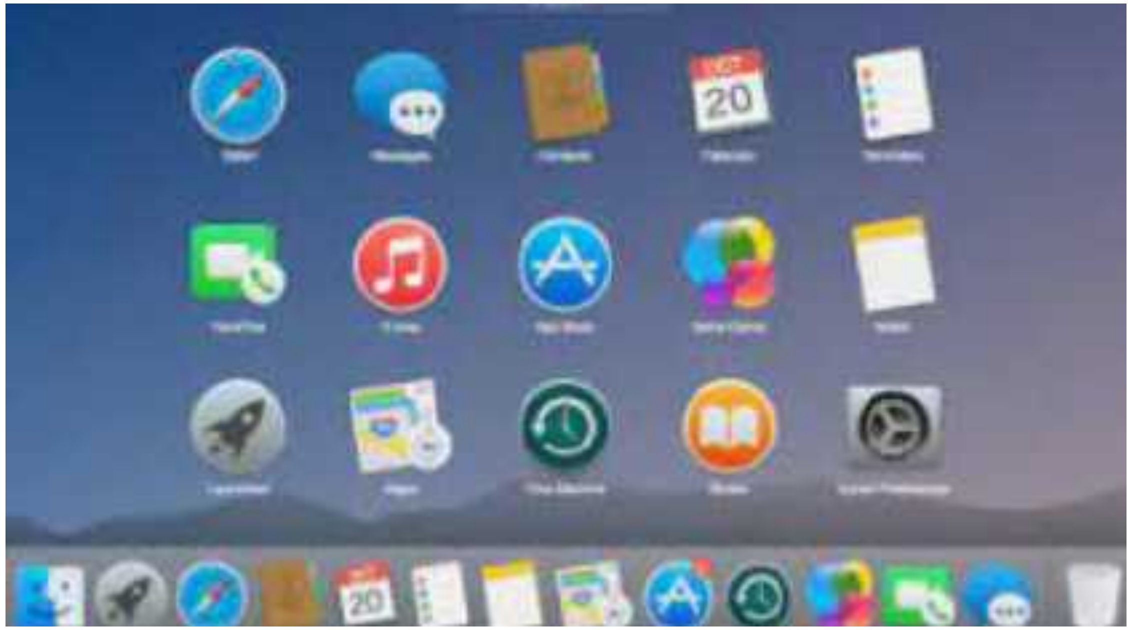
- Introduction to GUI
- JavaFX

GUI Overview

- **G**raphical **U**ser **I**nterface (GUI): a form of user interface that allows users to interact with electronic devices through graphical icons
- Easier to use compared to text-based user interface (e.g., CLI)



```
命令提示符
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation[illegible]

C:\Users\swordsong>E:

E:\>cd JavaDemo

E:\JavaDemo>javac Demo.java

E:\JavaDemo>java Demo
Hello, world!

E:\JavaDemo>_
```



Java GUI History

Abstract Window Toolkit (AWT)

- JDK 1.0
- Most of AWT's UI components have become obsolete

Swing

- JDK 1.2, enhancement of AWT
- Becomes legacy GUI library (only used in old projects)

JavaFX

- JDK 8, replacement to Swing
- Actively maintained and expected to grow in future

AWT

- **Components**: e.g., `Button, Label, and TextField`
- **Container**: used to hold components (e.g., `Frame, Panel`)

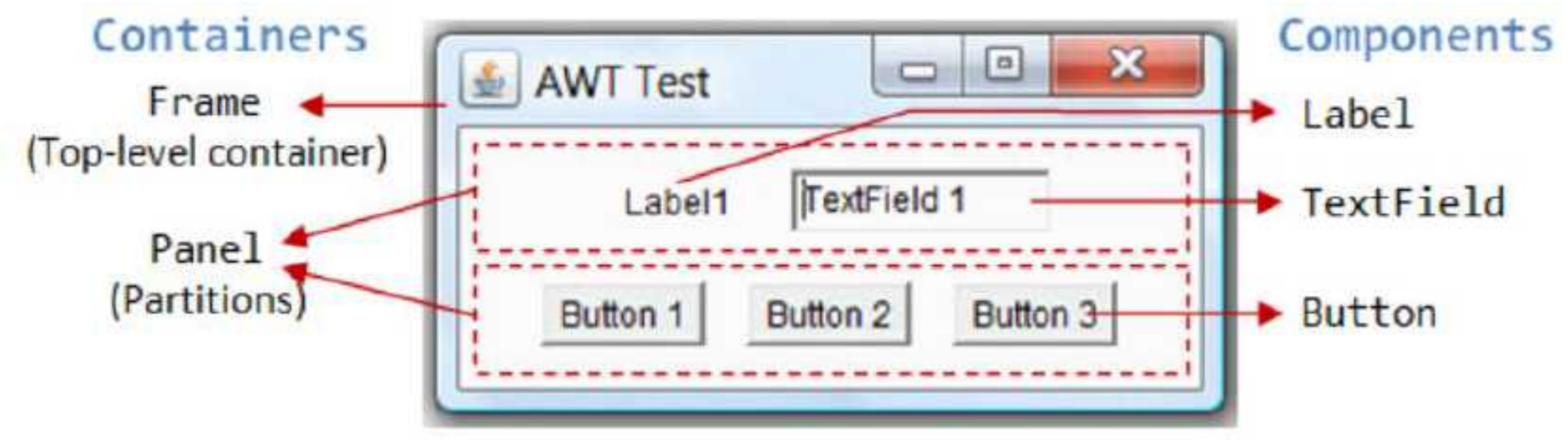


Image source: https://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html

AWT

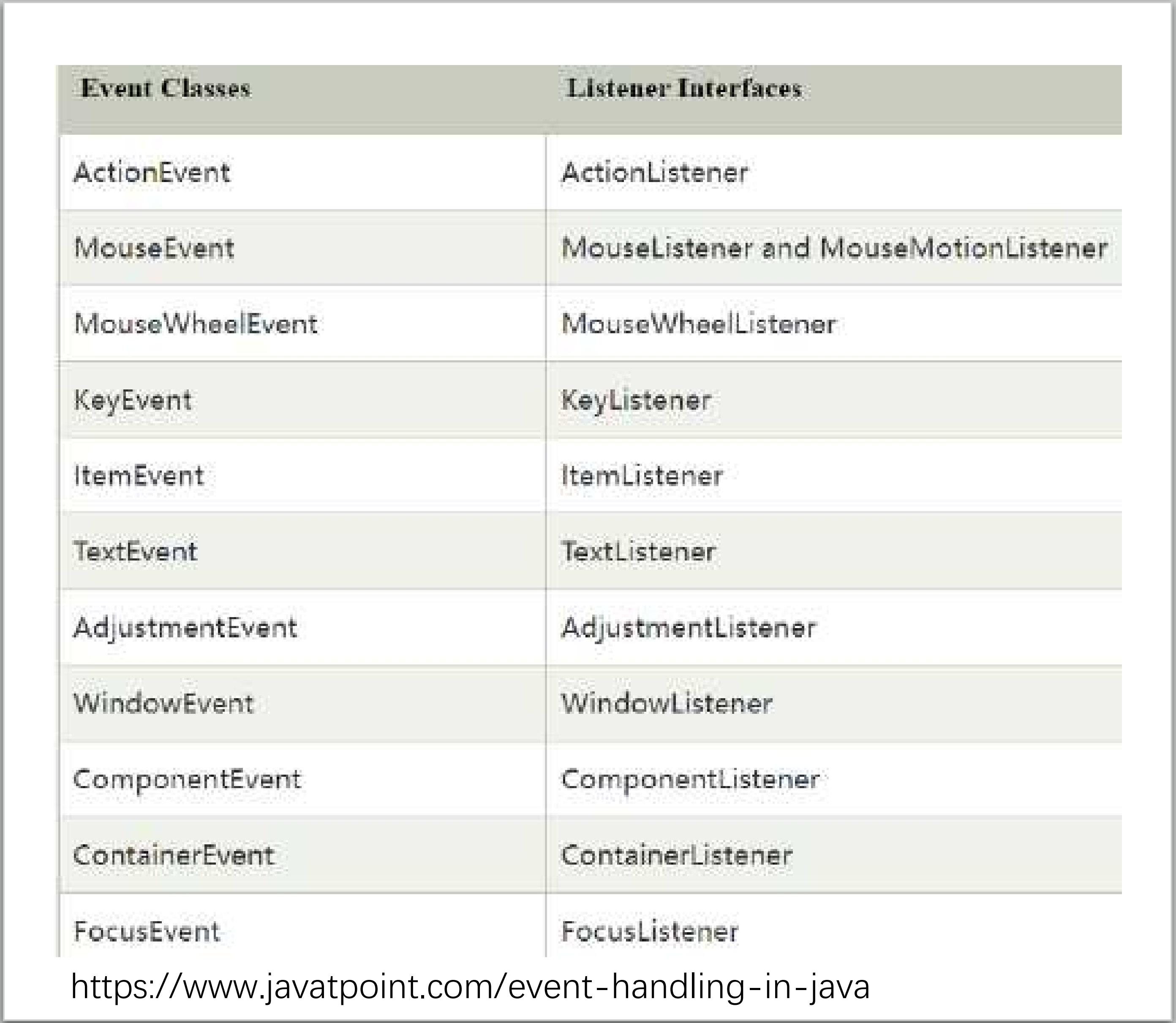
- A *component* must be added to a *container*

```
Panel pnl = new Panel();               // Panel is a container
Button btn = new Button("Press");      // Button is a component
pnl.add(btn);                          // Add button to the panel
```

Done? What else should be implemented?

Event Listener

- **Event**: mouse clicked, mouse moved, key press, etc.
- **Event listener**: listens for an event and responds accordingly



Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

https://www.javatpoint.com/event-handling-in-java

AWT Button Click Event

- An event listener must be "registered" in an event object (e.g., button)
- The listeners must implement the `java.awt.ActionListener` interface (`actionPerformed()` method)

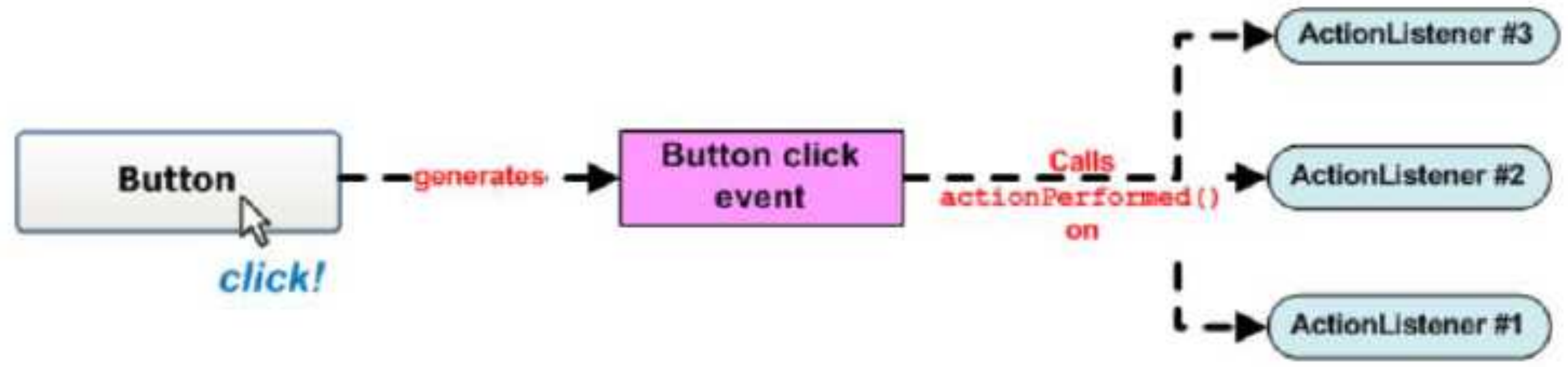
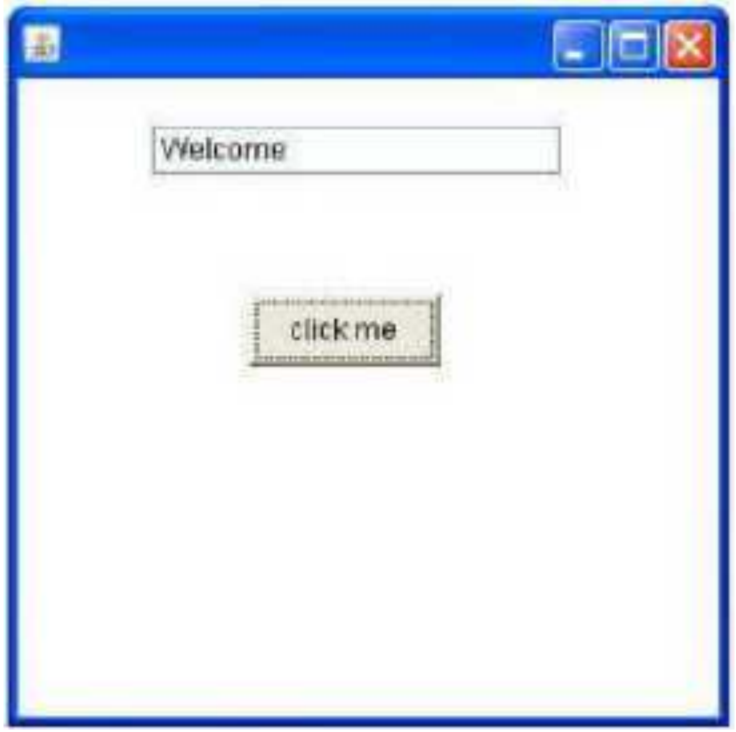


Image source: https://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html

AWT Button Click Event

```
TextField tf = new TextField();
Button btn=new Button("click me");

btn.addActionListener(new ActionListener(){
        public void actionPerformed(){
                tf.setText(“Welcome");
        }
});

panel.add(tf);
panel.add(btn);
```

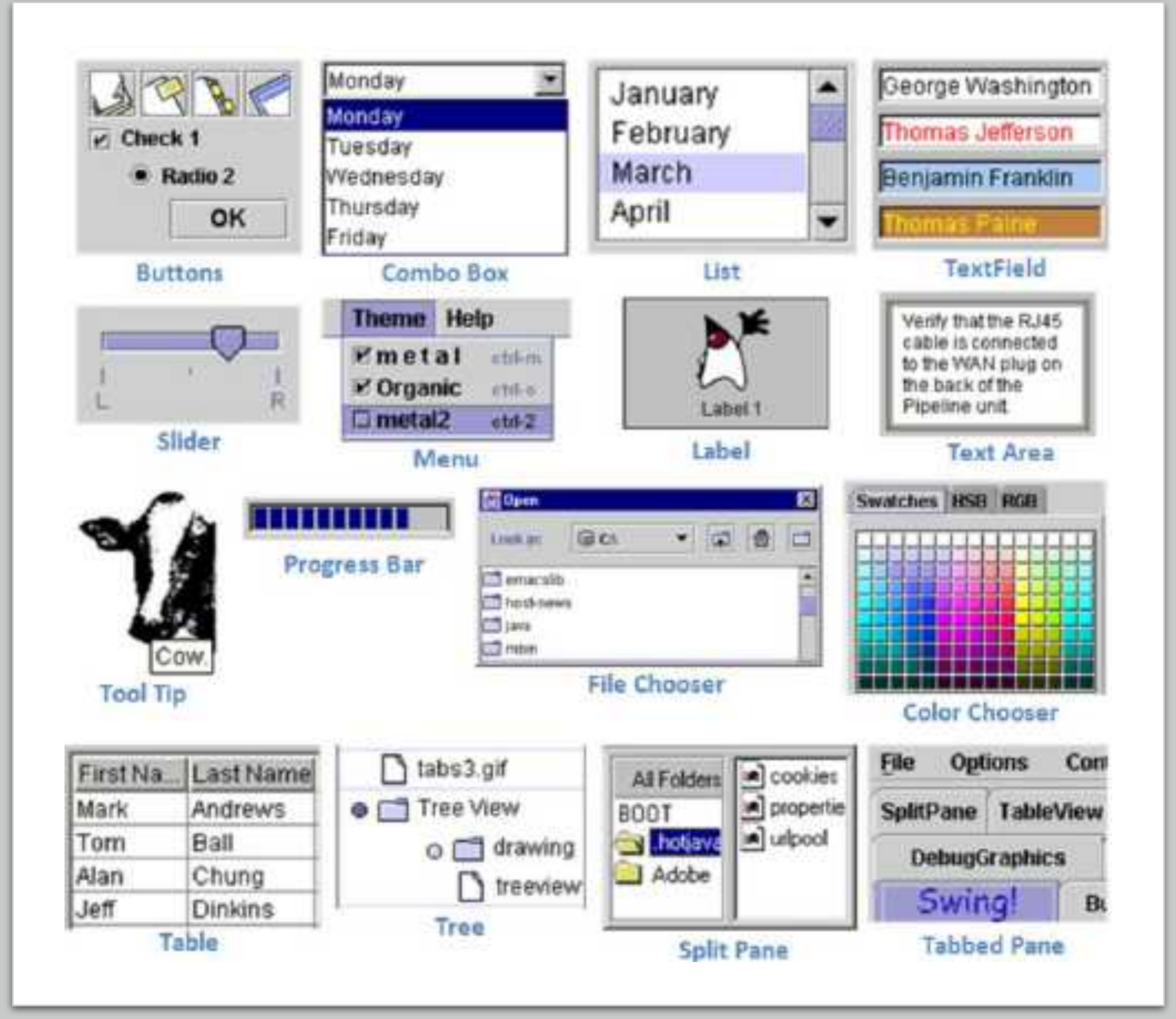


Swing

Swing extends AWT by adding richer graphics functionalities and interactivity to Java applications

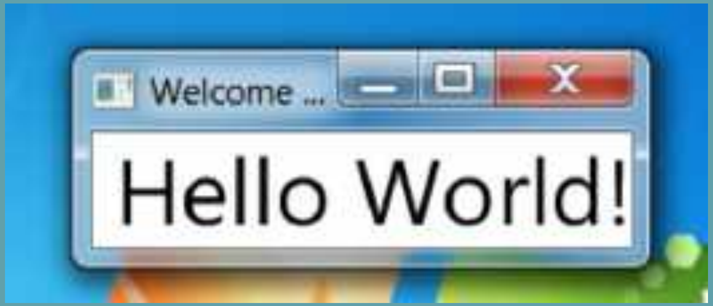
(more comprehensive components)

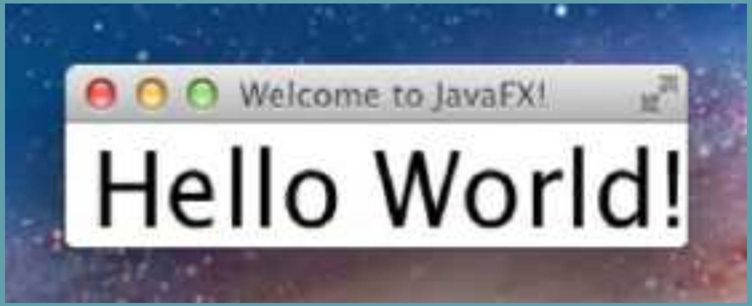
Image source: https://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html

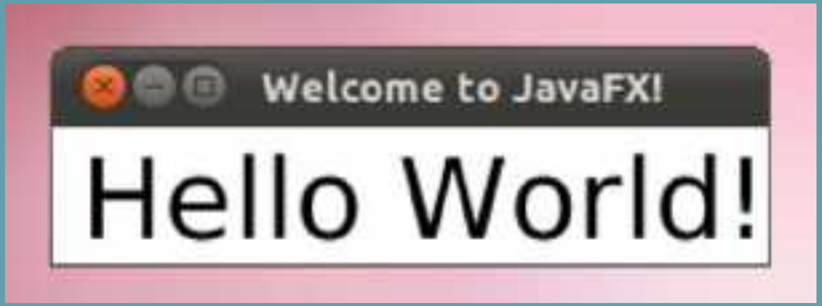


Swing look-and-feel

You can create GUIs that can either look the same across platforms or can assume the look and feel of the current OS platform (such as Microsoft Windows, Linux).







Swing Class Hierarchy

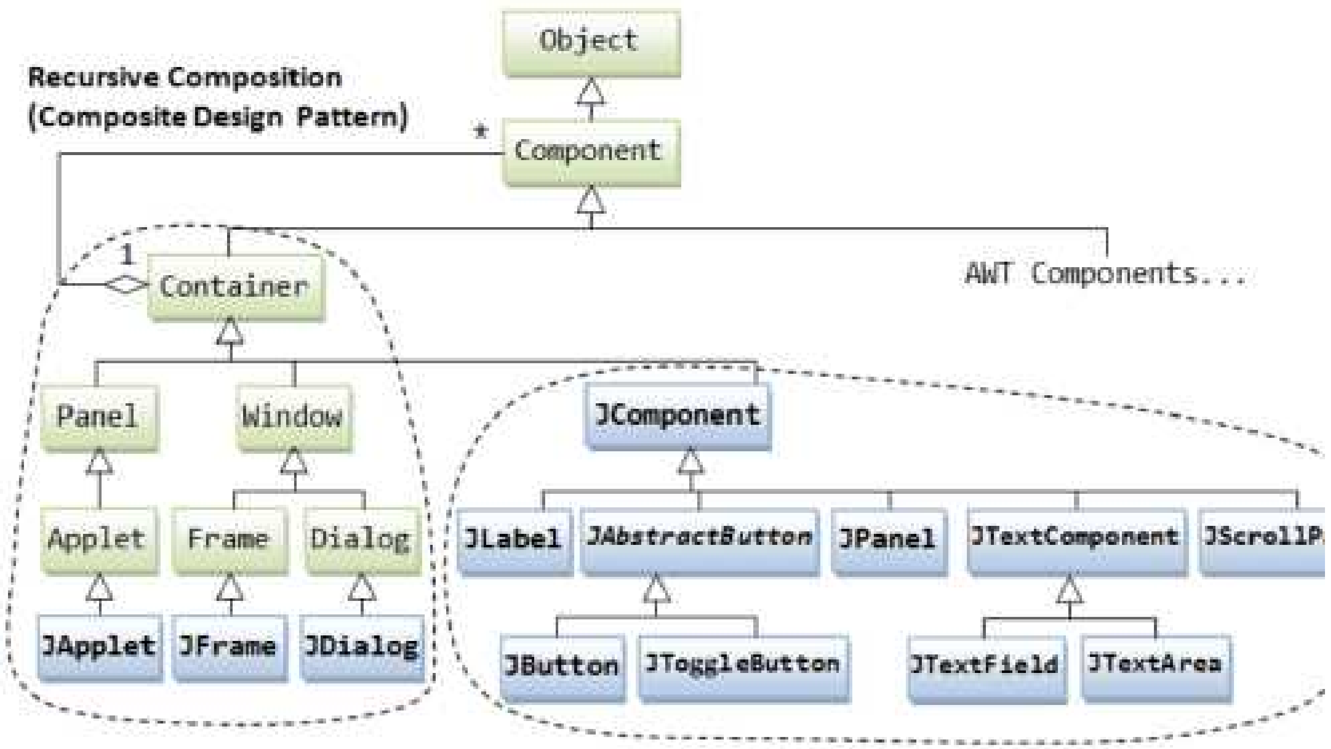
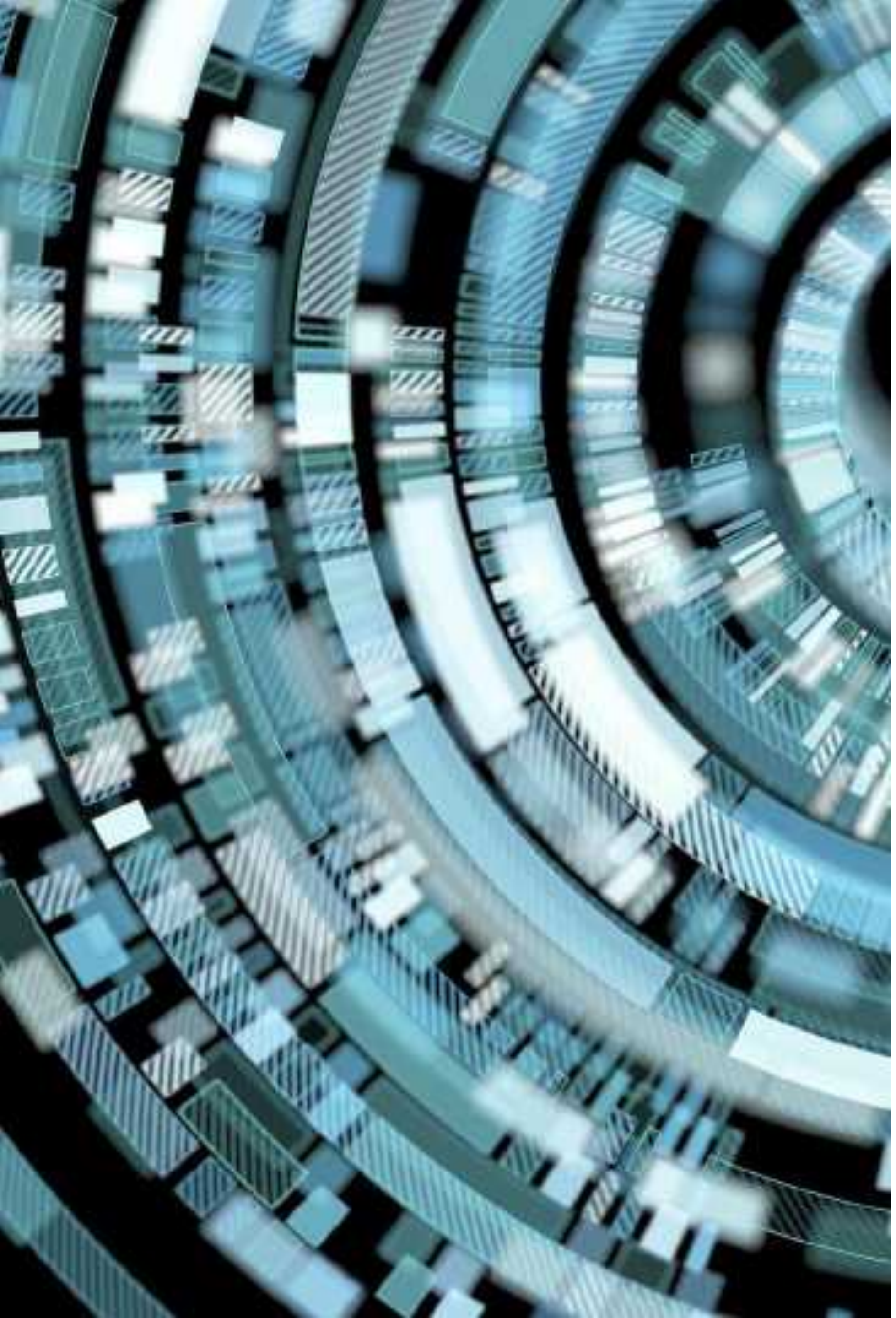


Image source: https://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html

- Swing also has *containers* and *components*
- Swing component classes (`javax.swing`) begin with a prefix `"J"`

Swing Workflow

1. Make a window (JFrame)
2. Make a container (JPanel)
   - Add it to the window
3. Add components to the container
   - Buttons, textbox, etc
   - Setup layout to control positions
   - Setup listeners to react to events
4. Let the window display the container
5. Wait for the events……



Lecture 5

- Introduction to GUI
- JavaFX
  - Overview
  - Hello World
  - Design & Concepts
  - Layouts, Shapes, UI controls
  - Charts and Axis
  - Transformation, Animation, Effects

JavaFX Overview

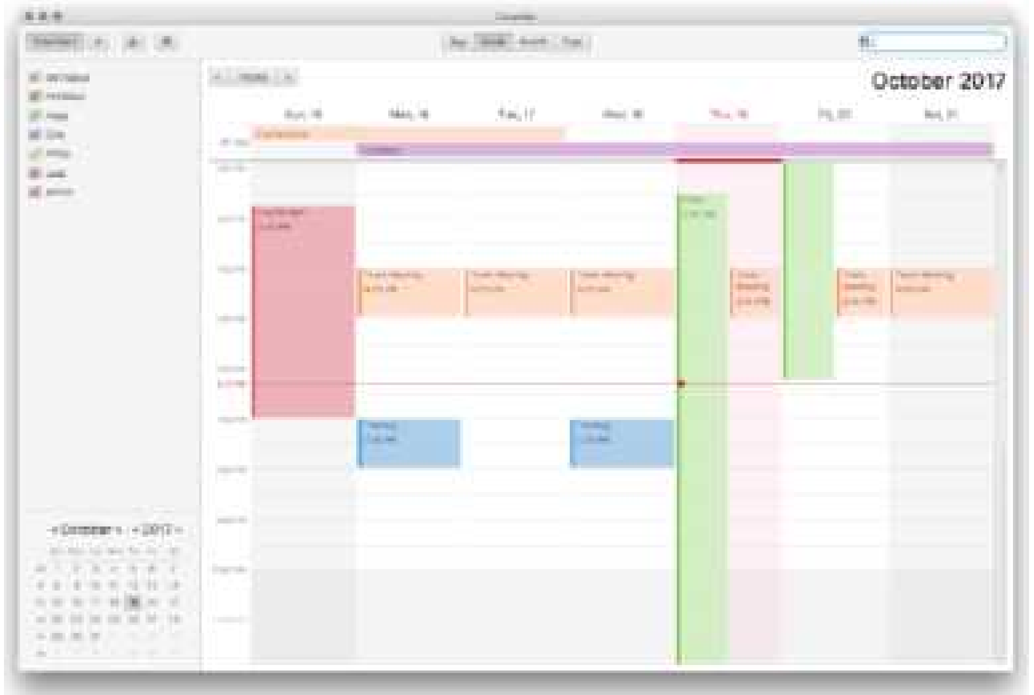
- Official doc: JavaFX is an open source, next generation client application platform for desktop, mobile and embedded systems built on Java (i.e., a GUI toolkit for Java)
- JavaFX can run on various OS and devices
  - Windows
  - Linux
  - Mac
  - iOS
  - Android/Chromebook
  - Raspberry Pi

JavaFX Showcases

Images from JavaFX official site

AsciidocFX

An Asciidoc editor to build PDF, Epub, Mobi and HTML books, documents and slides



CalendarFX

A Java framework for creating sophisticated calendar views

Gluon Maps

Tiles based geo-location map framework

JavaFX Showcases

Images from JavaFX official site

TilesFX

A JavaFX library containing tiles for Dashboards

FXGL

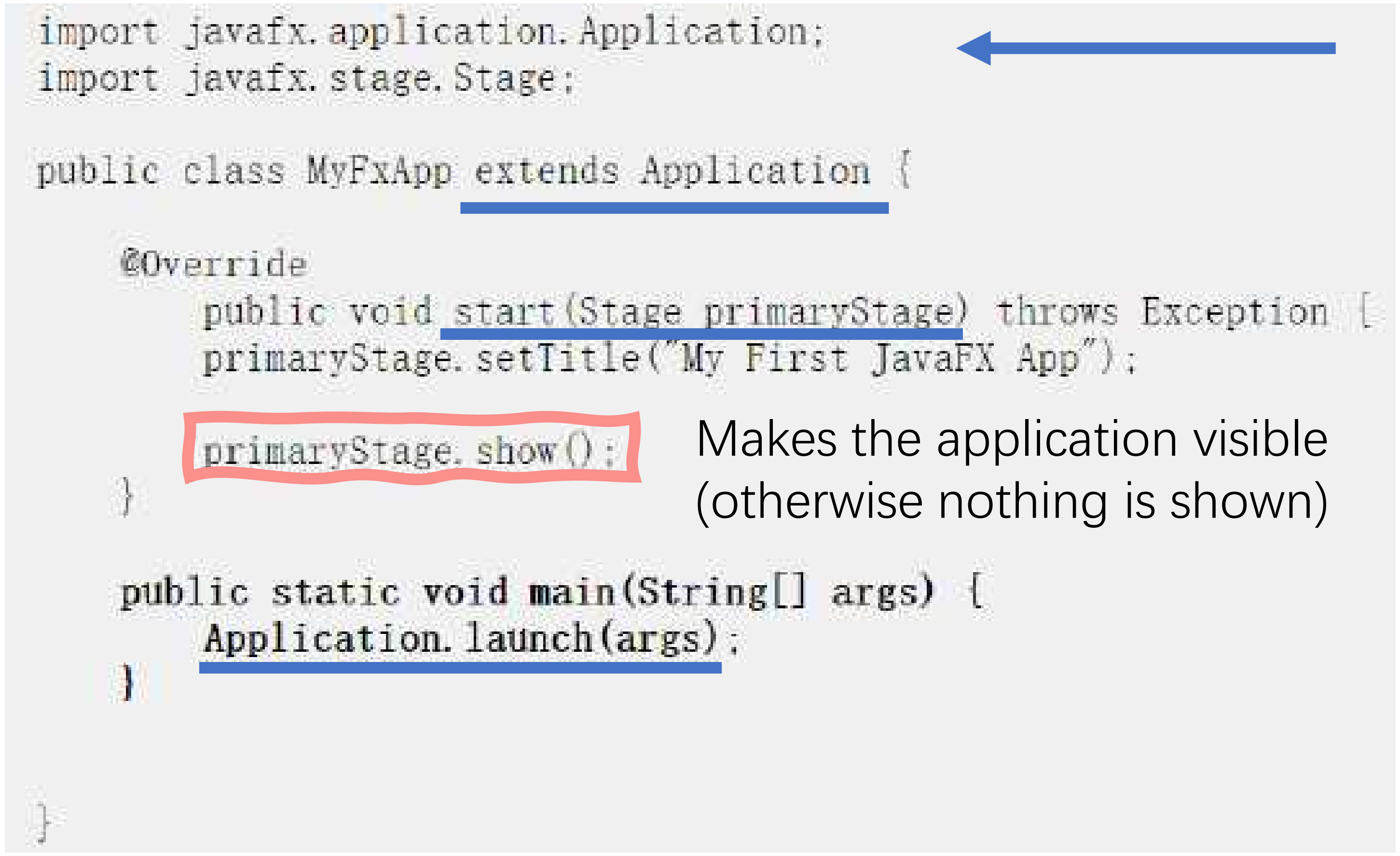
JavaFX game engine

JavaFX Showcases

北航1921 C50组大作业 基于JavaFX的植物大战僵尸

8713播放 · 总弹幕数9 2021-06-12 02:10:59

JavaFX Hello World



```
import javafx.application.Application;
import javafx.stage.Stage;

public class MyFxApp extends Application {

    @Override
        public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("My First JavaFX App");

        primaryStage.show();
    }

    public static void main(String[] args) {
        Application.launch(args);
    }


}
```

Makes the application visible (otherwise nothing is shown)

Import necessary classes from `javafx`

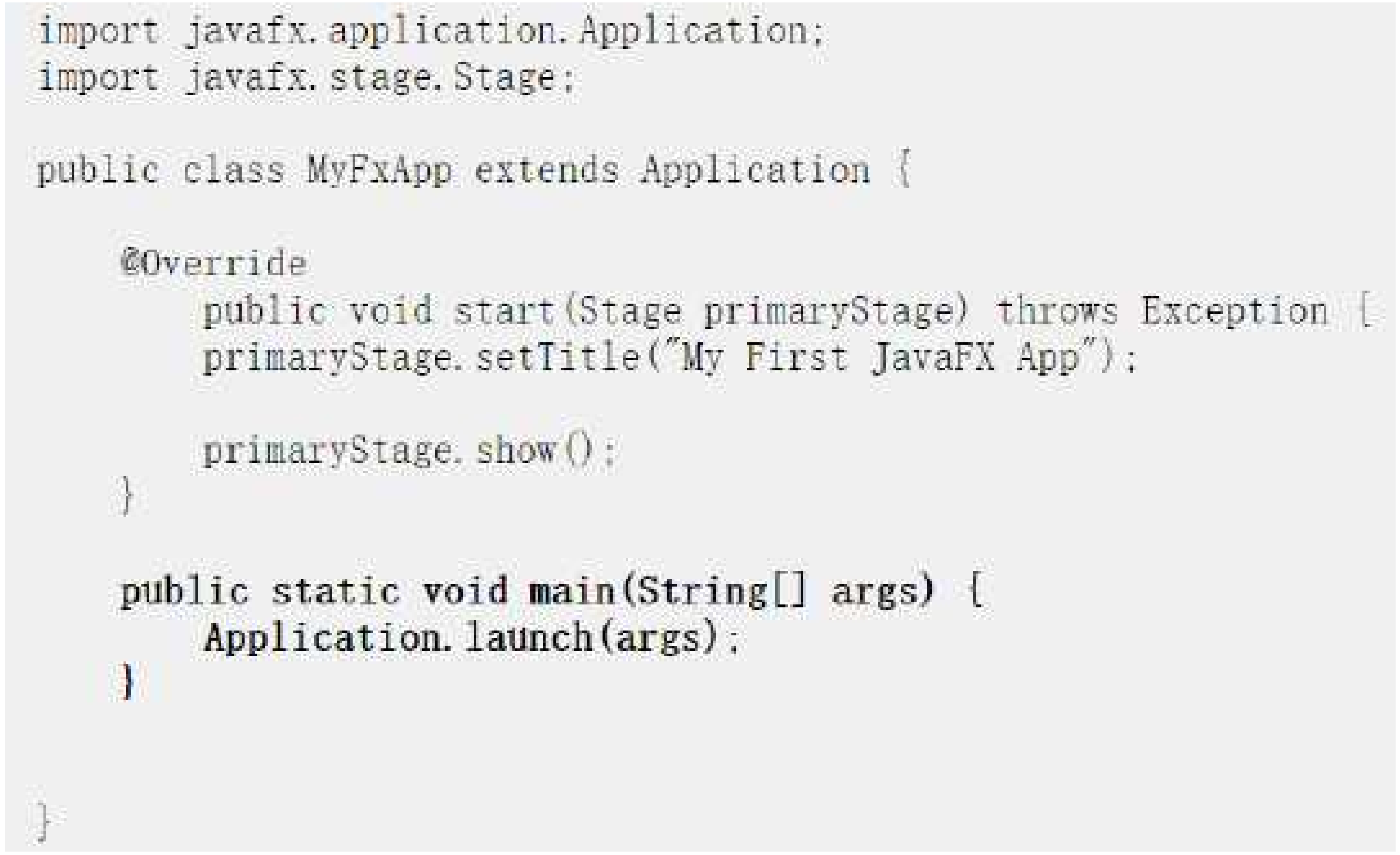
Extend the abstract `Application` class

Implement the abstract `start()` method of the `Application` class (called when a JavaFX application starts)

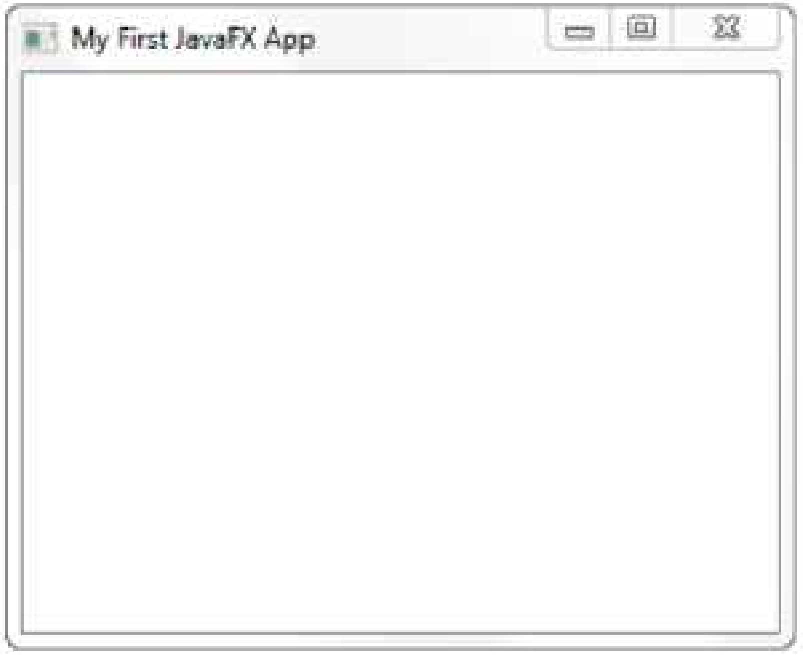
`launch()` launches the JavaFX runtime and your JavaFX application.

http://tutorials.jenkov.com/javafx/your-first-javafx-application.html

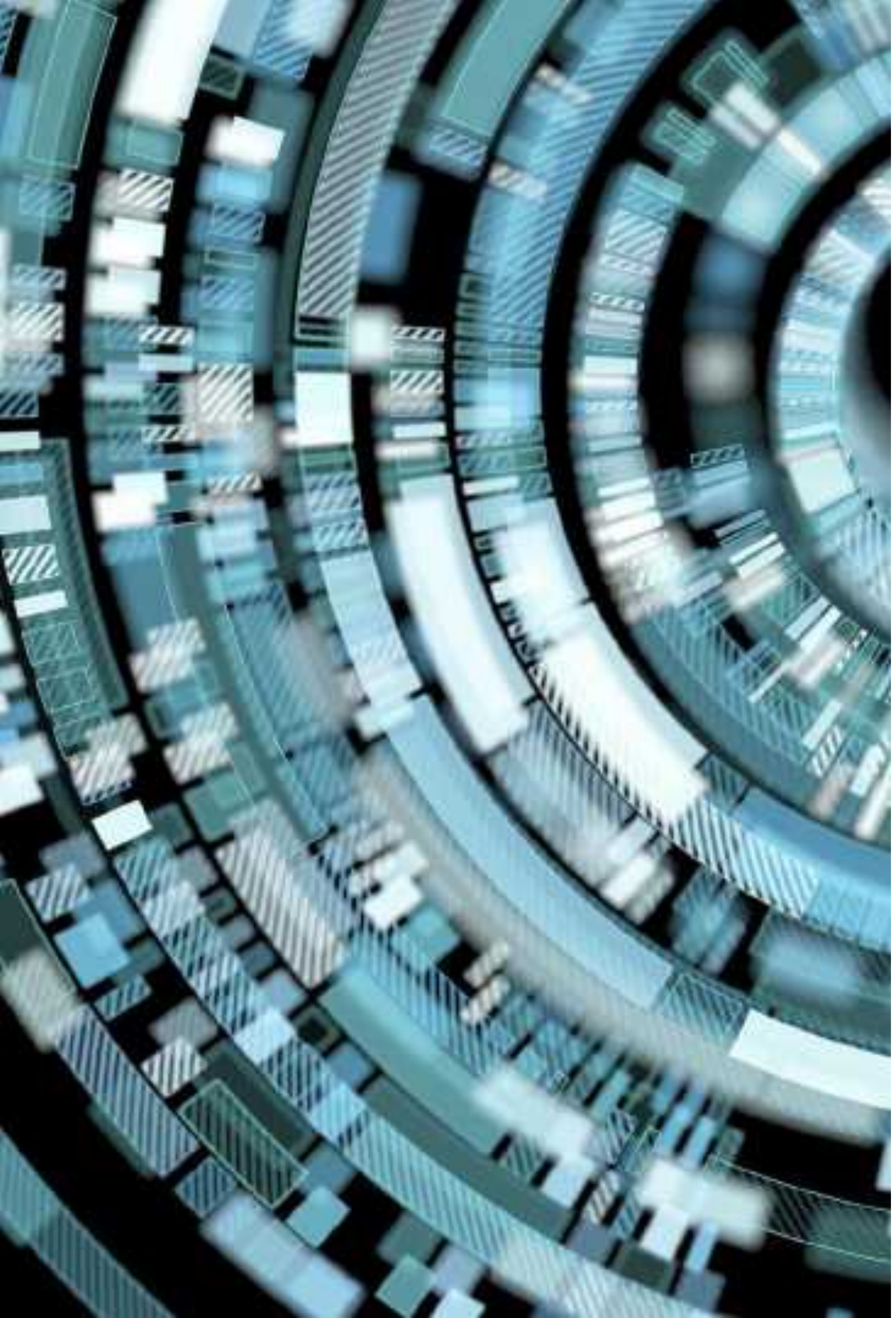
JavaFX Hello World



```
import javafx.application.Application;
import javafx.stage.Stage;

public class MyFxApp extends Application {

    @Override
        public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("My First JavaFX App");

        primaryStage.show();
    }

    public static void main(String[] args) {
        Application.launch(args);
    }


}
```



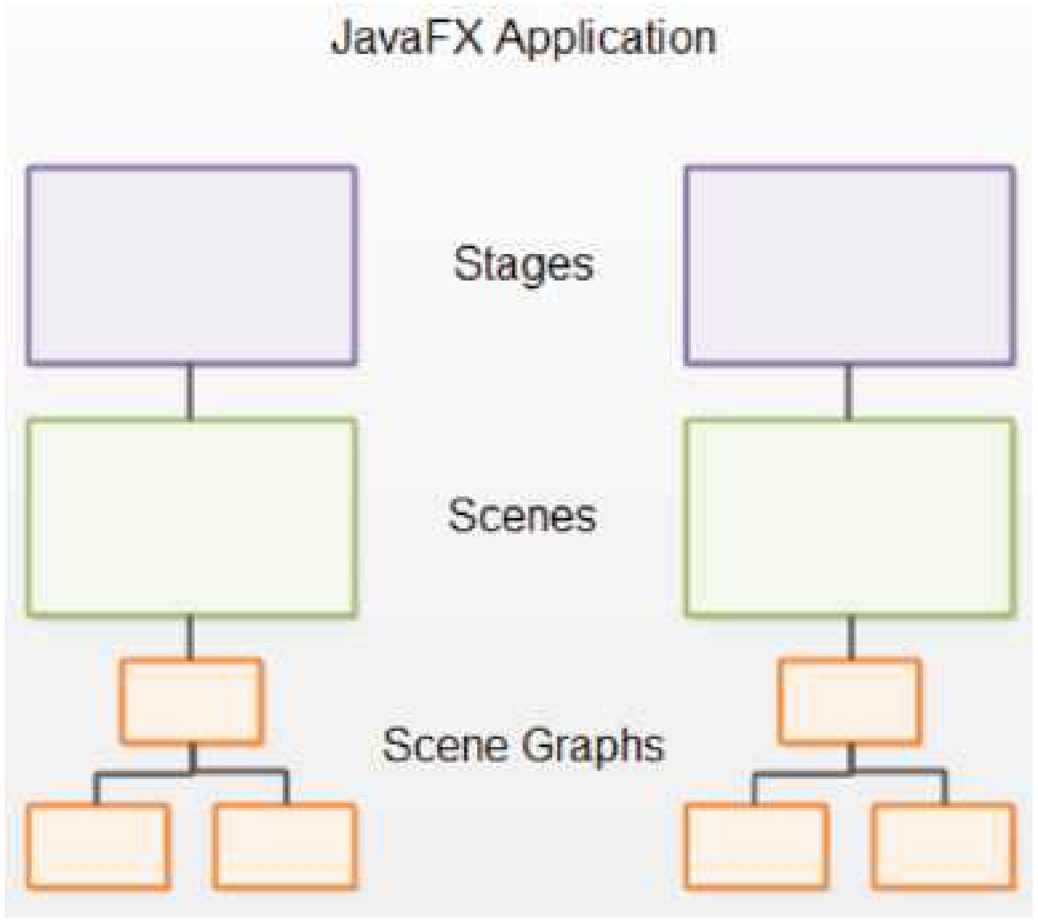
http://tutorials.jenkov.com/javafx/your-first-javafx-application.html



Lecture 5

- Introduction to GUI
- JavaFX
  - Overview
  - Hello World
  - Design & Concepts
  - Layouts, Shapes, UI controls
  - Charts and Axis
  - Transformation, Animation, Effects

JavaFX Design



http://tutorials.jenkov.com/javafx/your-first-javafx-application.html

Stage (窗体)

- The outer frame for a JavaFX application, typically corresponds to a window.
- A JavaFX application can have one or more stages (multiple windows open)

Scene (场景)

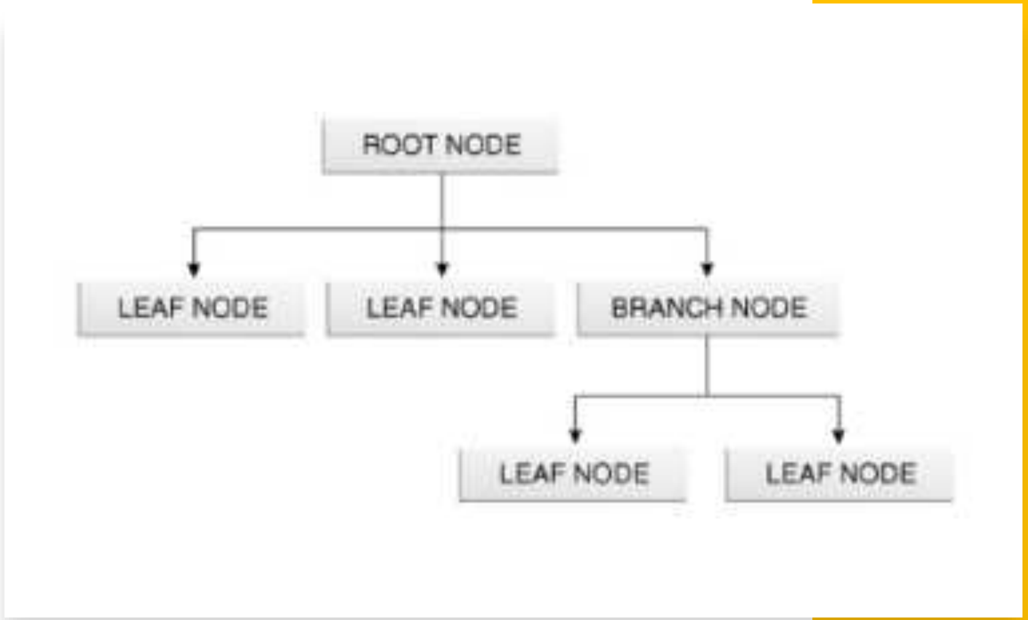
- Containing all GUI componenets visible in a window (i.e., to display things on the stage)
- A stage can only show one scene at a time, but it is possible to exchange the scene at runtime

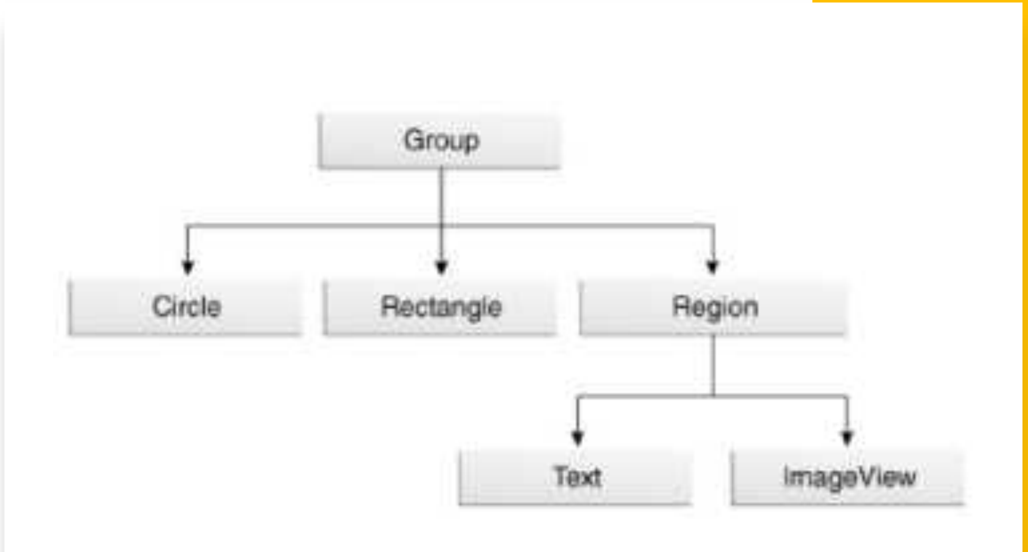
Scene Graphs (场景图)

- All visual components (controls, layouts etc.) attached to a scene is called the scene graph

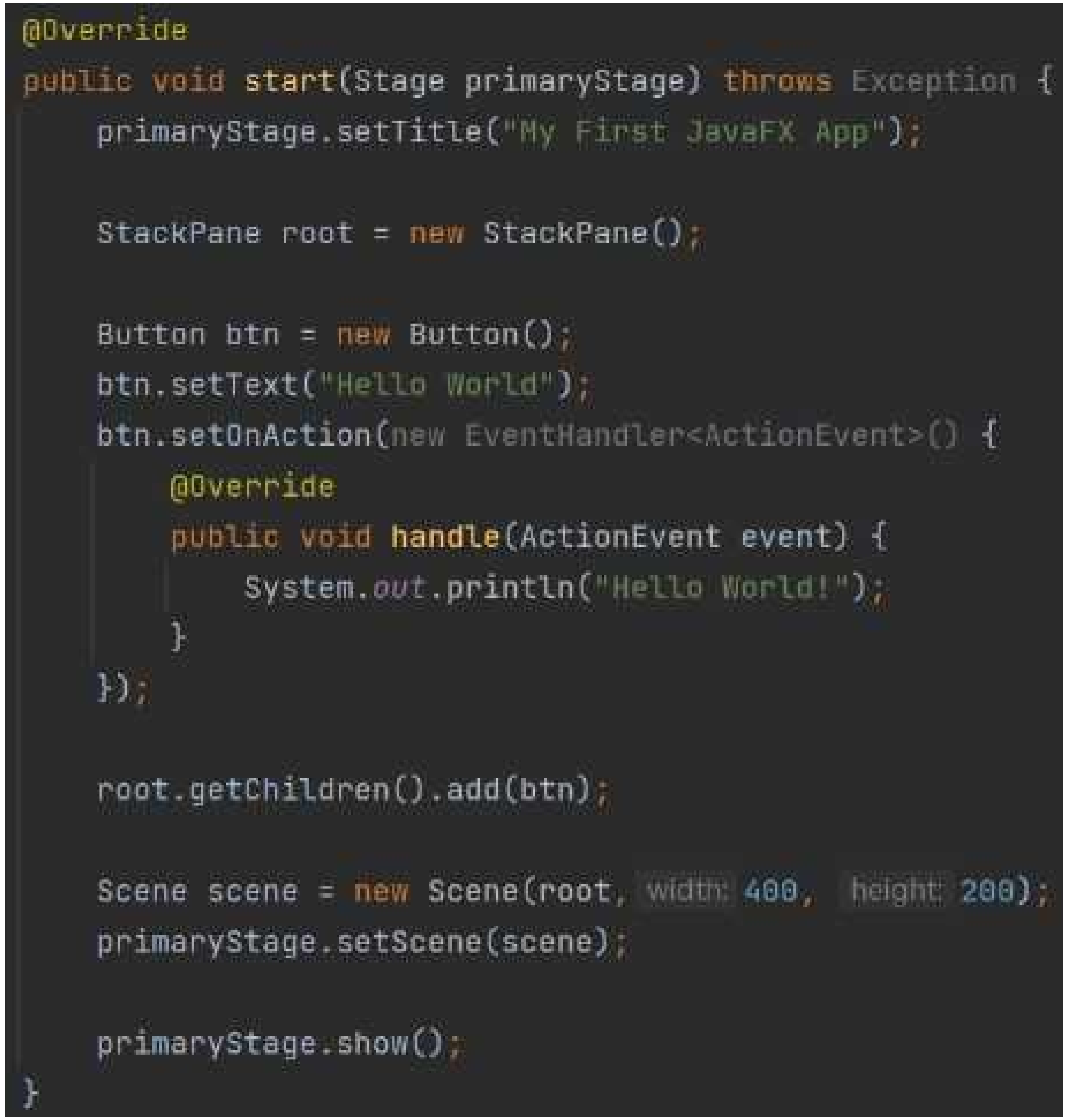
Scene Graph

- A tree data structure of nodes
- A node is a visual object of a JavaFX application
- Each node is classified as either a branch node (it can have children), or a leaf node (it cannot have children)
- A JavaFX application must specify the root node for the scene graph by setting the root property.





https://docs.oracle.com/javafx/2/scenegraph/jfxpub-scenegraph.htm



```
@Override
public void start(Stage primaryStage) throws Exception {
    primaryStage.setTitle("My First JavaFX App");

    StackPane root = new StackPane();

    Button btn = new Button();
    btn.setText("Hello World");
    btn.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            System.out.println("Hello World!");
        }
    });

    root.getChildren().add(btn);

    Scene scene = new Scene(root, width: 400, height: 200);
    primaryStage.setScene(scene);

    primaryStage.show();
}
```

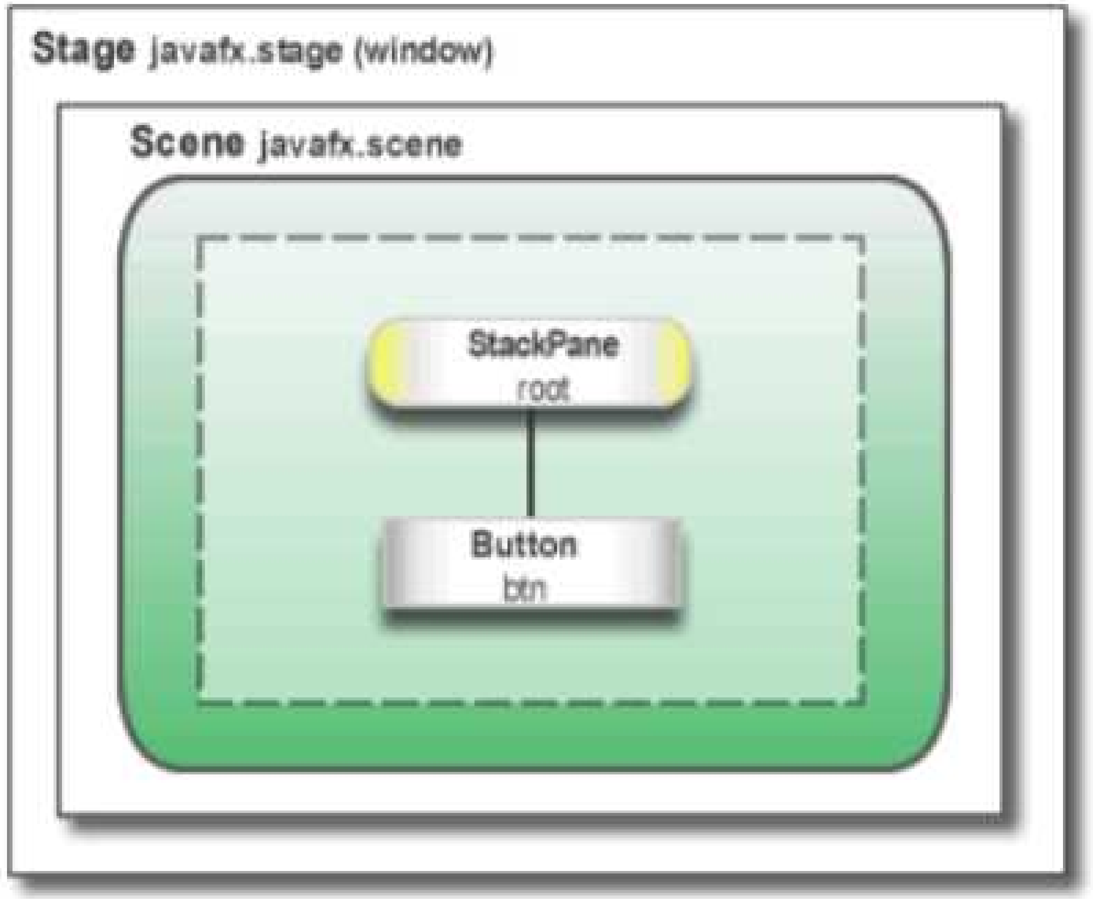
JavaFX Hello World

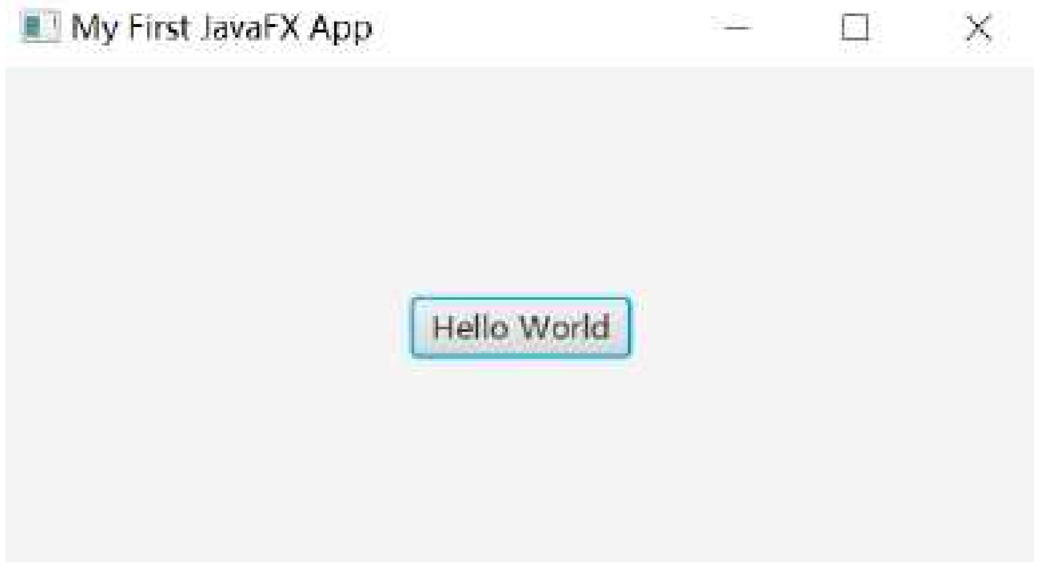
1. The root node is a StackPane object, a resizable layout node

2. The child node is a Button object, with an event handler for printing a message when pressed

3. Add button to the root node
4. Create a scene with the root
5. Set the scene for the stage and show

JavaFX Hello World





JavaFX Design

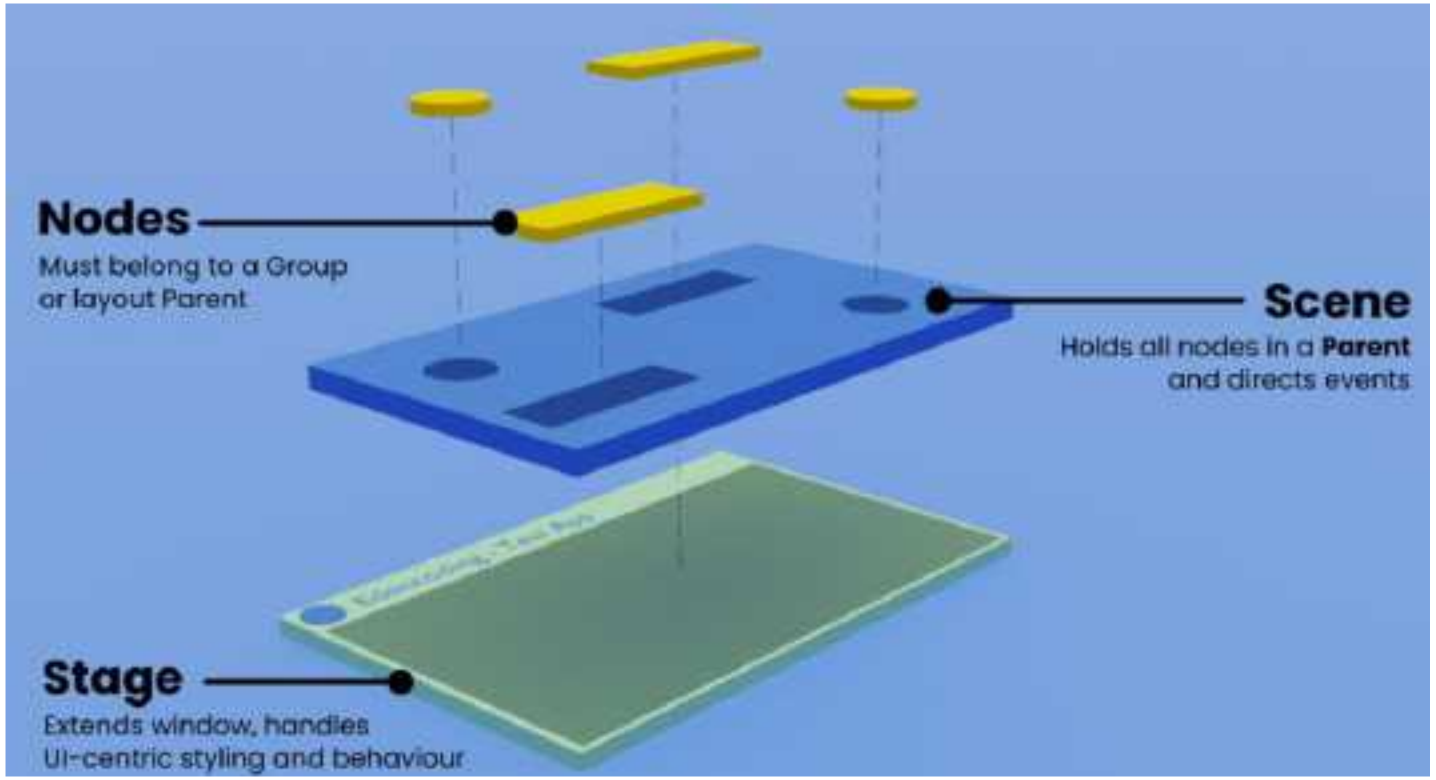
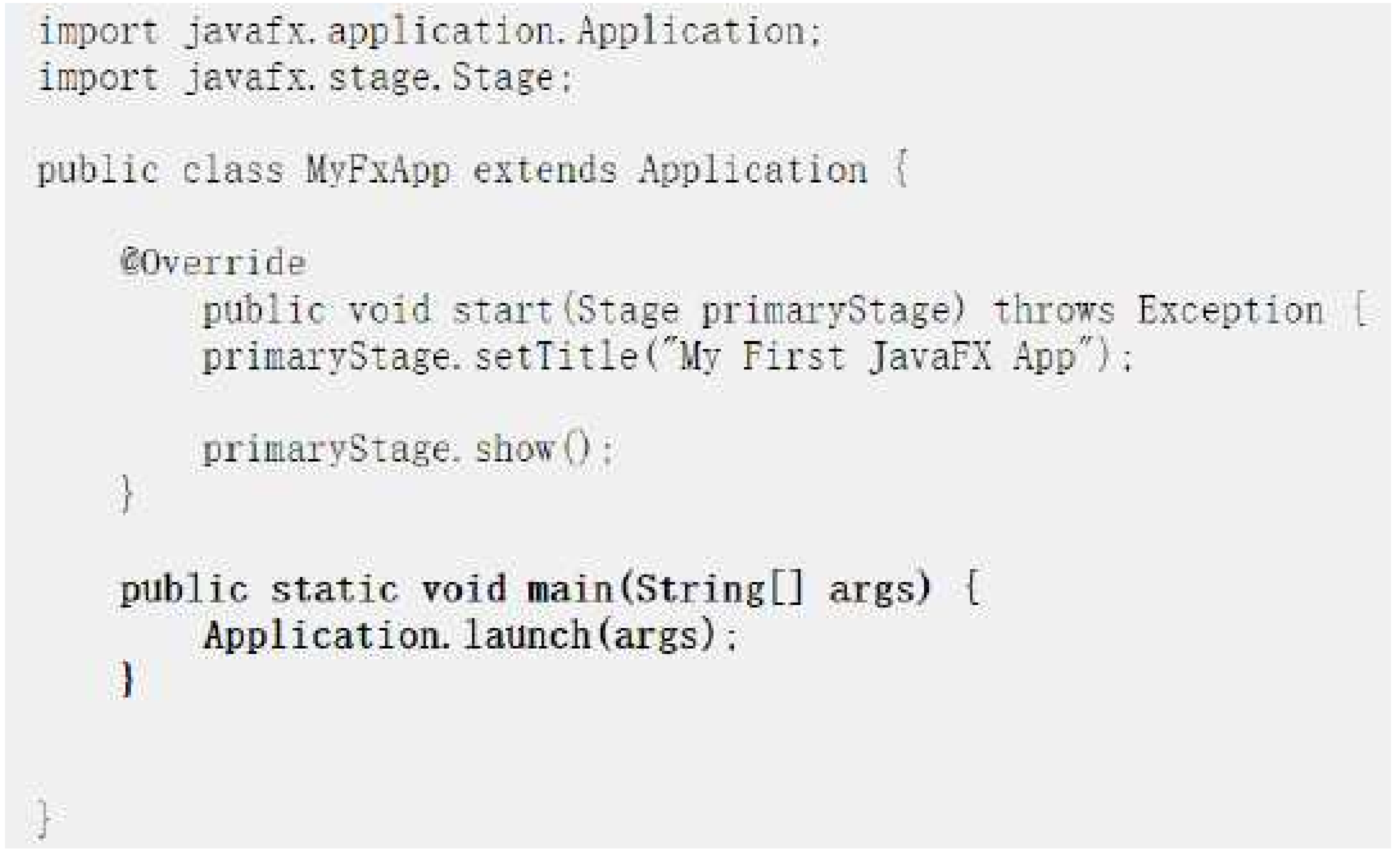

Nodes
Must belong to a Group or layout Parent
Scene
Holds all nodes in a **Parent** and directs events
Stage
Extends window, handles UI-centric styling and behaviour

Image source: https://edencoding.com/javafx-scene/

JavaFX Stage

- A Stage represents a window in a JavaFX application
- A Stage object is created and passed to the `start(Stage primaryStage)` method when a JavaFX application starts up
- New Stage objects could be created later if the application needs to open more windows

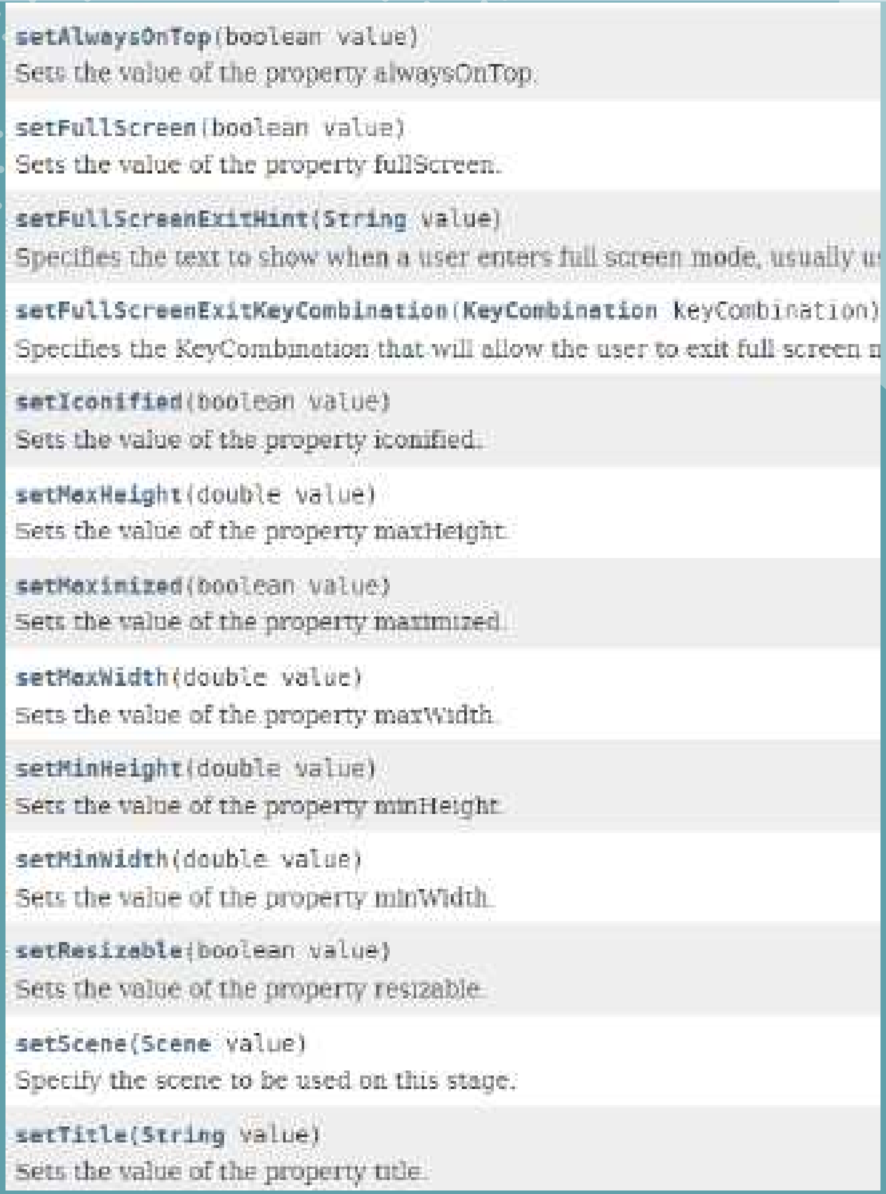


```
import javafx.application.Application;
import javafx.stage.Stage;

public class MyFxApp extends Application {

    @Override
        public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("My First JavaFX App");

        primaryStage.show();
    }

    public static void main(String[] args) {
        Application.launch(args);
    }


}
```

JavaFX Stage Properties

Please refer to the official documentation for full details

https://docs.oracle.com/javase/8/javafx/api/javafx/stage/Stage.html



`setAlwaysOnTop(boolean value)`
Sets the value of the property alwaysOnTop.

`setFullScreen(boolean value)`
Sets the value of the property fullScreen.

`setFullScreenExitHint(String value)`
Specifies the text to show when a user enters full screen mode, usually us

`setFullScreenExitKeyCombination(KeyCombination keyCombination)`
Specifies the KeyCombination that will allow the user to exit full screen m

`setIconified(boolean value)`
Sets the value of the property iconified.

`setMaxHeight(double value)`
Sets the value of the property maxHeight.

`setMaximized(boolean value)`
Sets the value of the property maximized.

`setMaxWidth(double value)`
Sets the value of the property maxWidth.

`setMinHeight(double value)`
Sets the value of the property minHeight.

`setMinWidth(double value)`
Sets the value of the property minWidth.

`setResizable(boolean value)`
Sets the value of the property resizable.

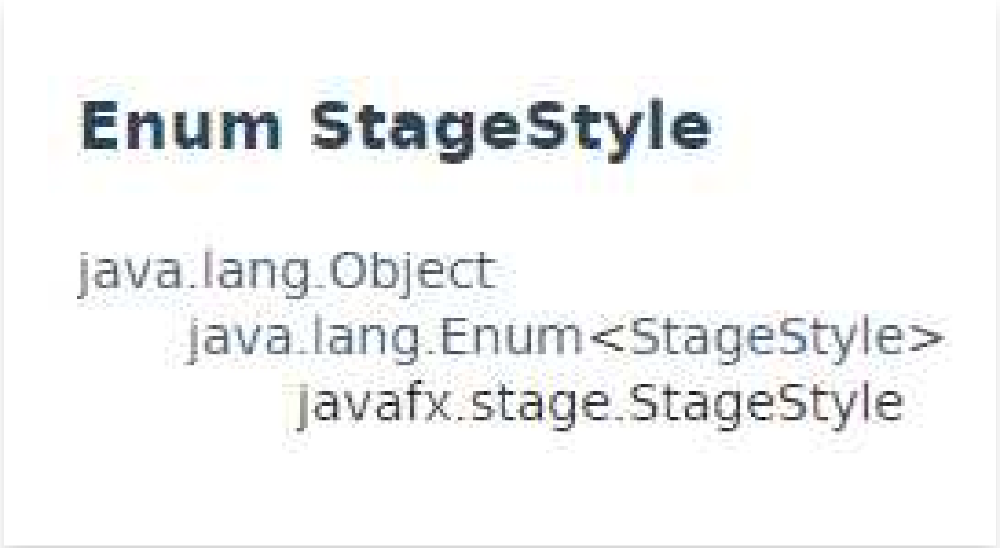
`setScene(Scene value)`
Specify the scene to be used on this stage.

`setTitle(String value)`
Sets the value of the property title.

TAO Yida@SUSTECH

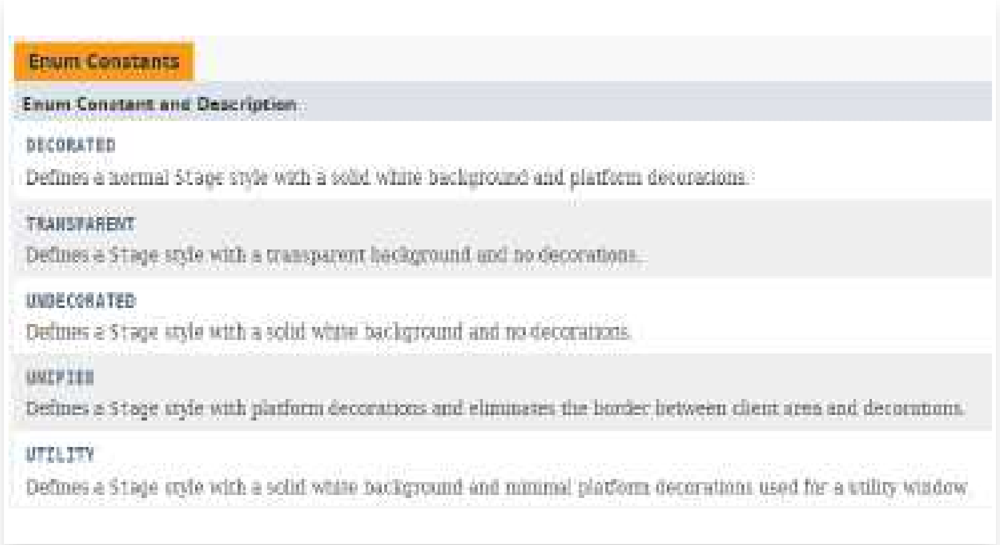
JavaFX Stage Style

```
stage.initStyle(StageStyle.DECORATED);

//stage.initStyle(StageStyle.UNDECORATED);
//stage.initStyle(StageStyle.TRANSPARENT);
//stage.initStyle(StageStyle.UNIFIED);
//stage.initStyle(StageStyle.UTILITY);
```



Enum StageStyle

```
java.lang.Object
    java.lang.Enum<StageStyle>
        javafx.stage.StageStyle
```

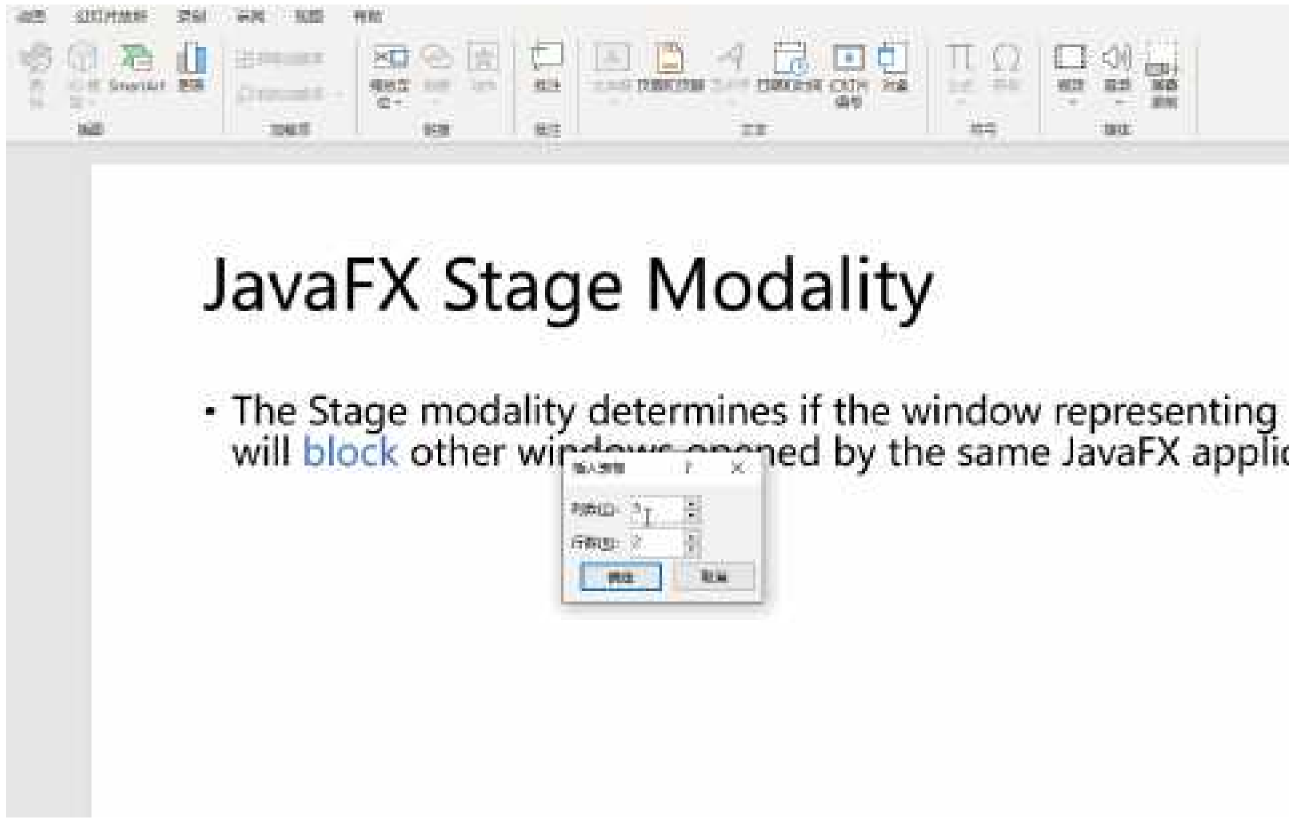


Enum Constants

Enum Constant and Description
`DECORATED` Defines a normal Stage style with a solid white background and platform decorations.
`TRANSPARENT` Defines a Stage style with a transparent background and no decorations.
`UNDECORATED` Defines a Stage style with a solid white background and no decorations.
`UNIFIED` Defines a Stage style with platform decorations and eliminates the border between client area and decorations.
`UTILITY` Defines a Stage style with a solid white background and minimal platform decorations used for a utility window.

JavaFX Stage Modality

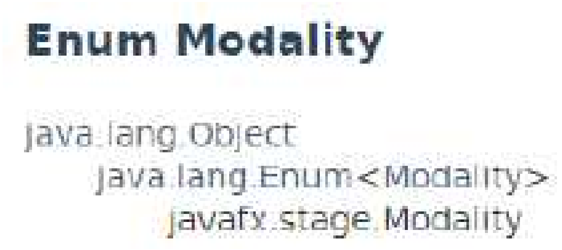
The Stage modality determines if the window representing the Stage will **block** other windows opened by the same JavaFX application.



JavaFX Stage Modality

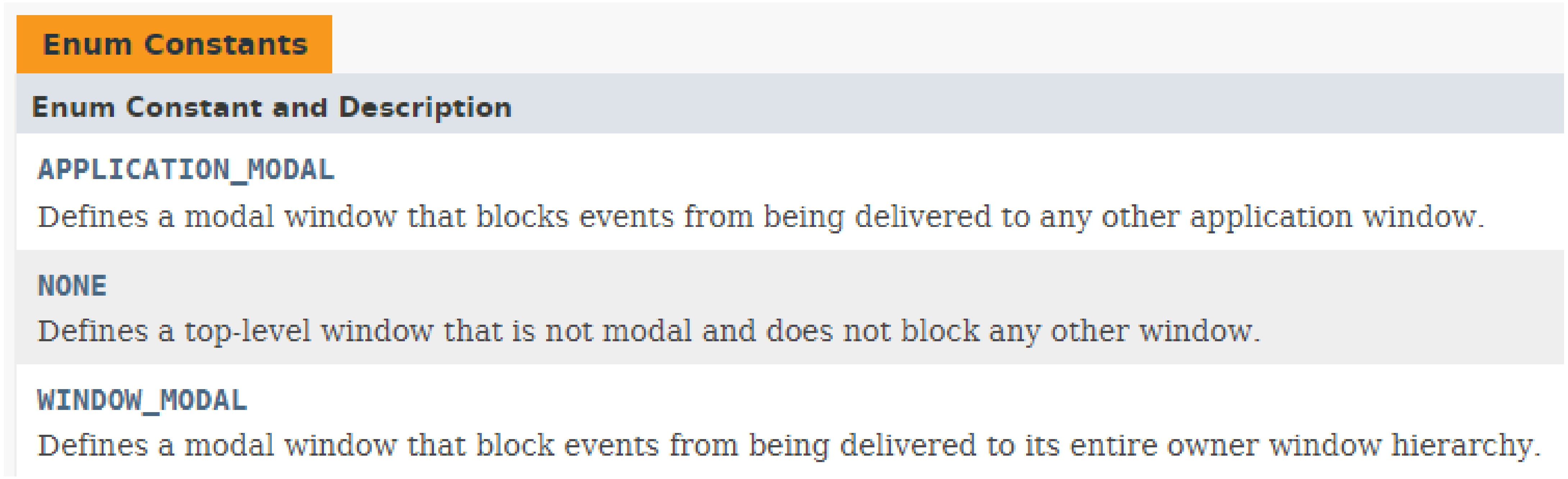
• The Stage modality determines if the window representing
will block other wir ned by the same JavaFX applic

JavaFX Stage Modality



Enum Modality

java.lang.Object
    java.lang.Enum<Modality>
        javafx.stage.Modality

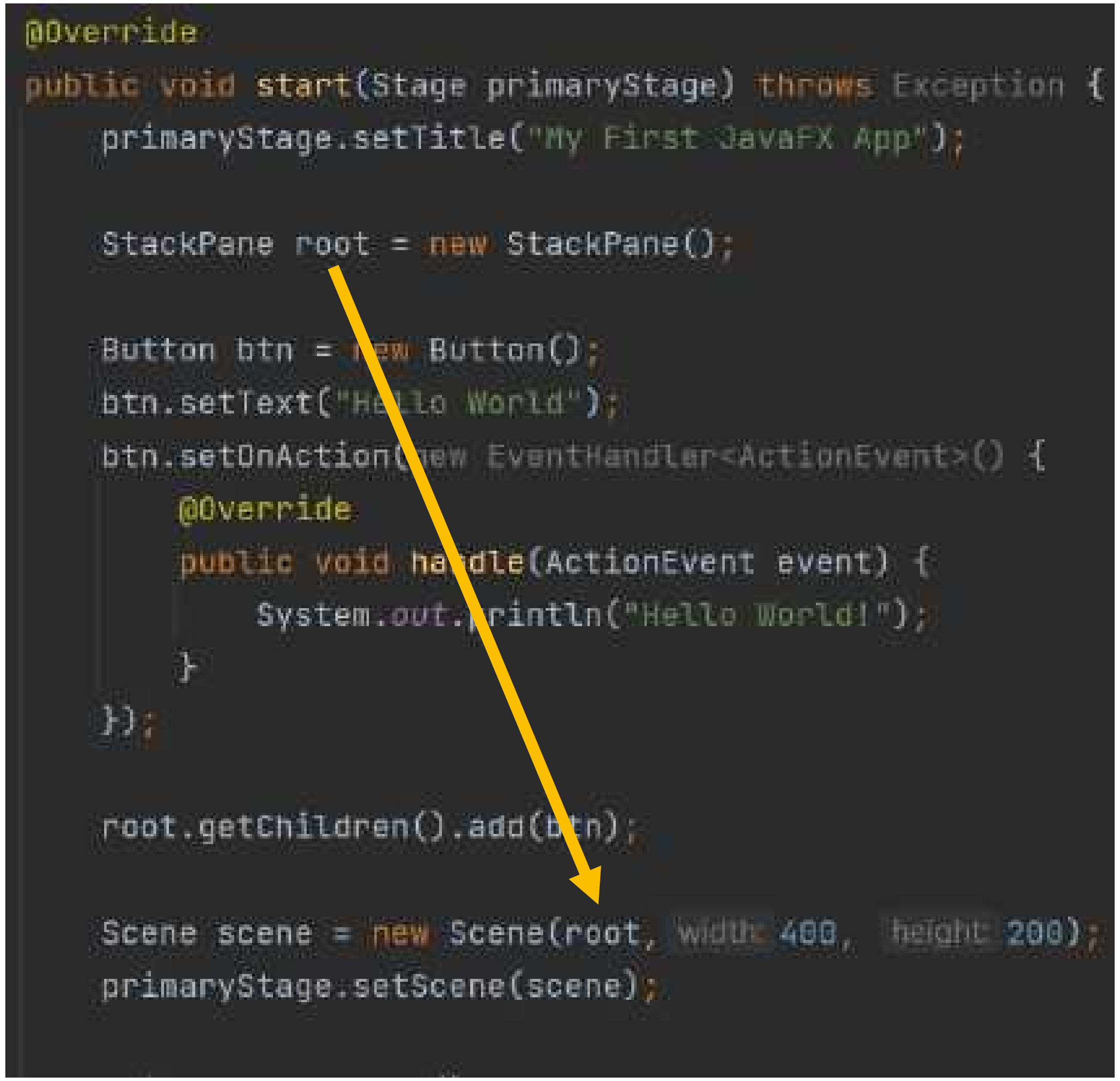


Enum Constants

Enum Constant and Description
`APPLICATION_MODAL` Defines a modal window that blocks events from being delivered to any other application window.
`NONE` Defines a top-level window that is not modal and does not block any other window.
`WINDOW_MODAL` Defines a modal window that block events from being delivered to its entire owner window hierarchy.

JavaFX Scene

- A JavaFX Scene contains all the visual JavaFX GUI components inside it
- A JavaFX Scene object is created by specifying a root GUI component (root node in the Scene Graph)
- A JavaFX Scene must be set on a JavaFX Stage to be visible
- A Scene can be attached to only a single Stage at a time, and Stage can also only display one Scene at a time.

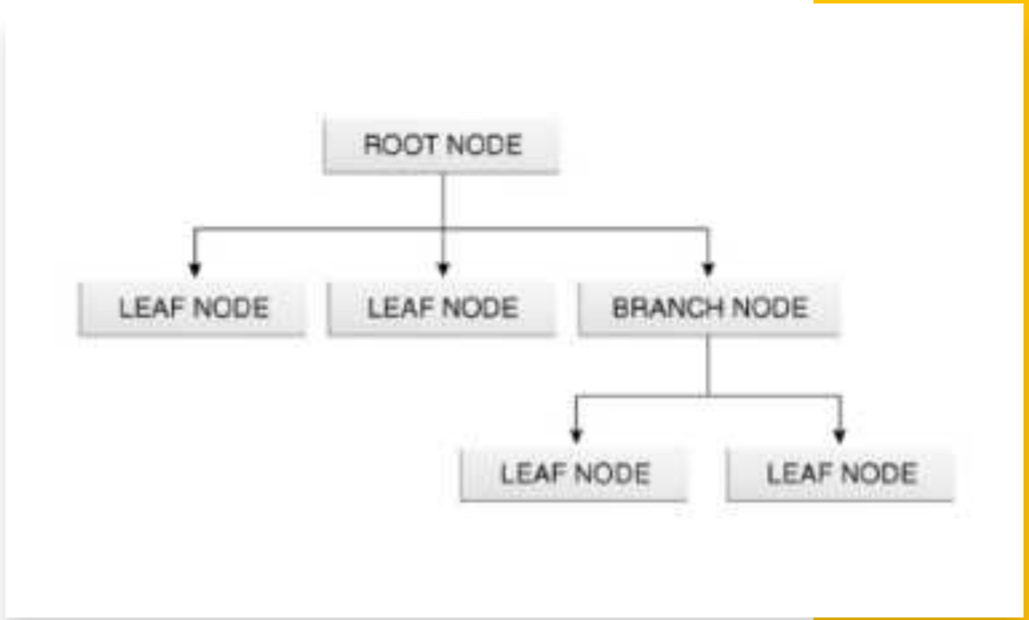


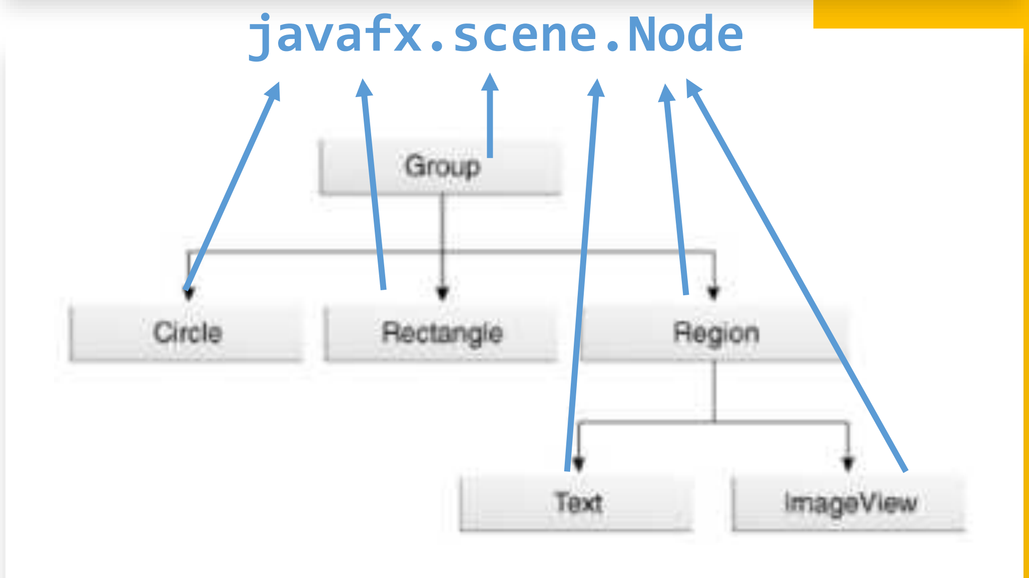
```
@Override
public void start(Stage primaryStage) throws Exception {
    primaryStage.setTitle("My First JavaFX App");

    StackPane root = new StackPane();

    Button btn = new Button();
    btn.setText("Hello World");
    btn.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            System.out.println("Hello World!");
        }
    });

    root.getChildren().add(btn);

    Scene scene = new Scene(root, width: 400, height: 200);
    primaryStage.setScene(scene);
```

Recall: Scene Graph

- A tree data structure of nodes, which is a visual object of a JavaFX application
- A JavaFX application must specify the root node for the scene graph

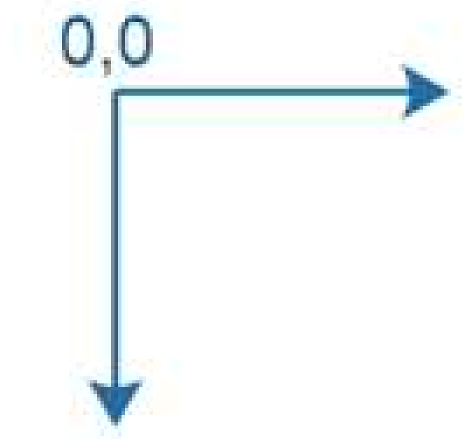
The `javafx.scene.Node` abtract class is the superclass for all GUI components added to the Scene Graph;
All GUI components share some common properties defined in `javafx.scene.Node`

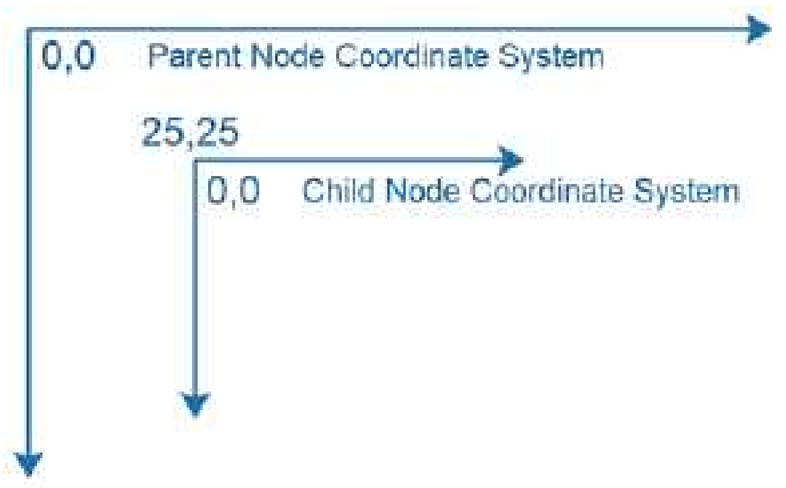




https://docs.oracle.com/javafx/2/scenegraph/jfxpub-scenegraph.htm

JavaFX Node Coordinate System (坐标系统)



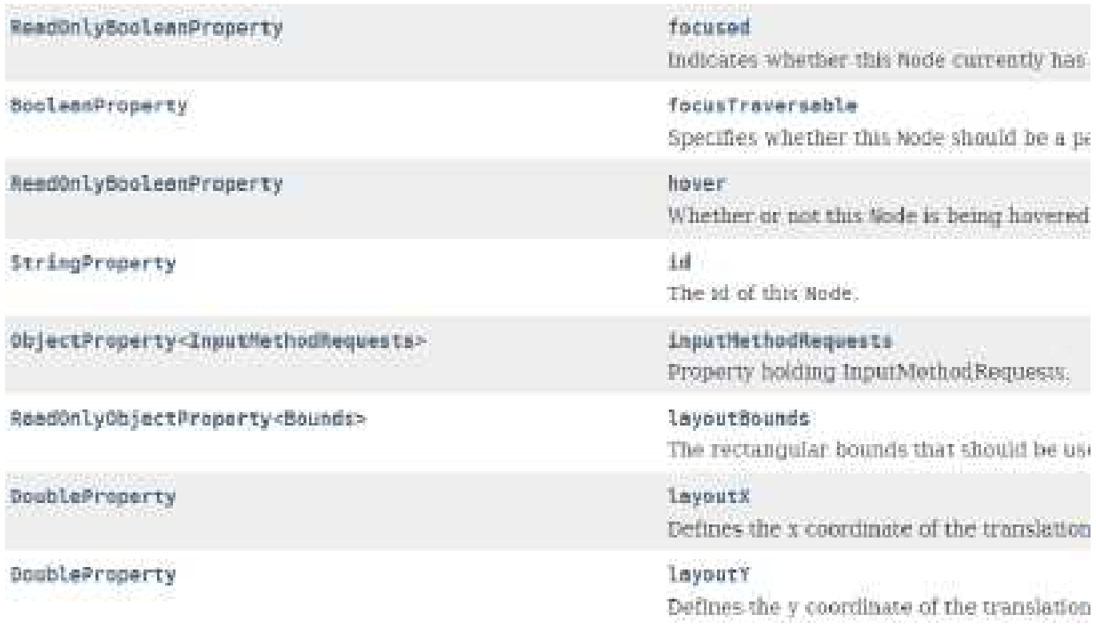


- Each JavaFX Node has its own coordinate system.
- Difference from regular coordinate system: Y axis is reversed
- Use the coordinates to position child Node instances within the parent Node (see layoutX, layoutY)

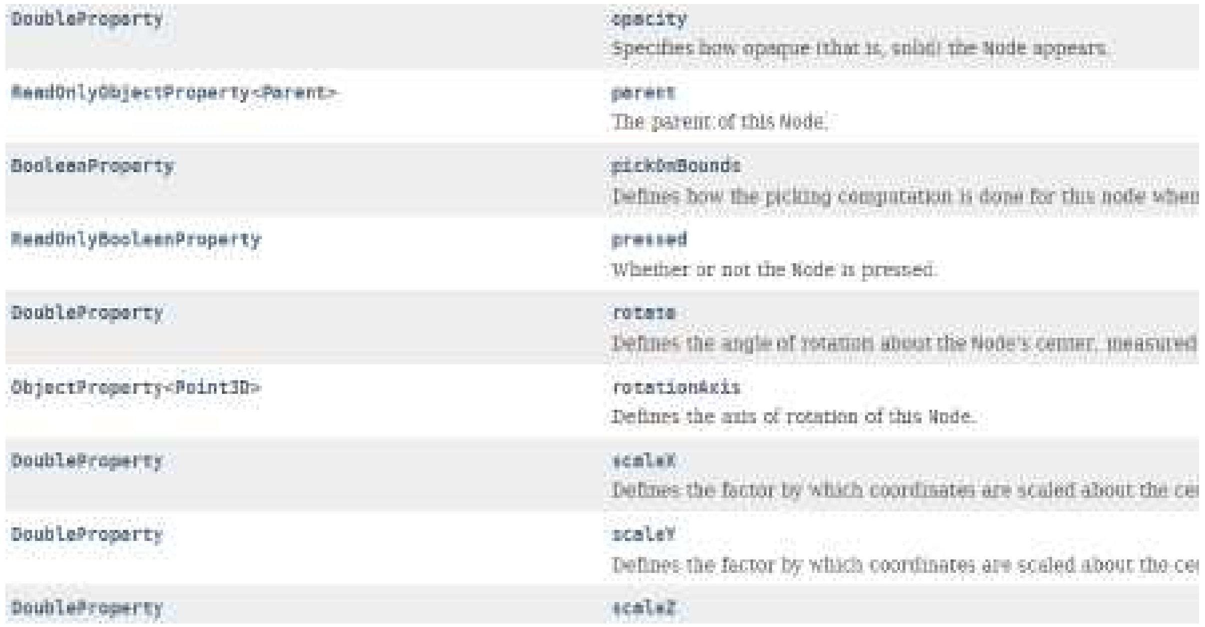
http://tutorials.jenkov.com/javafx/node.html

JavaFX Node Property

(Writable) properties include X and Y position, width and height, text, children, event handlers, etc.



ReadOnlyBooleanProperty	focused Indicates whether this Node currently has
BooleanProperty	focusTraversable Specifies whether this Node should be a pa
ReadOnlyBooleanProperty	hover Whether or not this Node is being hovered
StringProperty	id The id of this Node.
ObjectProperty<InputMethodRequests>	inputMethodRequests Property holding InputMethodRequests.
ReadOnlyObjectProperty<Bounds>	layoutBounds The rectangular bounds that should be us
DoubleProperty	layoutX Defines the x coordinate of the translation
DoubleProperty	layoutY Defines the y coordinate of the translation

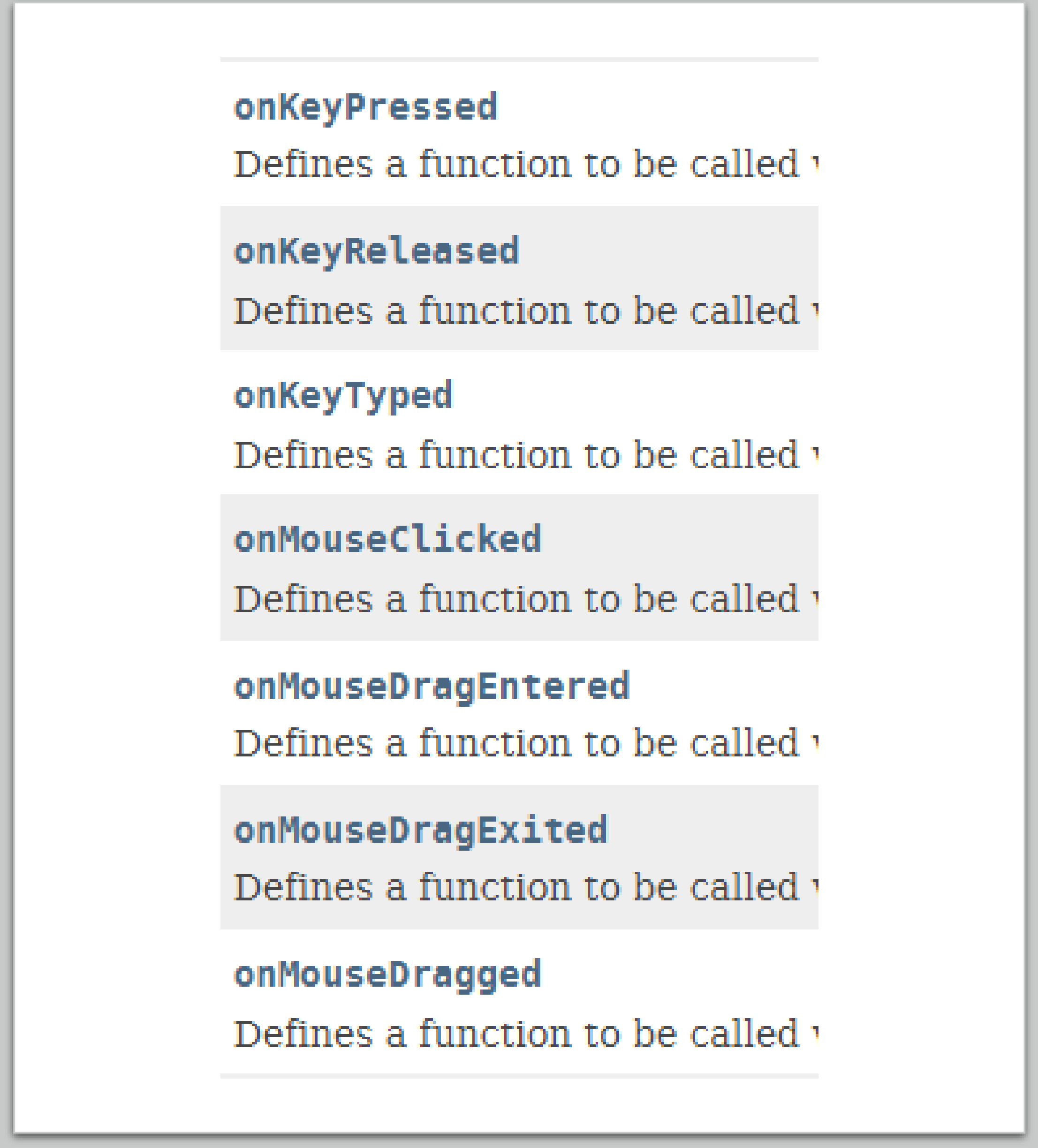


DoubleProperty	opacity Specifies how opaque (that is, solid) the Node appears.
ReadOnlyObjectProperty<Parent>	parent The parent of this Node.
BooleanProperty	pickOnBounds Defines how the picking computation is done for this node when
ReadOnlyBooleanProperty	pressed Whether or not the Node is pressed.
DoubleProperty	rotate Defines the angle of rotation about the Node's center, measured
ObjectProperty<Point3D>	rotationAxis Defines the axis of rotation of this Node.
DoubleProperty	scaleX Defines the factor by which coordinates are scaled about the ce
DoubleProperty	scaleY Defines the factor by which coordinates are scaled about the ce
DoubleProperty	scaleZ

JavaFX Node EventHandler Property

Node contains various Event Handler properties which can be set to user defined Event Handlers using the setter methods

Setter Naming Convention
`setOnTargetType(EventHandler<TargetEvent> v)`



onKeyPressed
Defines a function to be called

onKeyReleased
Defines a function to be called

onKeyTyped
Defines a function to be called

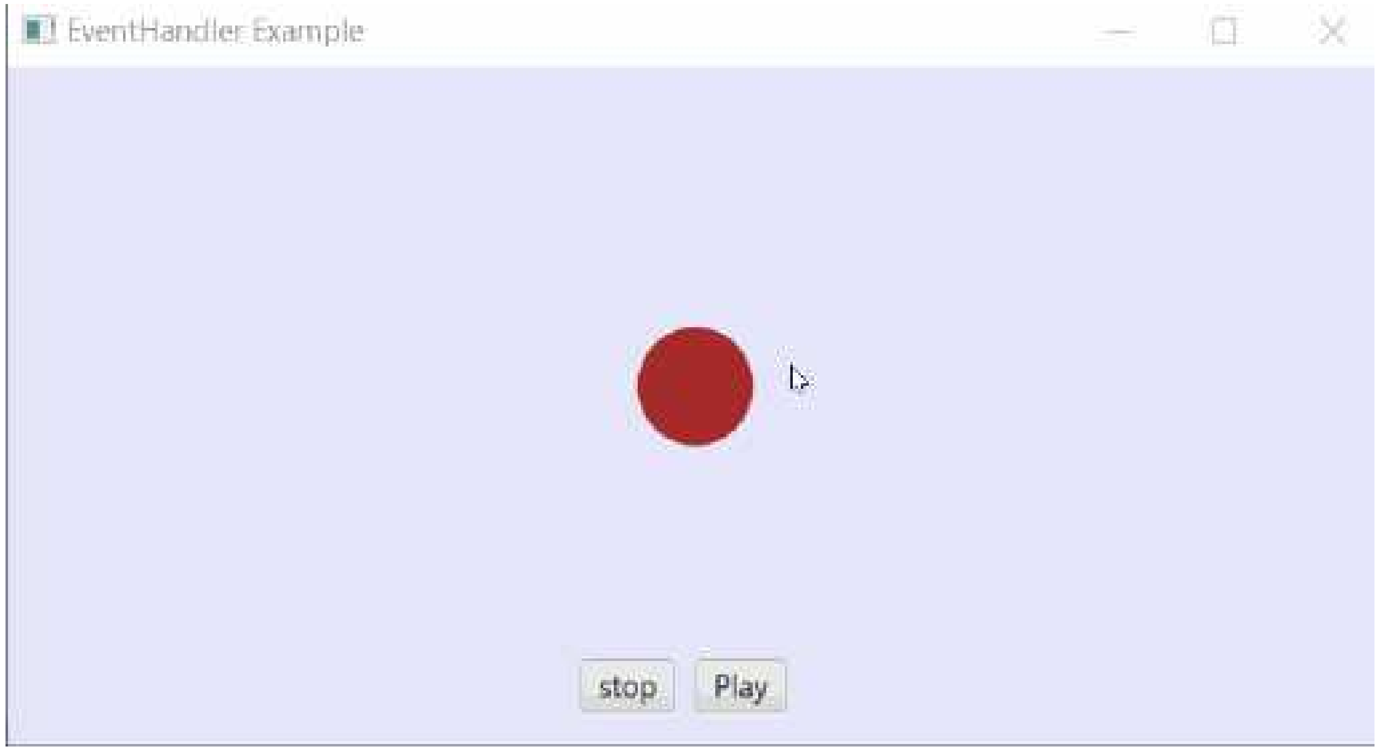
onMouseClicked
Defines a function to be called

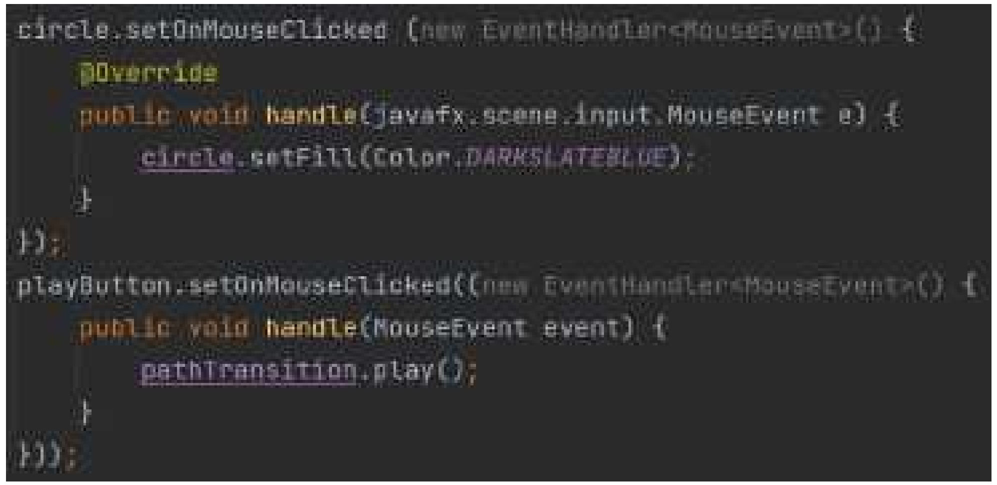
onMouseDragEntered
Defines a function to be called

onMouseDragExited
Defines a function to be called

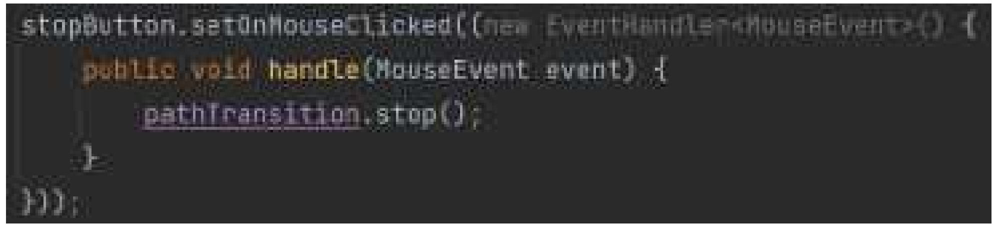
onMouseDragged
Defines a function to be called

How many events? What event handlers on which target?





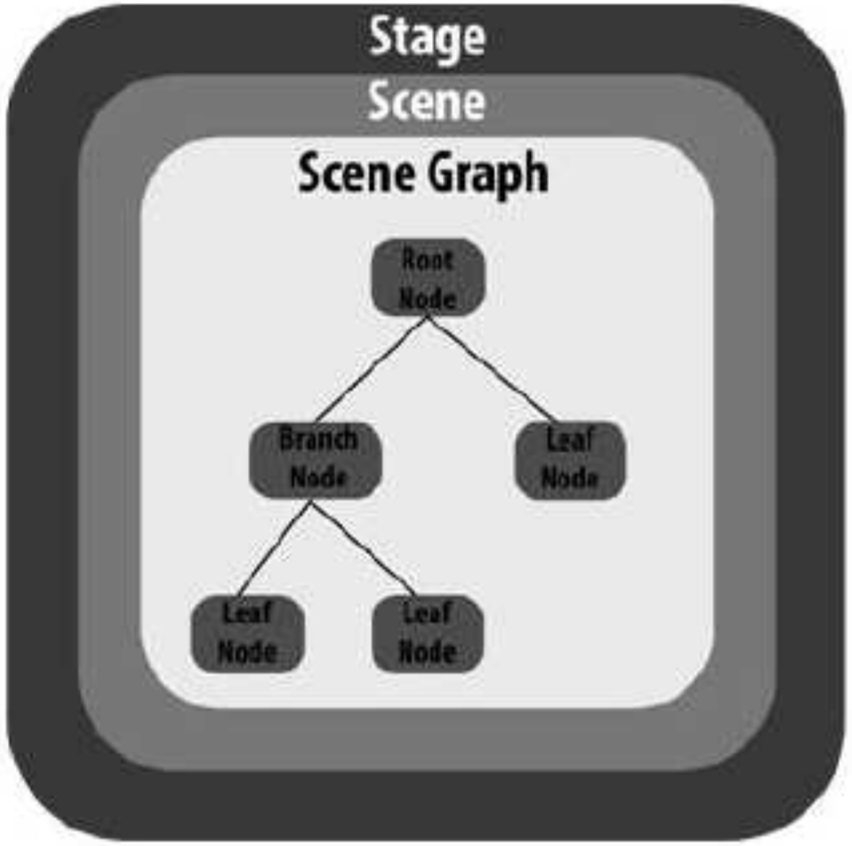
```
circle.setOnMouseClicked (new EventHandler<MouseEvent>() {
    @Override
    public void handle(javafx.scene.input.MouseEvent e) {
        circle.setFill(Color.DARKSLATEBLUE);
    }
});
playButton.setOnMouseClicked((new EventHandler<MouseEvent>() {
    public void handle(MouseEvent event) {
        pathTransition.play();
    }
}));
```



```
stopButton.setOnMouseClicked((new EventHandler<MouseEvent>() {
    public void handle(MouseEvent event) {
        pathTransition.stop();
    }
}));
```

Full example code: https://www.tutorialspoint.com/javafx/javafx_event_handling.htm

So far…



Next: Layout, Shapes, Controls

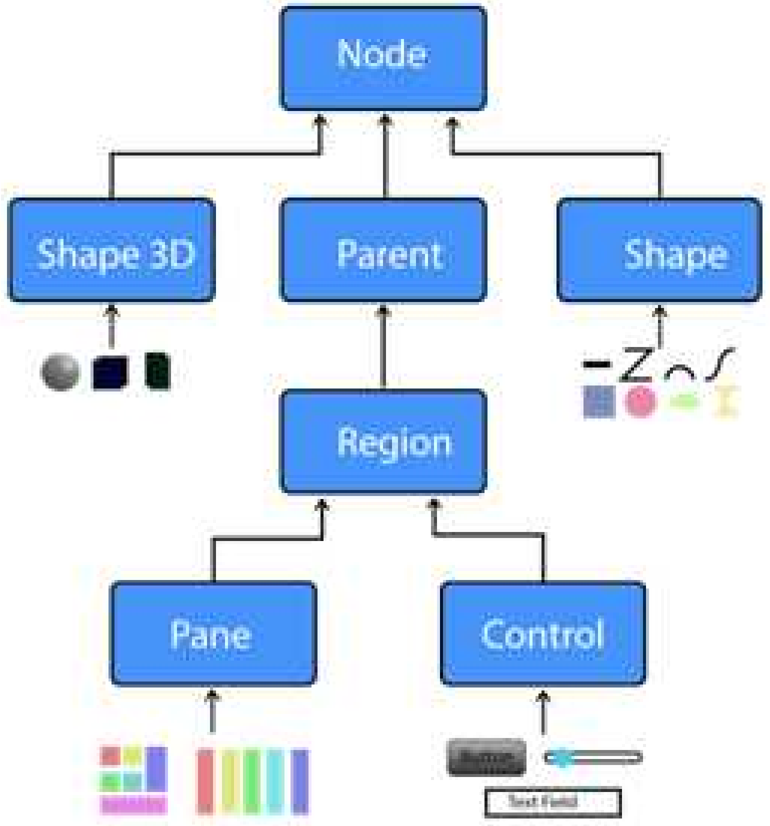
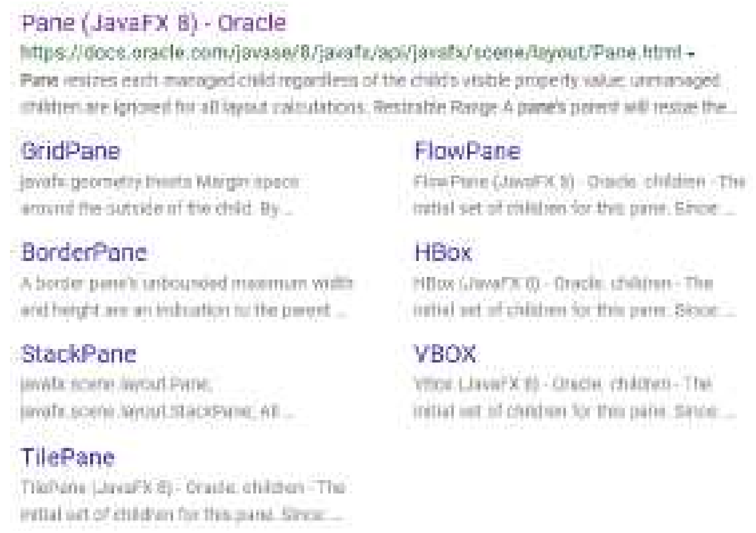


Image source :https://www.javatpoint.com/javafx-application-structure

JavaFX Layout

- Top-level container that organizes nodes in the scene graph
- `javafx.scene.layout` package provides various classes that represent the layouts
- `javafx.scene.layout.Pane` class is the parent class for all these built-in layout classes

```
Pane canvas = new Pane();
canvas.setStyle("-fx-background-color: black;");
canvas.setPrefSize(200,200);
Circle circle = new Circle(50,Color.BLUE);
circle.relocate(20, 20);
Rectangle rectangle = new Rectangle(100,100,Color.RED);
rectangle.relocate(70,70);
canvas.getChildren().addAll(circle,rectangle);
```



Pane (JavaFX 8) - Oracle
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/Pane.html
Pane resizes each managed child regardless of the child's visible property value; unmanaged children are ignored for all layout calculations. Resizable Range A pane's parent will resize the ...

GridPane
javafx.geometry.Insets Margin space around the outside of the child. By ...

FlowPane
FlowPane (JavaFX 8) - Oracle. children - The initial set of children for this pane. Since ...

BorderPane
A border pane's unbounded maximum width and height are an indication to the parent ...

HBox
HBox (JavaFX 8) - Oracle. children - The initial set of children for this pane. Since ...

StackPane
javafx.scene.layout.Pane; javafx.scene.layout.StackPane; All ...

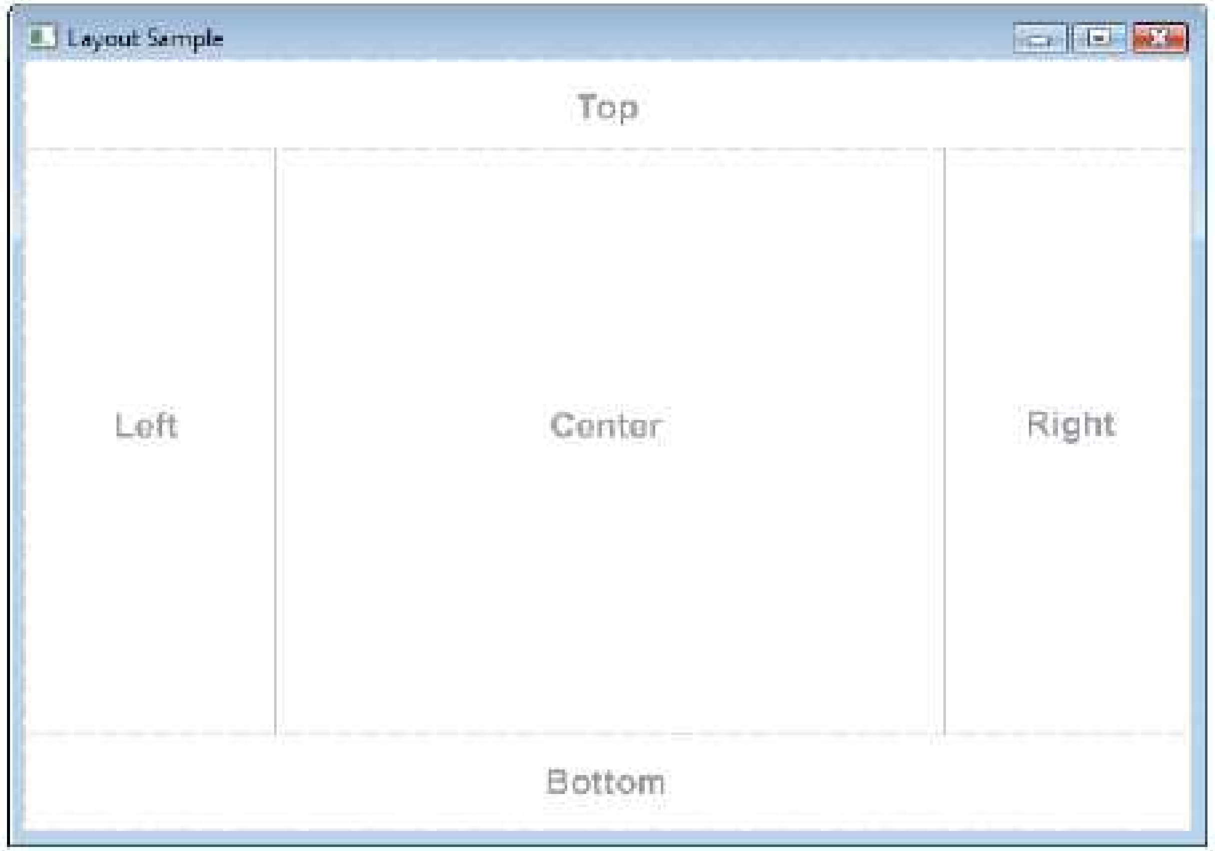
VBOX
VBox (JavaFX 8) - Oracle. children - The initial set of children for this pane. Since ...

TilePane
TilePane (JavaFX 8) - Oracle. children - The initial set of children for this pane. Since ...

BorderPane

The BorderPane layout pane provides five regions in which to place nodes: top, bottom, left, right, and center.

Figure 1-1 Sample Border Pane



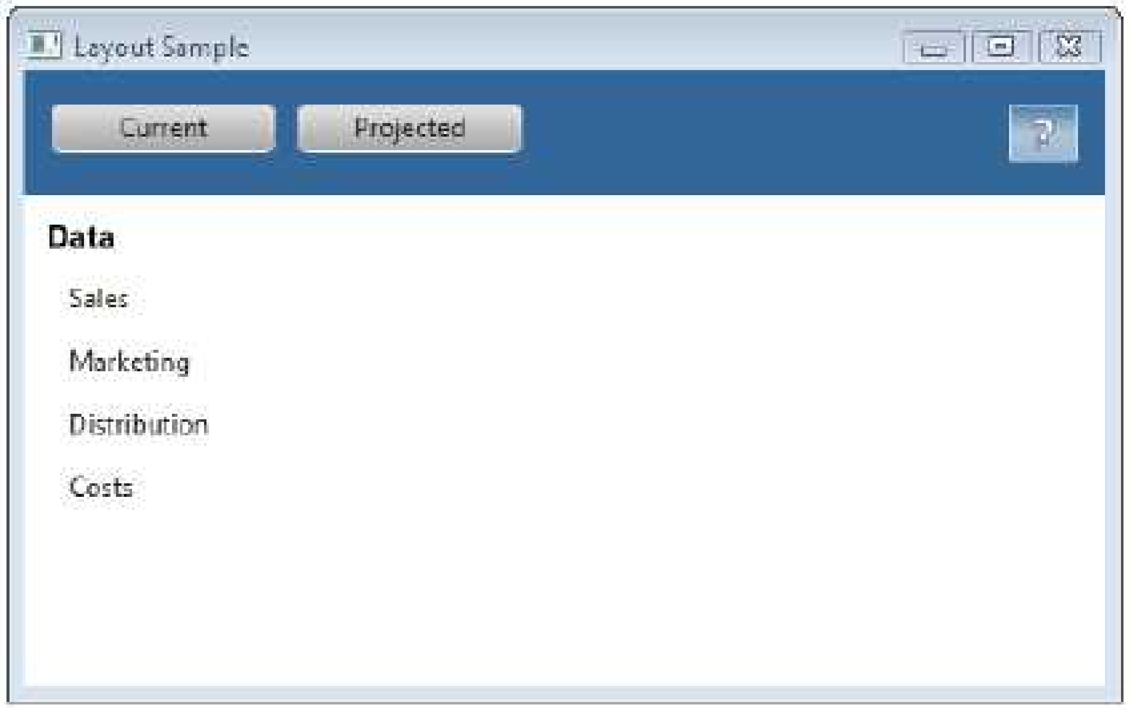
For more details:
https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm

HBox & VBox Pane

- The HBox layout pane provides an easy way for arranging a series of nodes in a single row
- The VBox layout pane provides an easy way for arranging a series of nodes in a single column

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

Figure 1-5 VBox Pane in a Border Pane

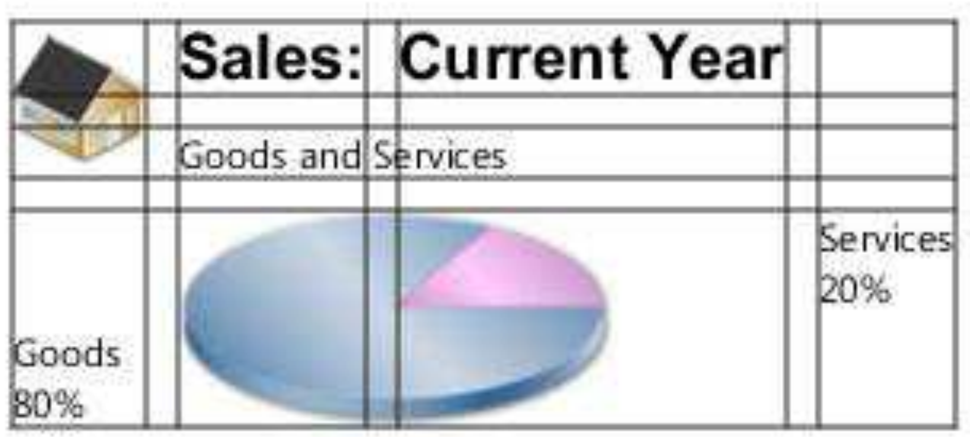


GridPane

The GridPane layout pane enables you to create a flexible grid of rows and columns in which to lay out nodes.

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

Figure 1-8 Sample Grid Pane



Combine Panes

Different Panes can be combined to make beautiful layout

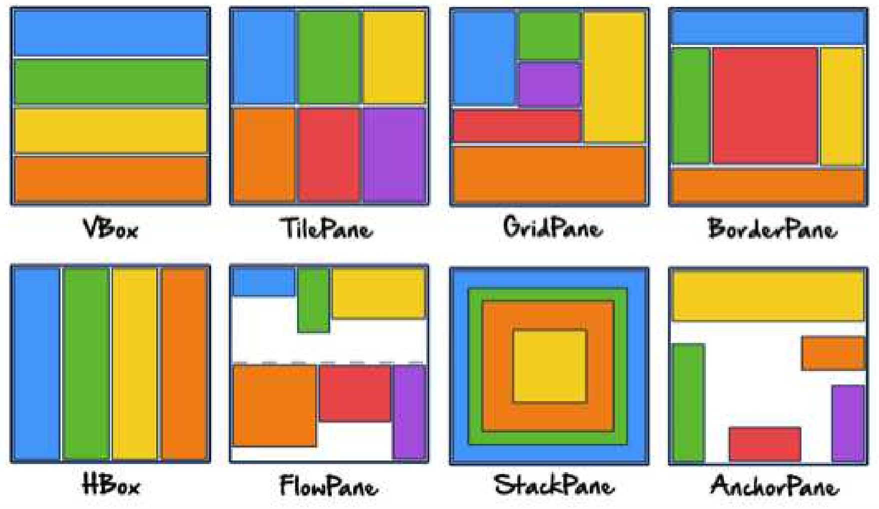
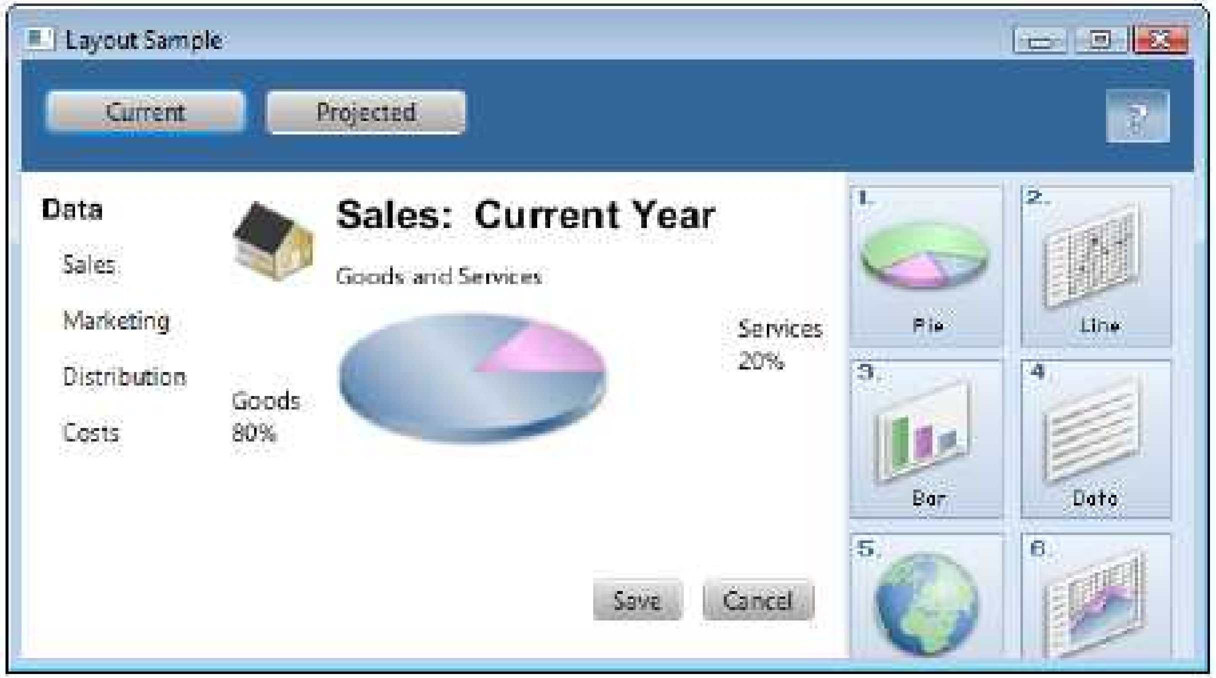


Image source: https://dzone.com/refcardz/javafx-8-1

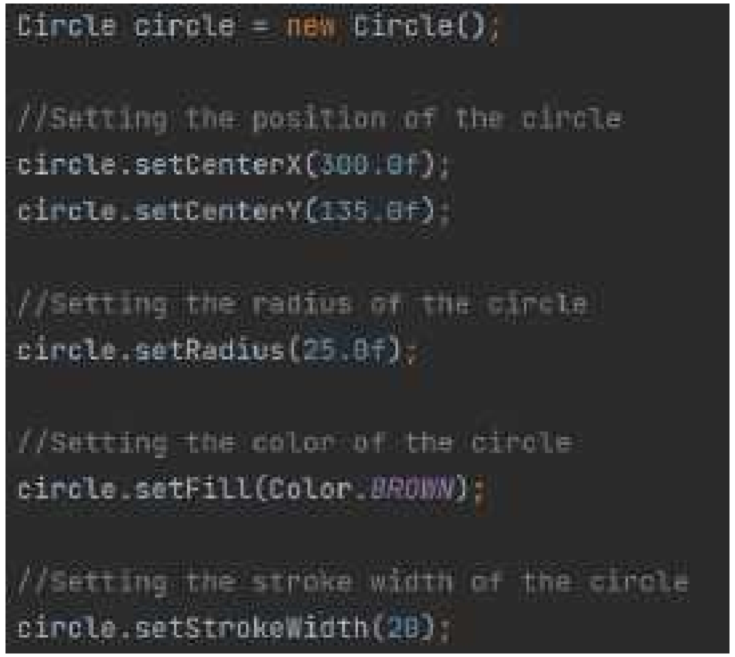


For more details:
https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm

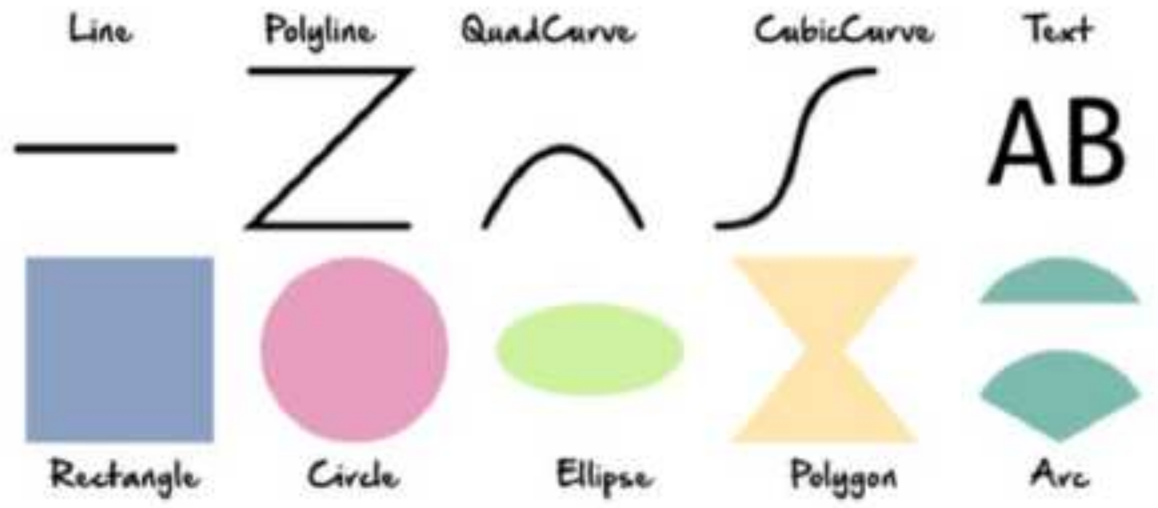
Image source: https://dzone.com/refcardz/javafx-8-1

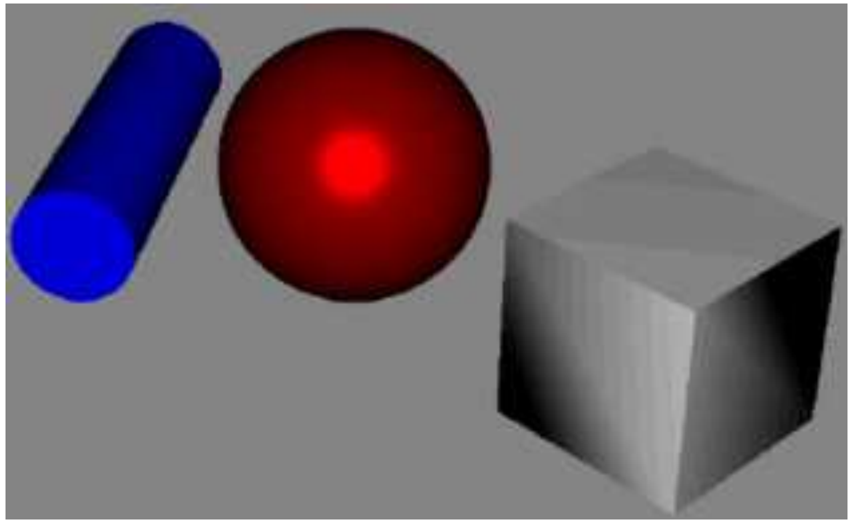
JavaFX Shape

The Shape class is the superclass of all geometric shapes



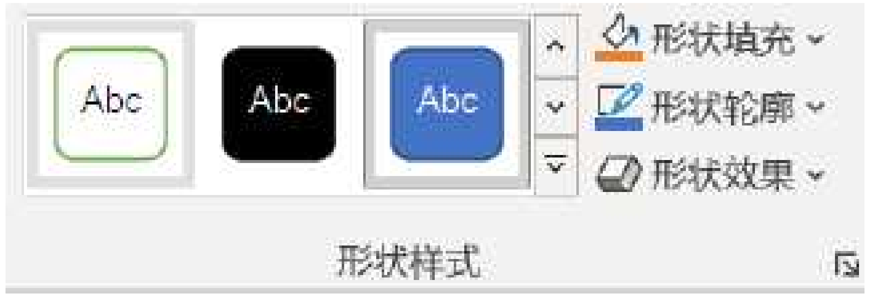
```
Circle circle = new Circle();

//Setting the position of the circle
circle.setCenterX(300.0f);
circle.setCenterY(135.0f);

//Setting the radius of the circle
circle.setRadius(25.0f);

//Setting the color of the circle
circle.setFill(Color.BROWN);

//Setting the stroke width of the circle
circle.setStrokeWidth(20);
```





Shape Properties

- Fill
- Stroke/Outline
- Decoration styles



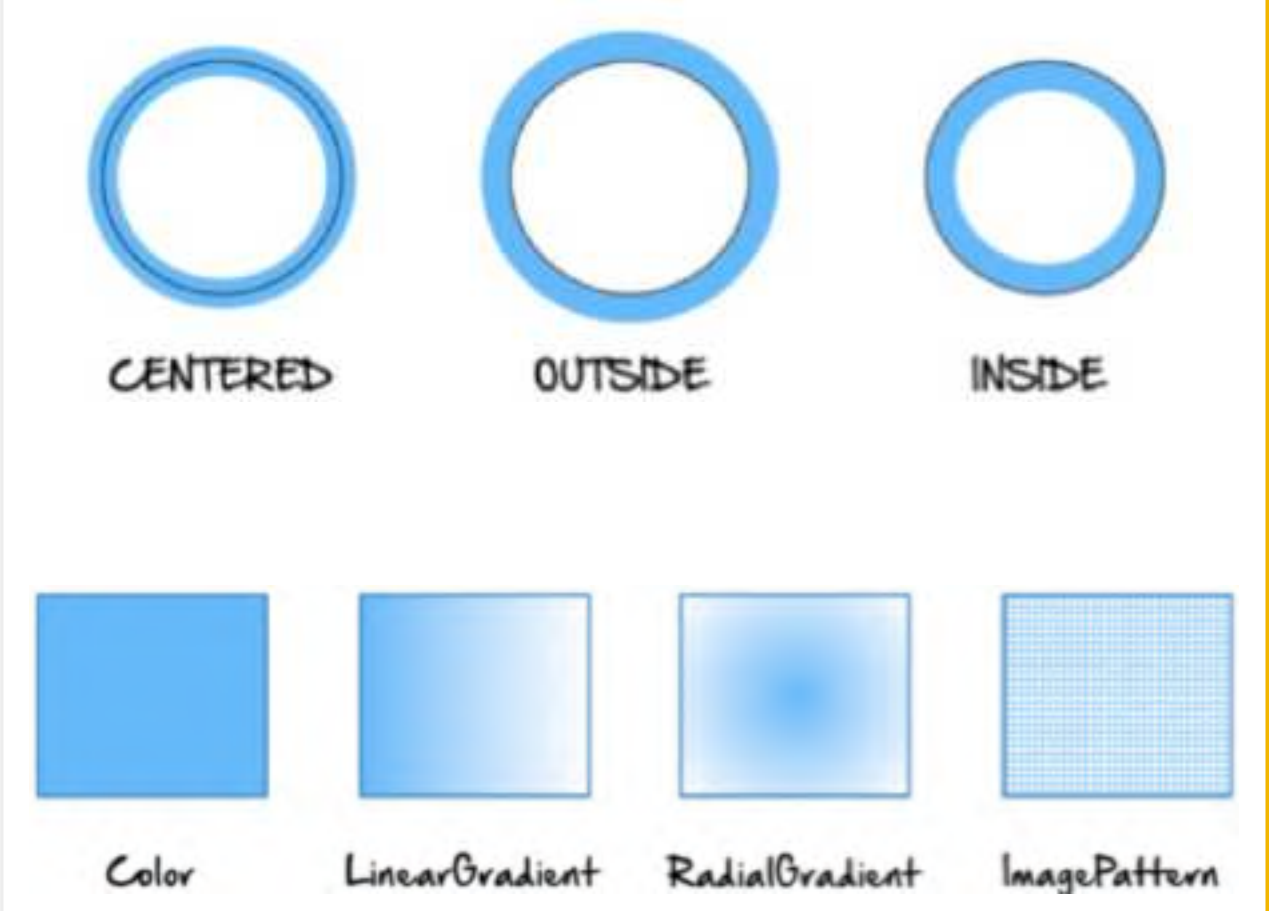


Image source: https://dzone.com/refcardz/javafx-8-1

Shape Operations

We could use operations including intersect, union, and subtract to create new shapes

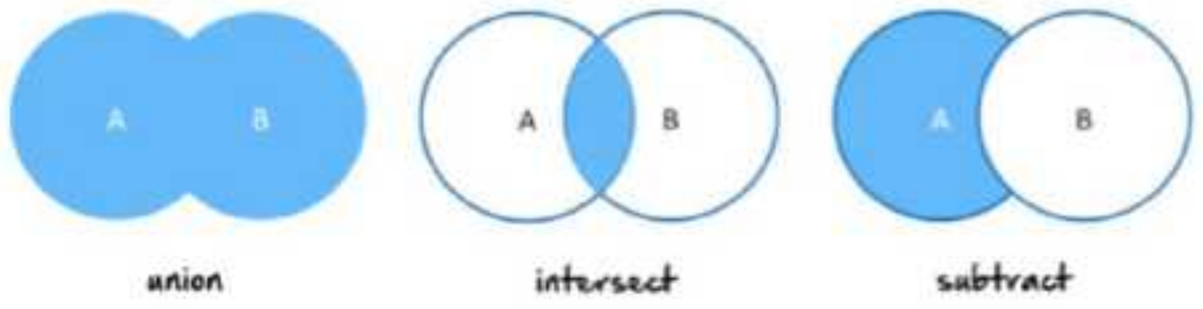


Image source: https://dzone.com/refcardz/javafx-8-1

JavaFX UI Controls

- A "Control" is a node in the scene graph which can be manipulated by the user
- The Control class is the base class of all controls (e.g., buttons, tables, textfields, etc.)

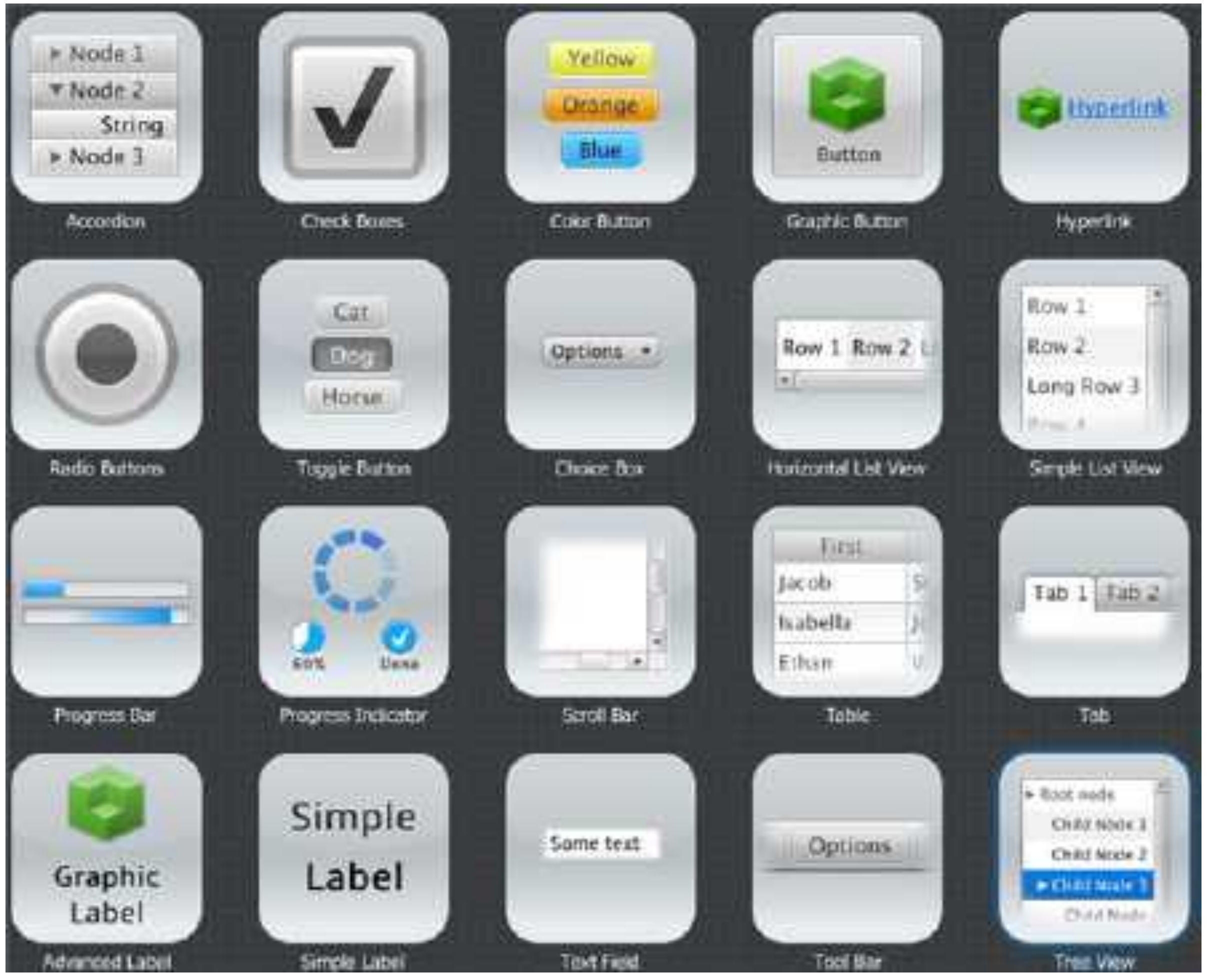
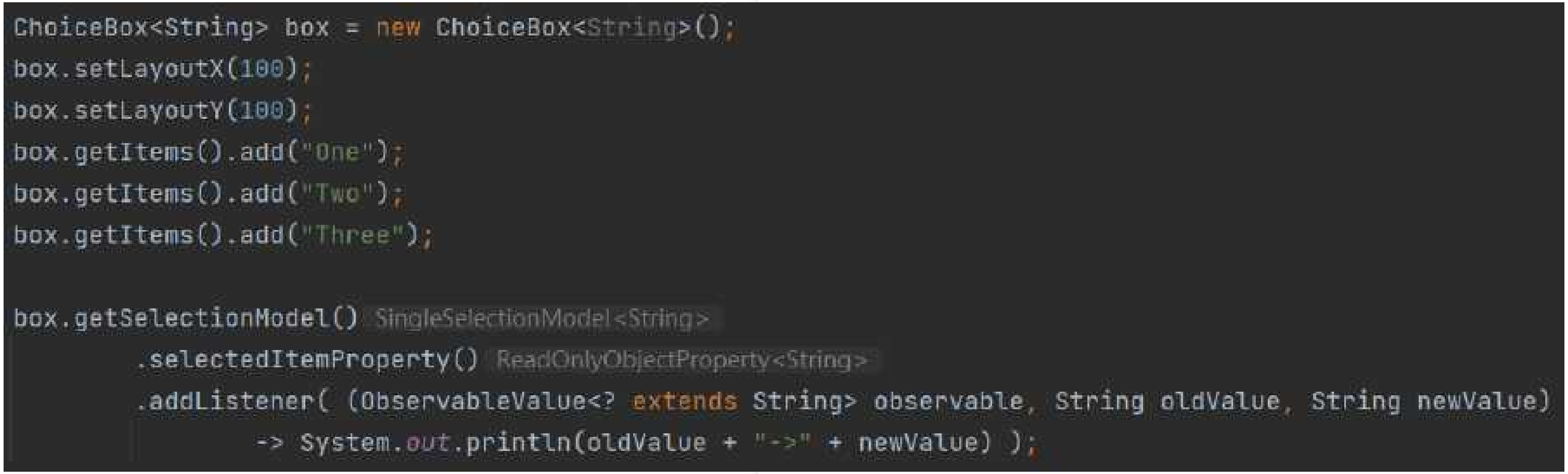


Image source: https://docs.oracle.com/javafx/2/ui_controls/overview.htm

Example: ChoiceBox

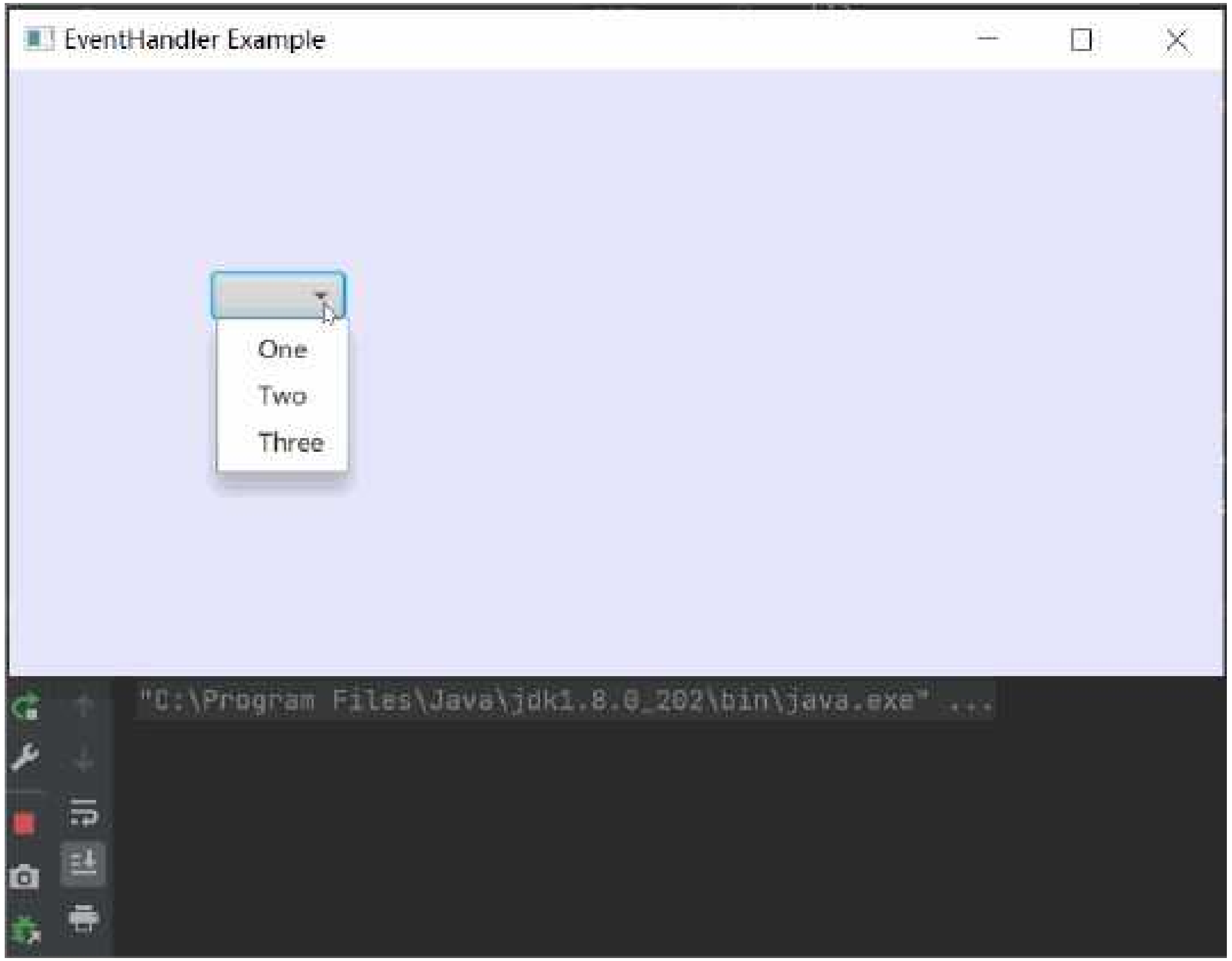
Add a ChangeListener which will be notified whenever the value of the choicebox changes.



```
ChoiceBox<String> box = new ChoiceBox<String>();
box.setLayoutX(100);
box.setLayoutY(100);
box.getItems().add("One");
box.getItems().add("Two");
box.getItems().add("Three");

box.getSelectionModel() SingleSelectionModel<String>
        .selectedItemProperty() ReadOnlyObjectProperty<String>
        .addListener( (ObservableValue<? extends String> observable, String oldValue, String newValue)
                -> System.out.println(oldValue + "->" + newValue) );
```

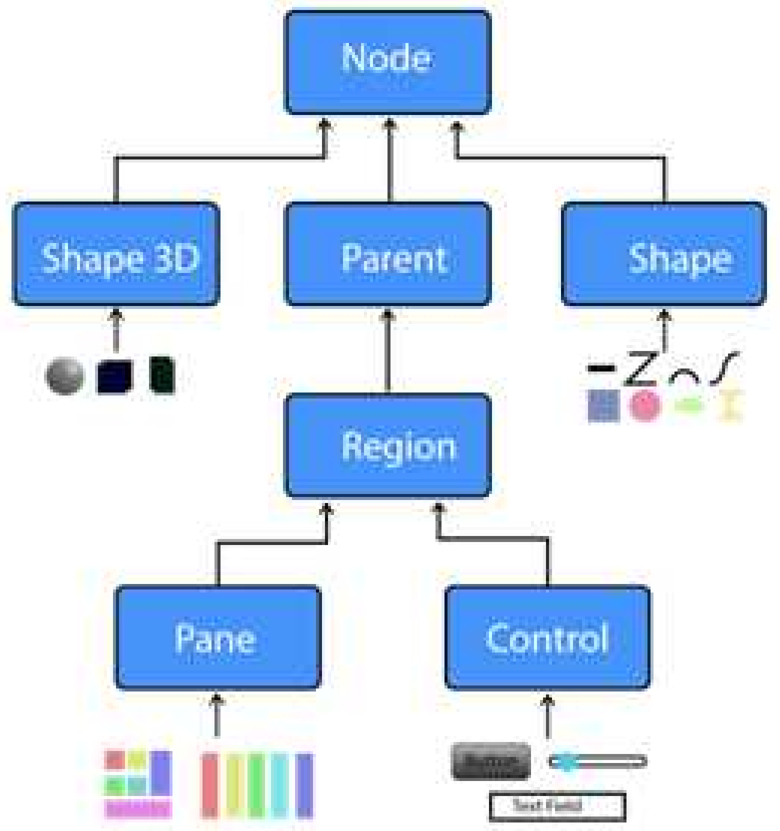
ChangeListener is functional interface, you can use lambda here

Example: ChoiceBox



```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
```

So far…



Next: Charts (图表)

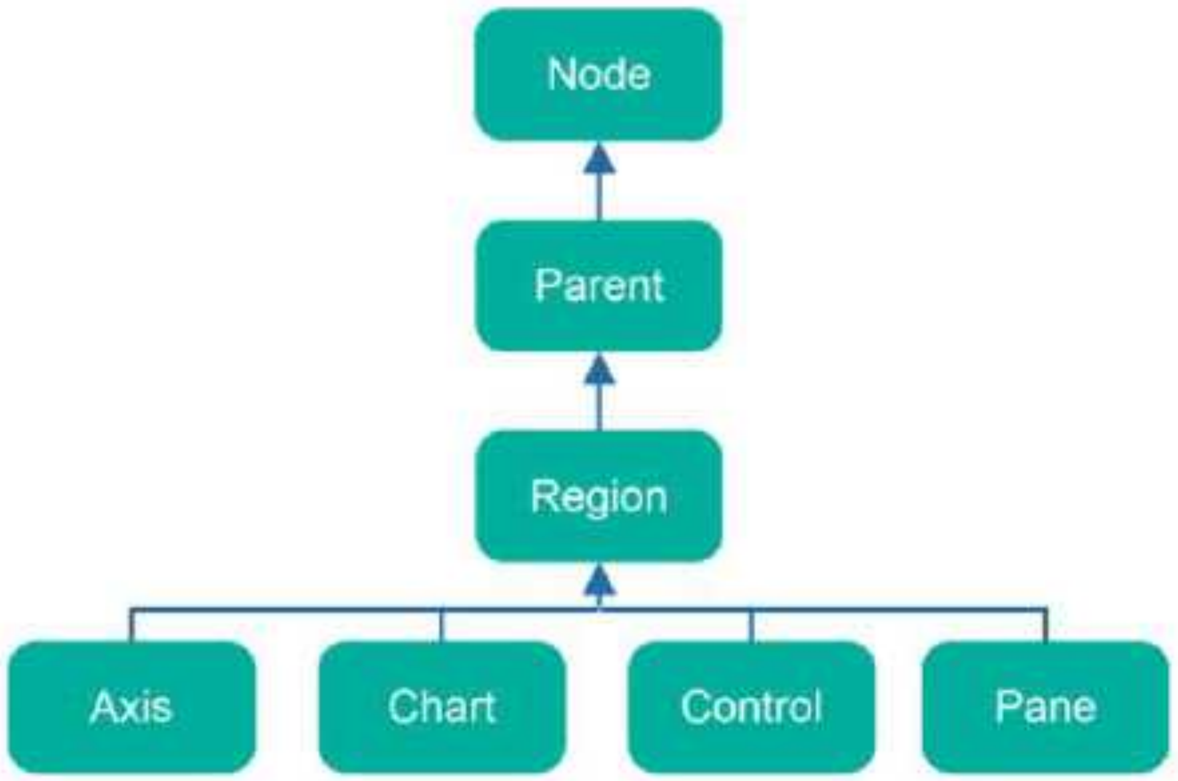


Image source :https://www.javatpoint.com/javafx-application-structure,
http://tutorials.jenkov.com/javafx/region.html

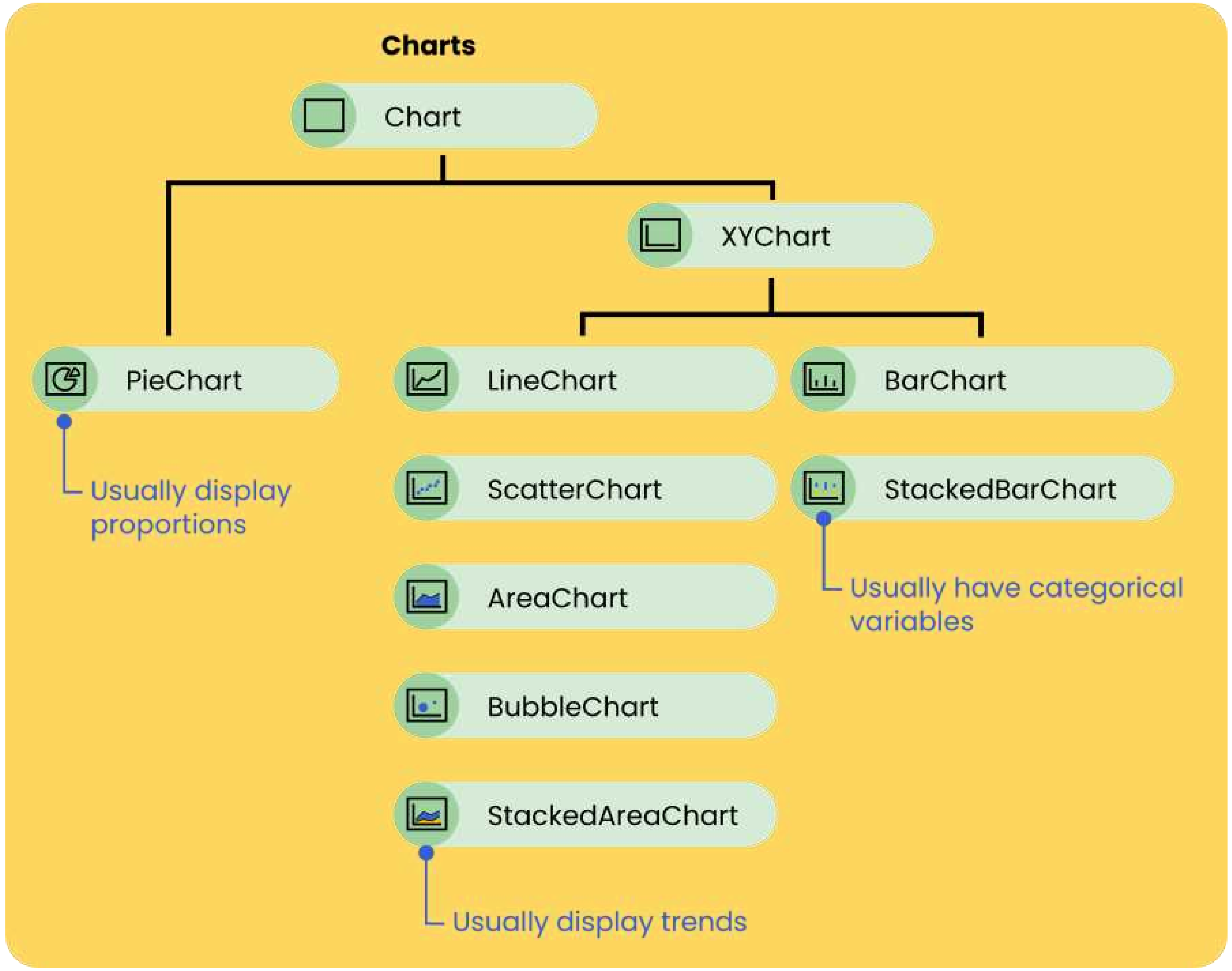
JavaFX Charts

- Chart: a graphical representation of data in the form of symbols
- JavaFX `Chart` (`javafx.scene.chart.Chart`) is the base class for all charts. It has 3 parts:
  - Title
  - Legend (图例)
  - chartContent

Types of Charts

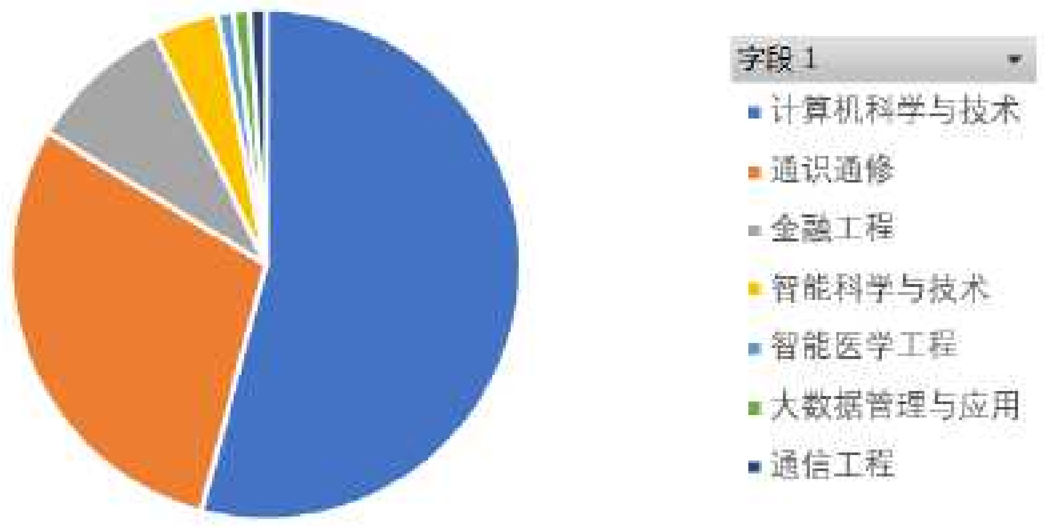
JavaFX provides 8 default charts to display data, which fall in two types (PieChart & XYChart)

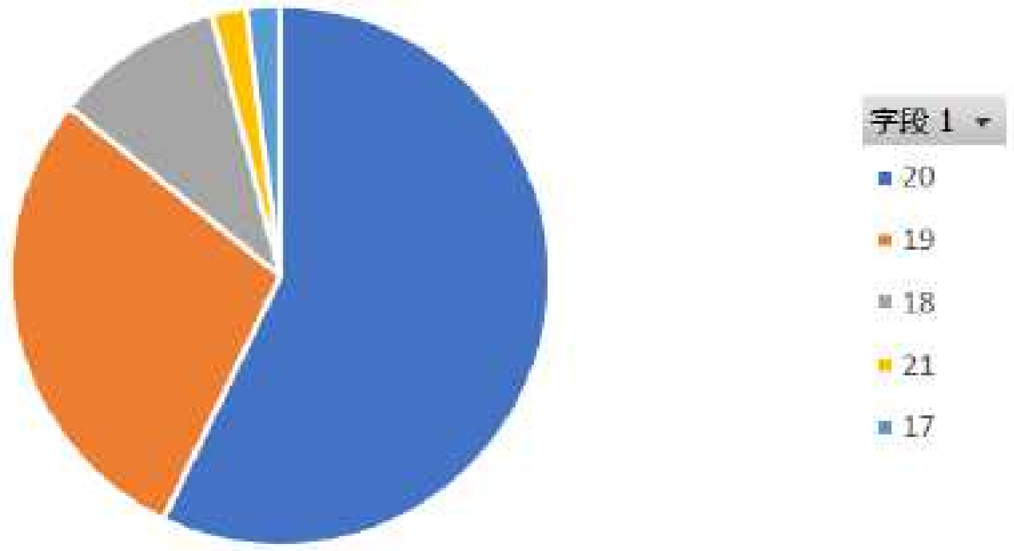
https://edencoding.com/javafx-charts/



PieChart (饼图)

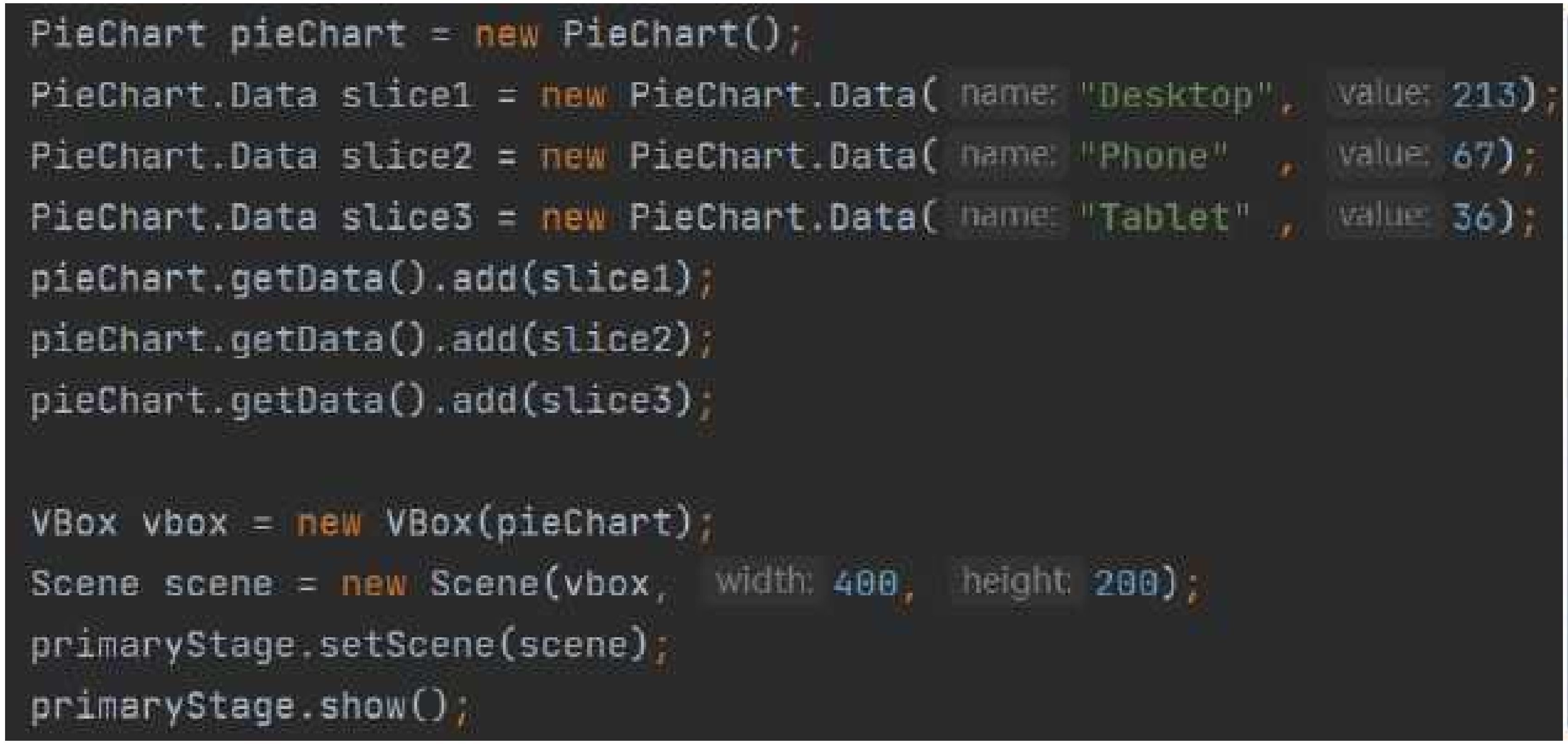
Works the best to find out the composition of something



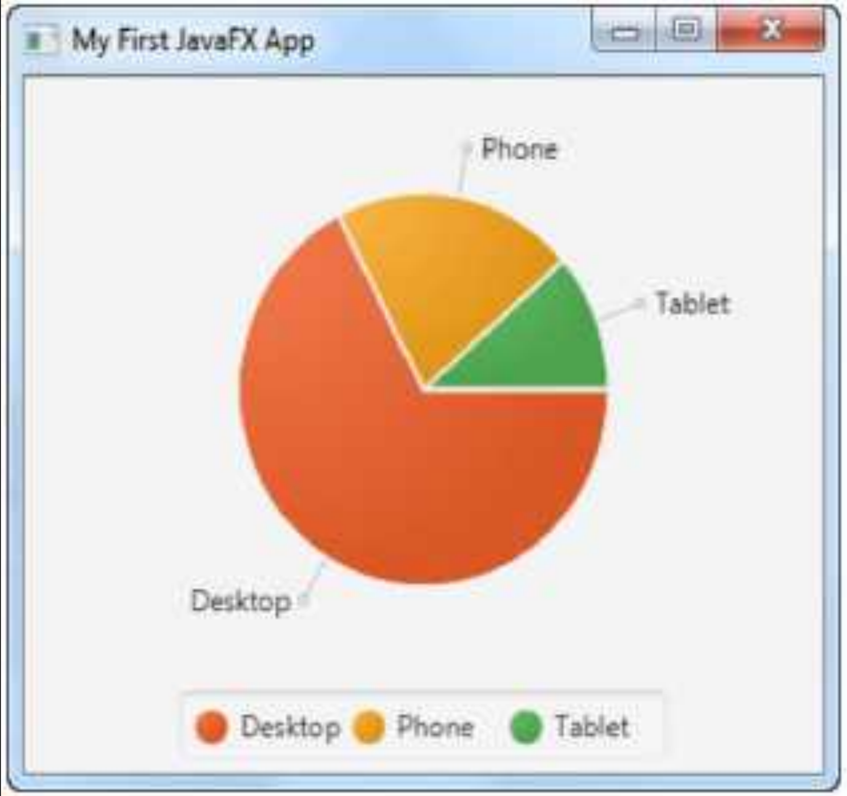


PieChart (饼图)

Works the best to find out the composition of something



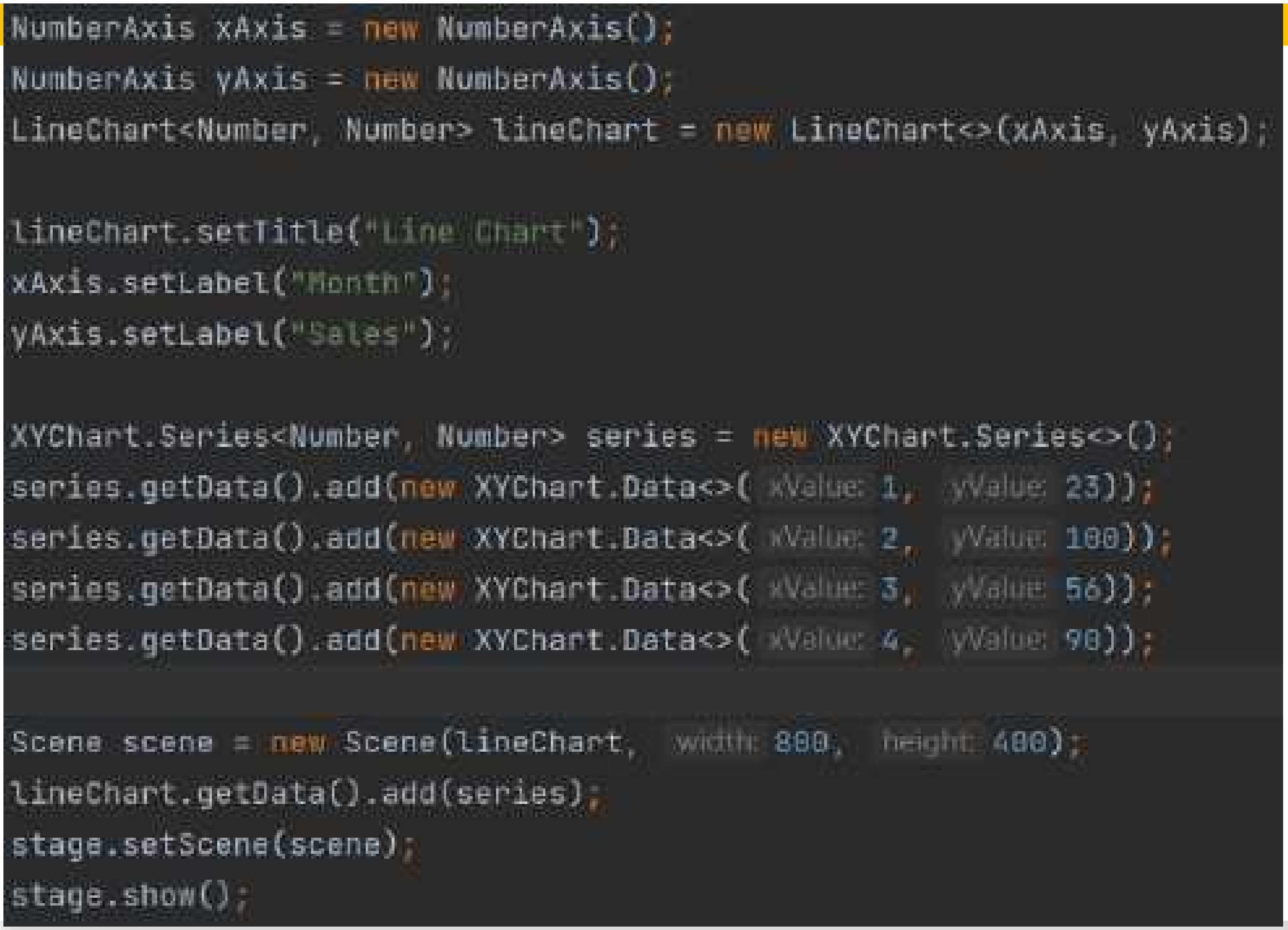
```
PieChart pieChart = new PieChart();
PieChart.Data slice1 = new PieChart.Data( name: "Desktop", value: 213);
PieChart.Data slice2 = new PieChart.Data( name: "Phone"  , value: 67);
PieChart.Data slice3 = new PieChart.Data( name: "Tablet" , value: 36);
pieChart.getData().add(slice1);
pieChart.getData().add(slice2);
pieChart.getData().add(slice3);

VBox vbox = new VBox(pieChart);
Scene scene = new Scene(vbox, width: 400, height: 200);
primaryStage.setScene(scene);
primaryStage.show();
```



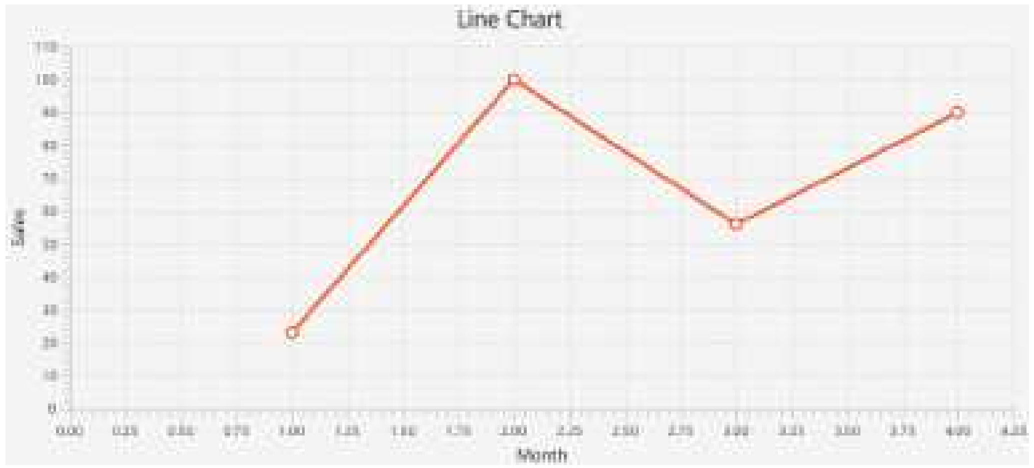
http://tutorials.jenkov.com/javafx/piechart.html

LineChart (折线图)

most often used to visualize data that changes over time

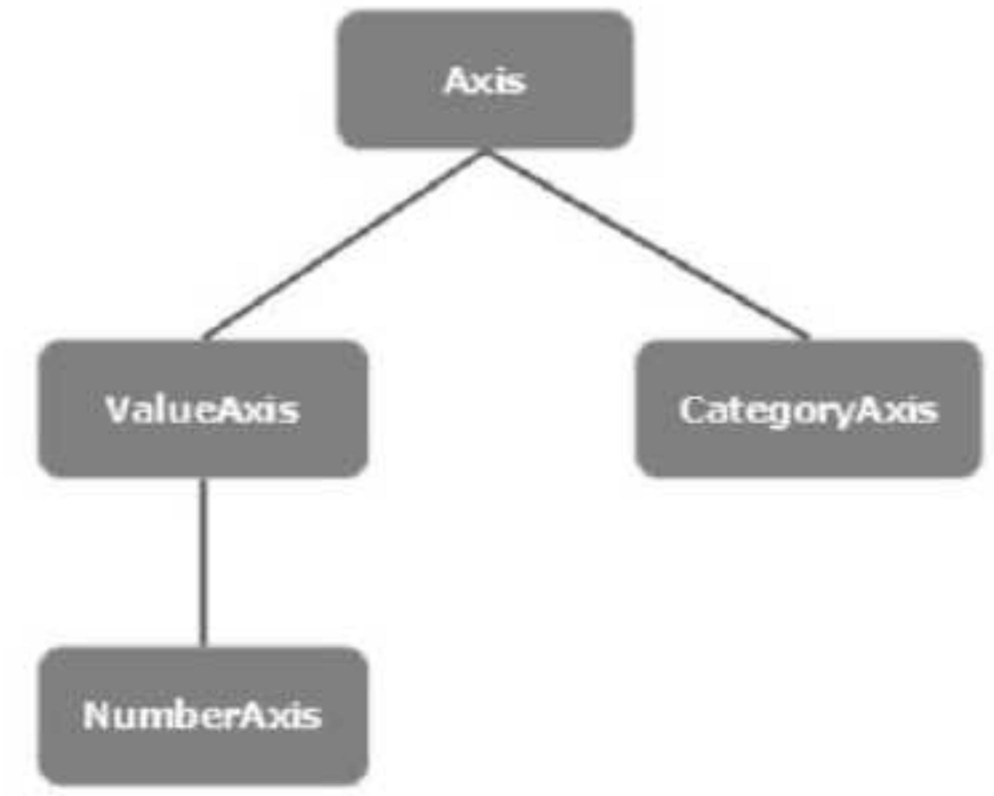


```
NumberAxis xAxis = new NumberAxis();
NumberAxis yAxis = new NumberAxis();
LineChart<Number, Number> lineChart = new LineChart<>(xAxis, yAxis);

lineChart.setTitle("Line Chart");
xAxis.setLabel("Month");
yAxis.setLabel("Sales");

XYChart.Series<Number, Number> series = new XYChart.Series<>();
series.getData().add(new XYChart.Data<>( xValue: 1,  yValue: 23));
series.getData().add(new XYChart.Data<>( xValue: 2,  yValue: 100));
series.getData().add(new XYChart.Data<>( xValue: 3,  yValue: 56));
series.getData().add(new XYChart.Data<>( xValue: 4,  yValue: 90));

Scene scene = new Scene(lineChart,  width: 800,  height: 400);
lineChart.getData().add(series);
stage.setScene(scene);
stage.show();
```



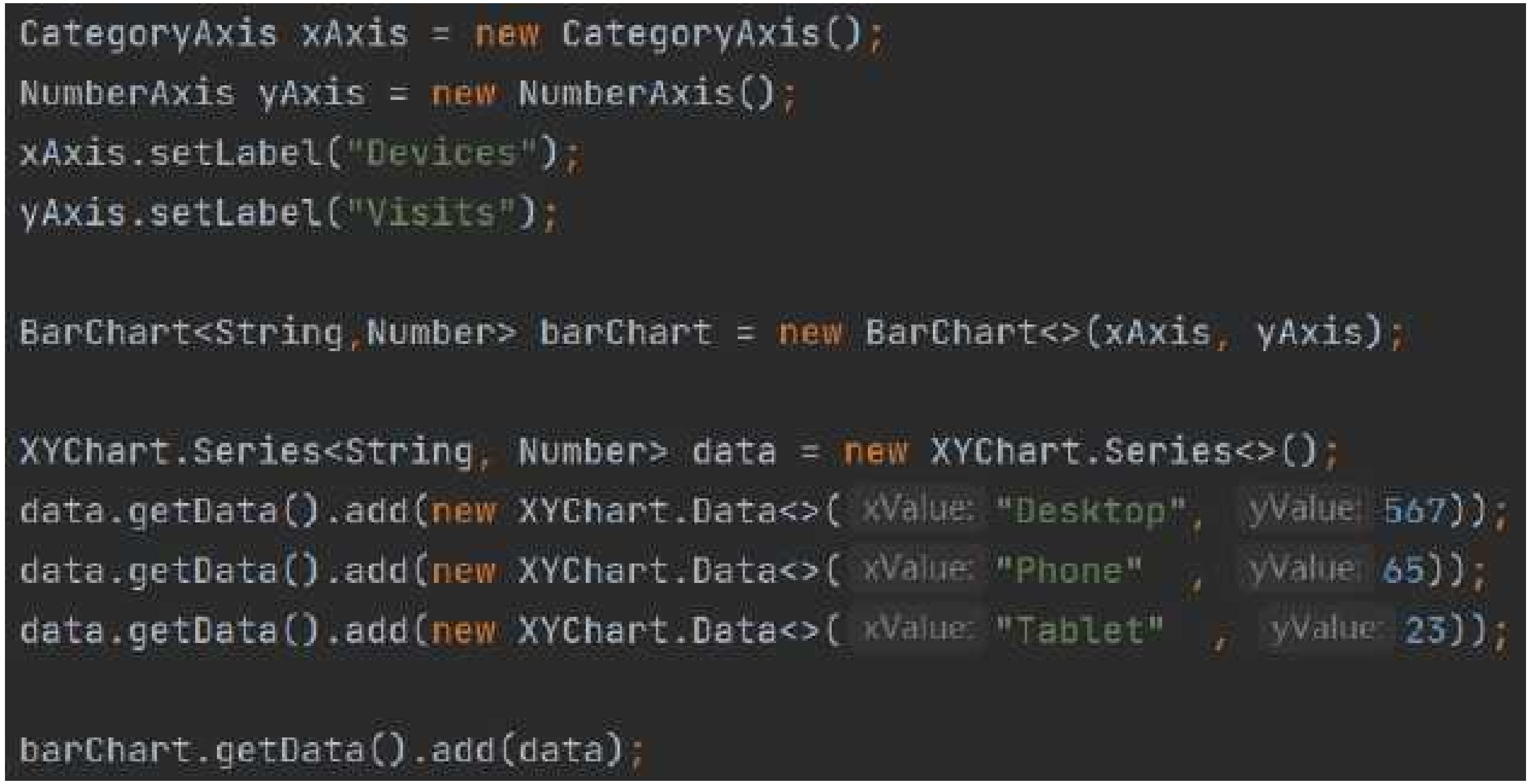
Axis

- An abstract class representing X or Y axis
- NumberAxis
  - Quantity, Age, Population, etc.
- CategoryAxis
  - Countries, Weekdays, Colors, etc.

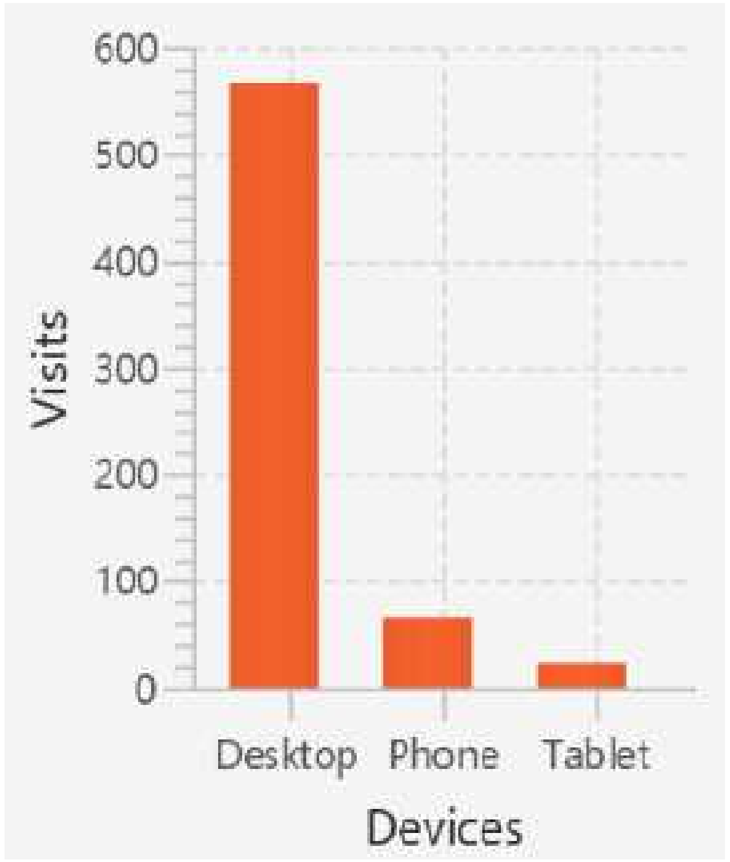


https://www.tutorialspoint.com/javafx/javafx_charts.htm

Using CategoryAxis



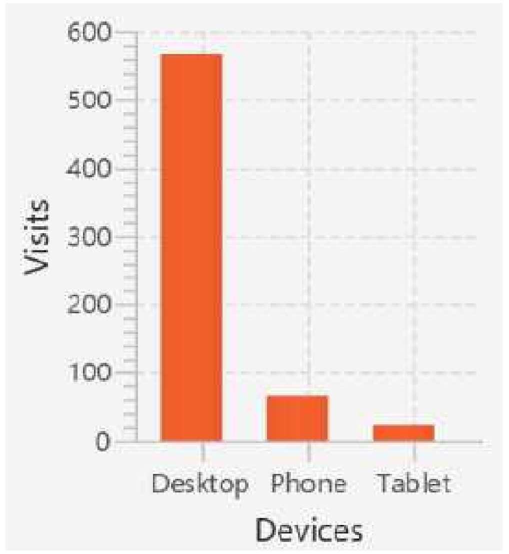
```
CategoryAxis xAxis = new CategoryAxis();
NumberAxis yAxis = new NumberAxis();
xAxis.setLabel("Devices");
yAxis.setLabel("Visits");

BarChart<String,Number> barChart = new BarChart<>(xAxis, yAxis);

XYChart.Series<String, Number> data = new XYChart.Series<>();
data.getData().add(new XYChart.Data<>( xValue: "Desktop",  yValue: 567));
data.getData().add(new XYChart.Data<>( xValue: "Phone"   ,  yValue: 65));
data.getData().add(new XYChart.Data<>( xValue: "Tablet"   ,  yValue: 23));

barChart.getData().add(data);
```



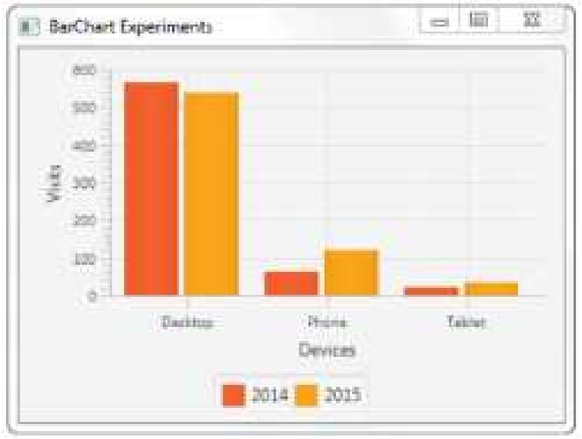
Full example: http://tutorials.jenkov.com/javafx/barchart.html

Series & Data Points

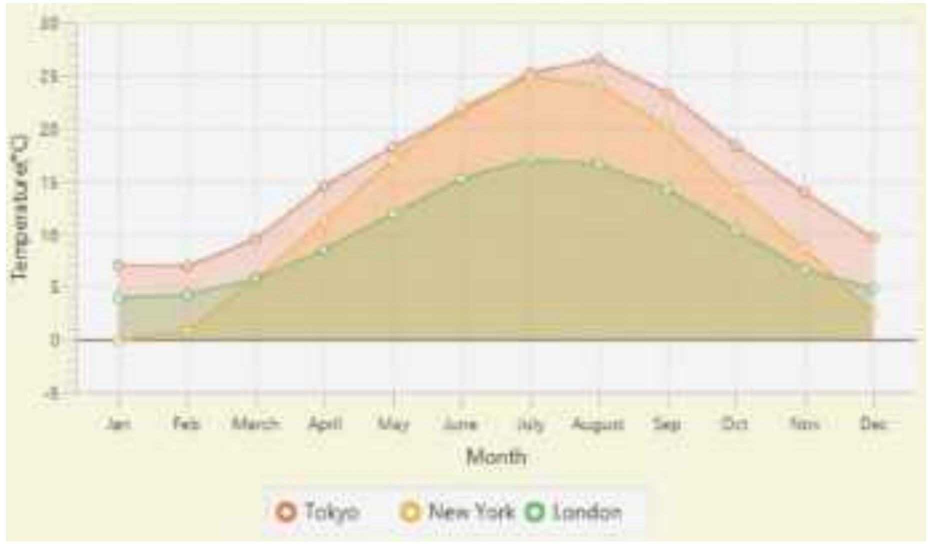
- A **series** consists of zero or more **data points**
- A **chart** may consist **multiple series**



1 series, 3 data points

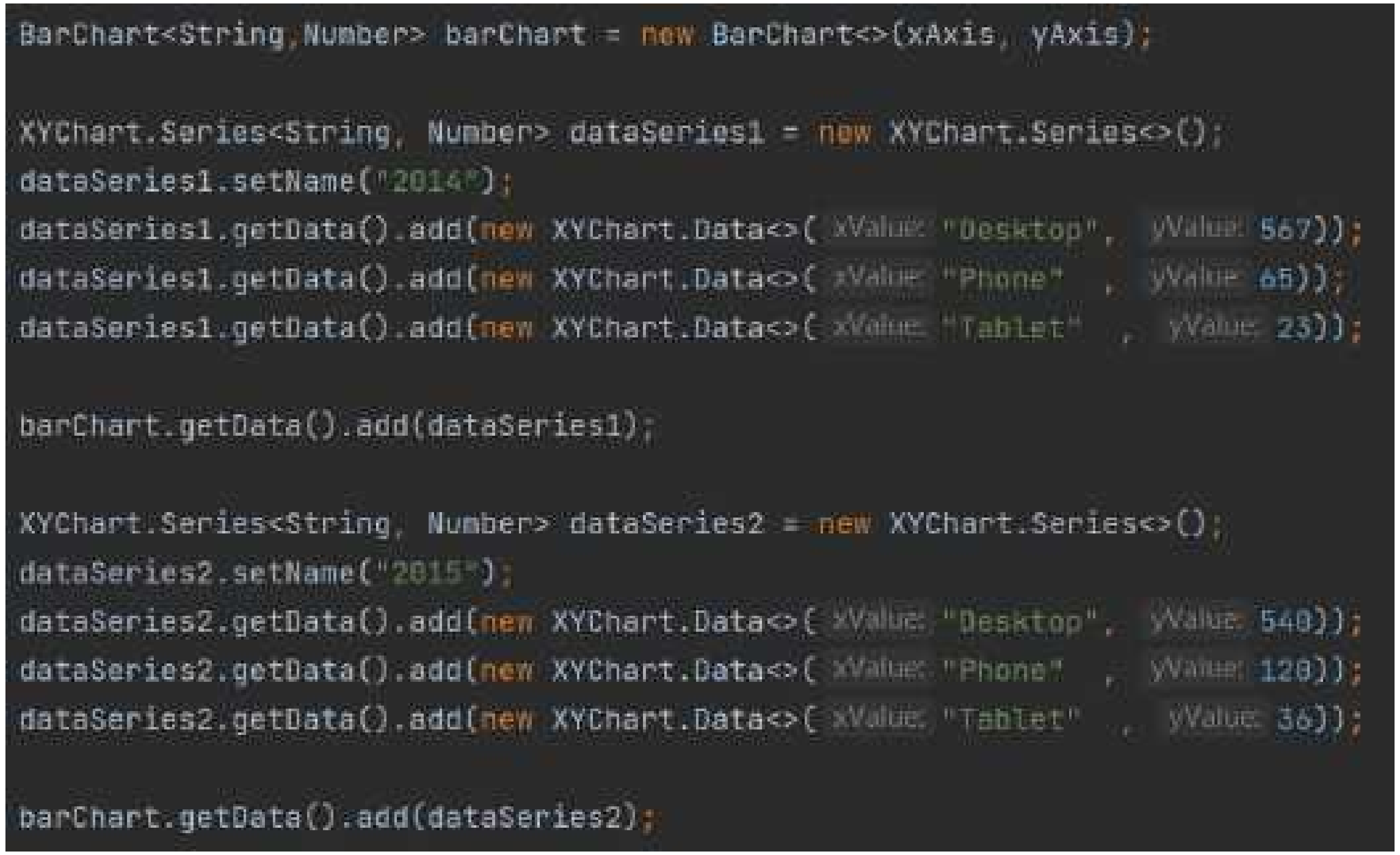


2 series, each has 3 data points



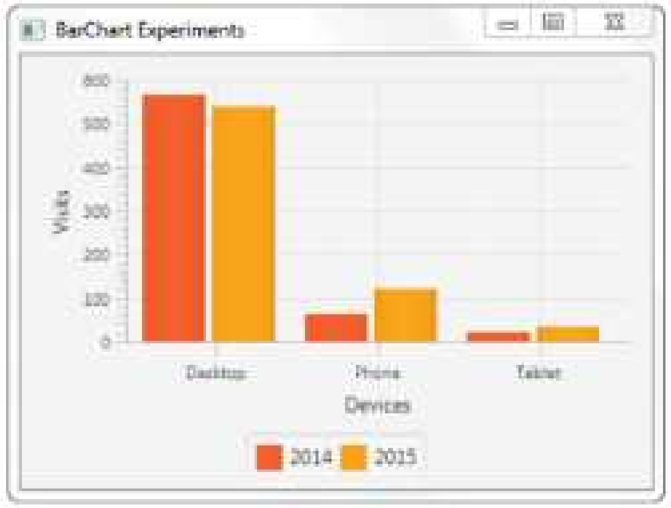
3 series, each has 12 data points

Updating Chart Data

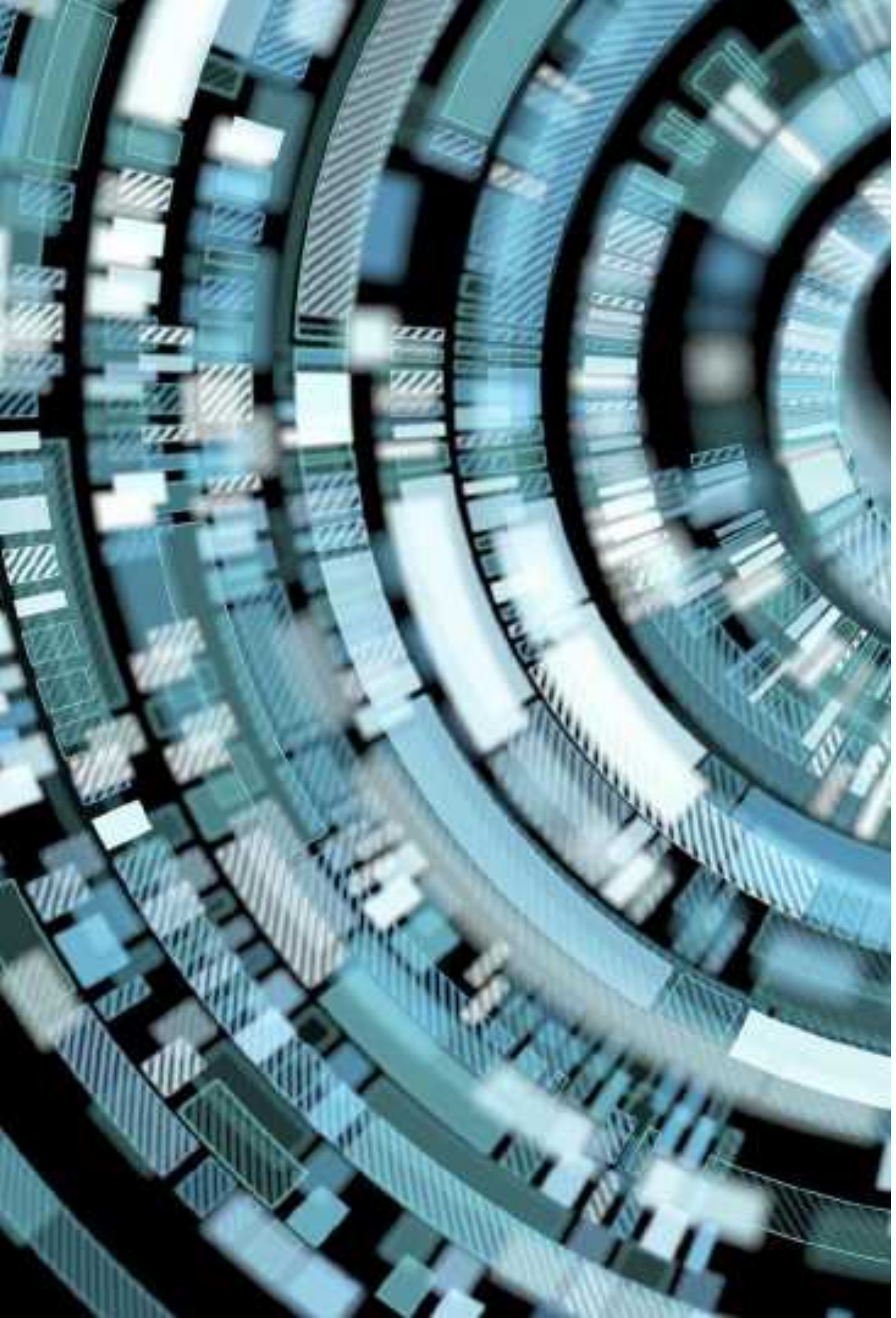


```
BarChart<String,Number> barChart = new BarChart<>(xAxis, yAxis);

XYChart.Series<String, Number> dataSeries1 = new XYChart.Series<>();
dataSeries1.setName("2014");
dataSeries1.getData().add(new XYChart.Data<>( xValue: "Desktop",  yValue: 567));
dataSeries1.getData().add(new XYChart.Data<>( xValue: "Phone"  ,  yValue: 65));
dataSeries1.getData().add(new XYChart.Data<>( xValue: "Tablet"  ,  yValue: 23));

barChart.getData().add(dataSeries1);

XYChart.Series<String, Number> dataSeries2 = new XYChart.Series<>();
dataSeries2.setName("2015");
dataSeries2.getData().add(new XYChart.Data<>( xValue: "Desktop",  yValue: 540));
dataSeries2.getData().add(new XYChart.Data<>( xValue: "Phone"  ,  yValue: 120));
dataSeries2.getData().add(new XYChart.Data<>( xValue: "Tablet"  ,  yValue: 36));

barChart.getData().add(dataSeries2);
```

- Adding/removing a series from a chart
- Adding/removing data points from a specific series



Full example: http://tutorials.jenkov.com/javafx/barchart.html



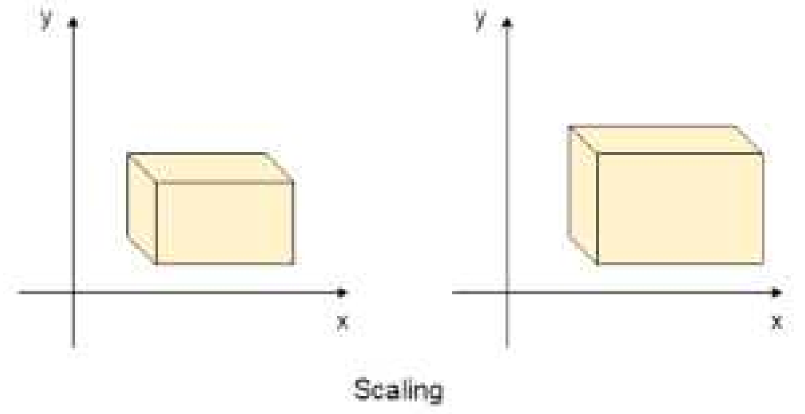
Lecture 5

- Introduction to GUI
- JavaFX
  - Overview
  - Hello World
  - Design & Concepts
  - Layouts, Shapes, UI controls
  - Charts and Axis
  - Transformation, Animation, Effects

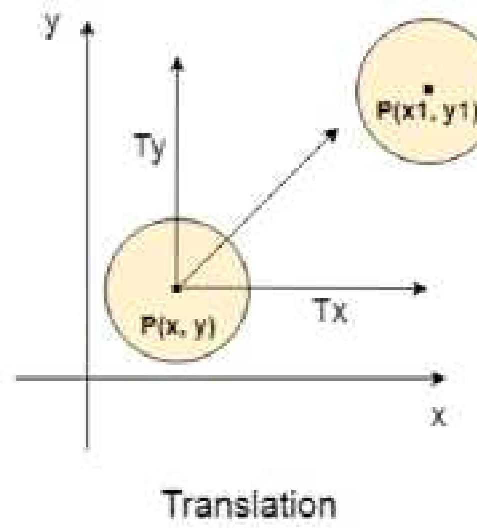
JavaFX Transformation

A transformation changes the place of a graphical object in a coordinate system according to certain parameters.

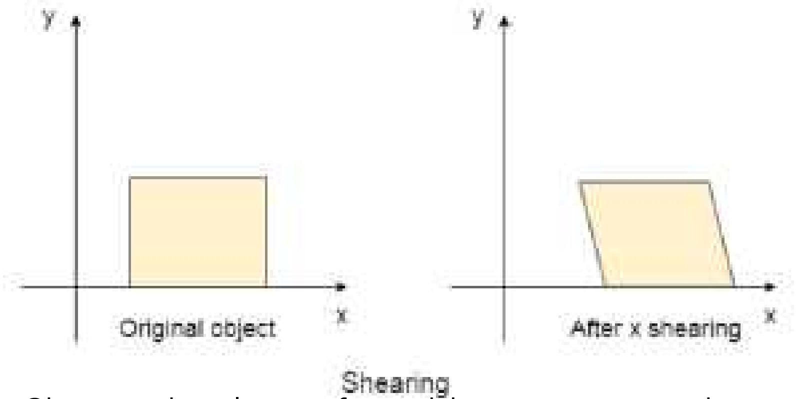
Source: https://www.javatpoint.com/javafx-transformation



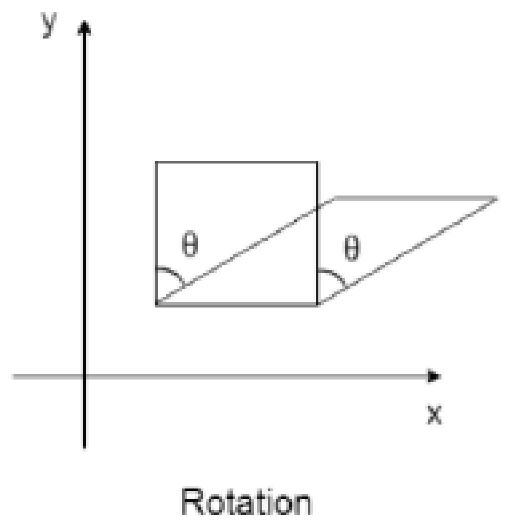
Change the size of an object



Change in the position of an object

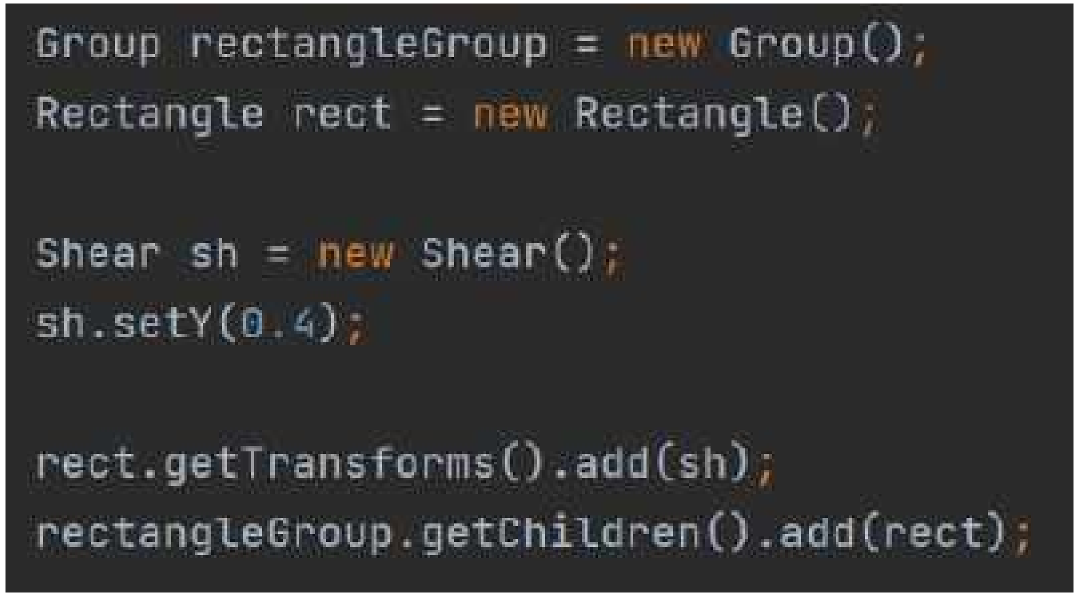


Change the slope of an object w.r.t. any axis

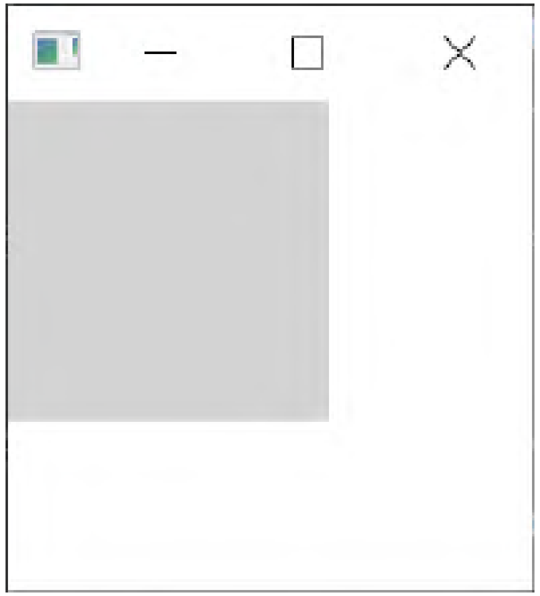


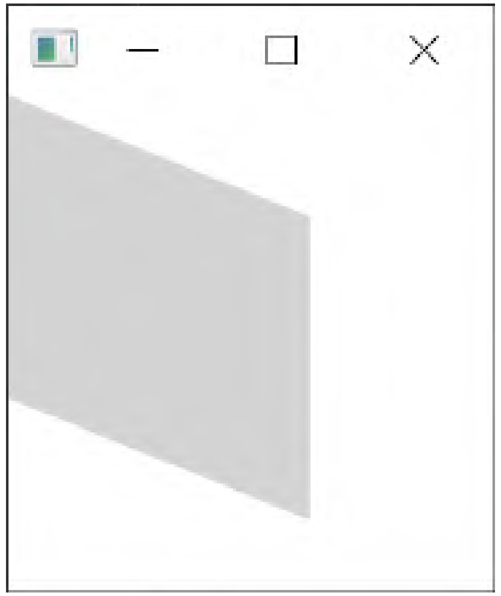
Rotate an object by a certain angle θ

Example

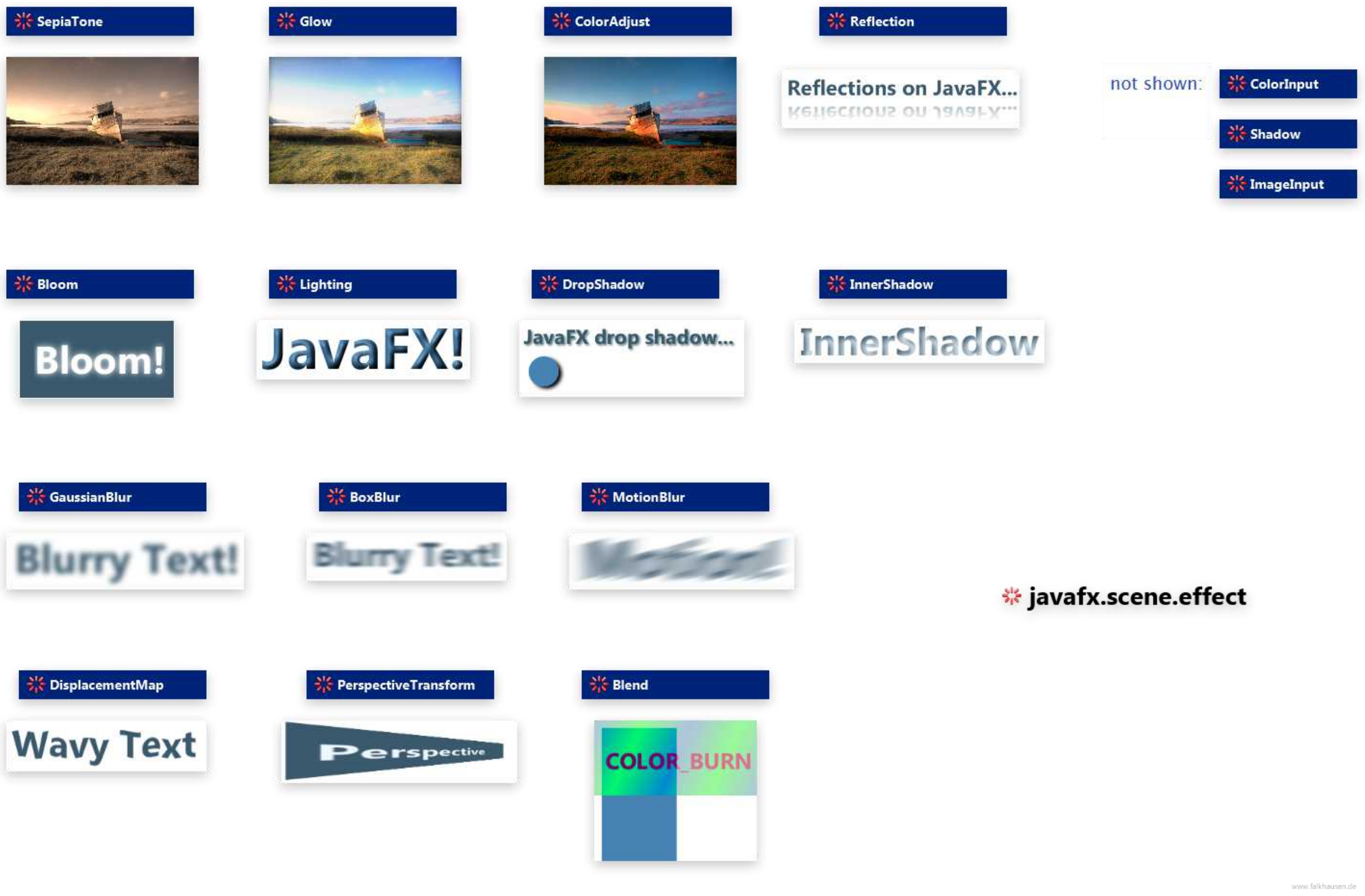


```
Group rectangleGroup = new Group();
Rectangle rect = new Rectangle();

Shear sh = new Shear();
sh.setY(0.4);

rect.getTransforms().add(sh);
rectangleGroup.getChildren().add(rect);
```

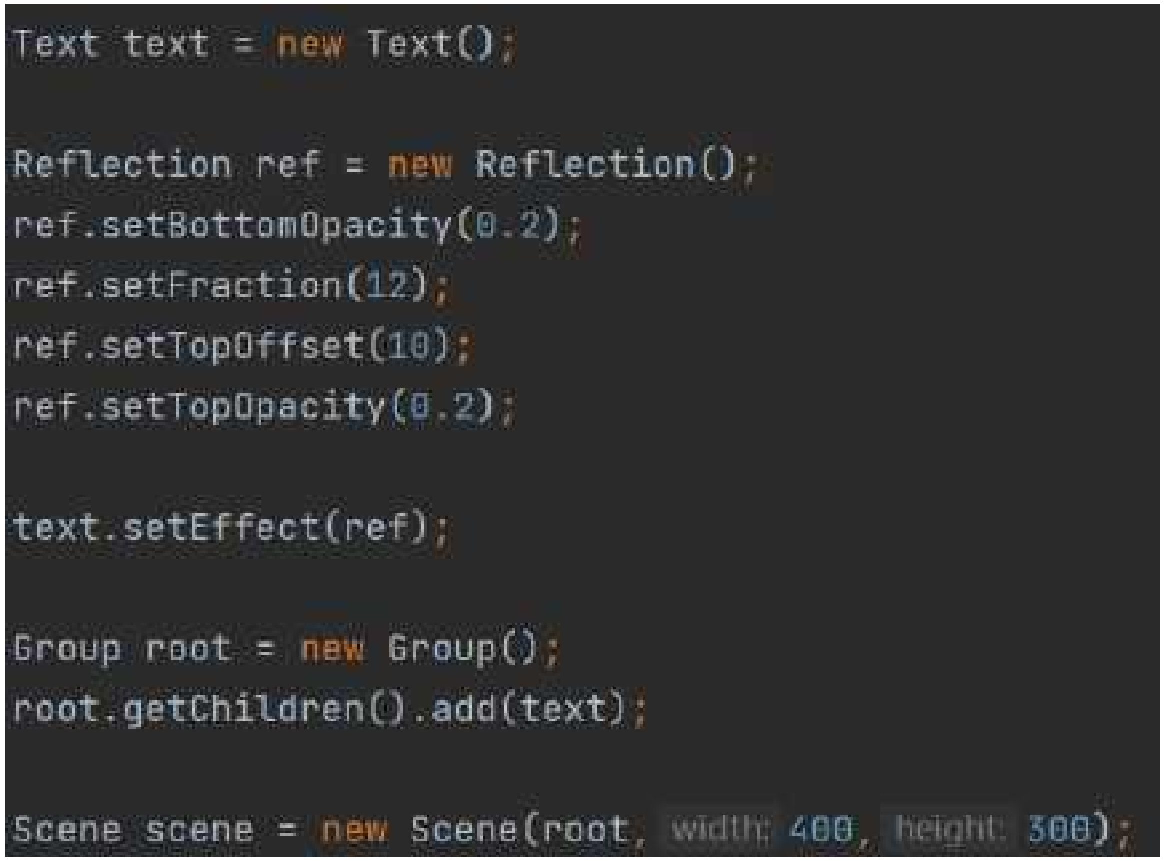




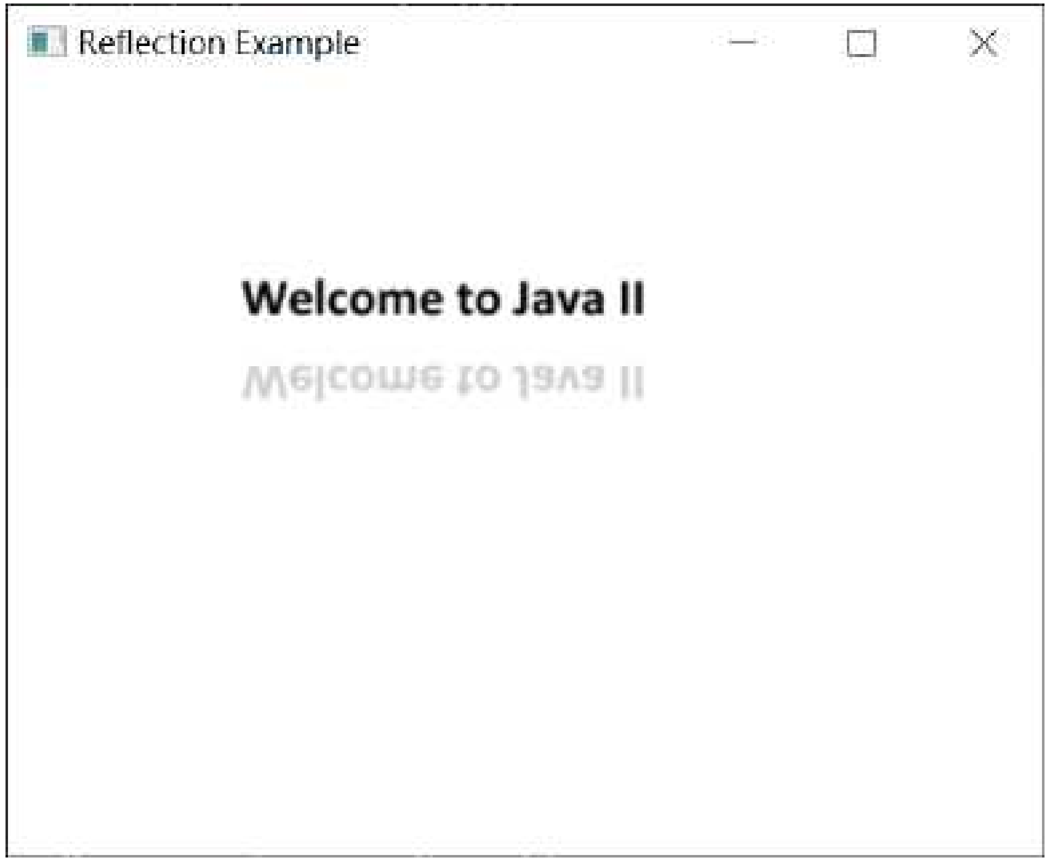
JavaFX Effects



https://www.falkhausen.de/JavaFX-10/scene.effect/Effect-examples.html



```
Text text = new Text();

Reflection ref = new Reflection();
ref.setBottomOpacity(0.2);
ref.setFraction(12);
ref.setTopOffset(10);
ref.setTopOpacity(0.2);

text.setEffect(ref);

Group root = new Group();
root.getChildren().add(text);

Scene scene = new Scene(root, width: 400, height: 300);
```

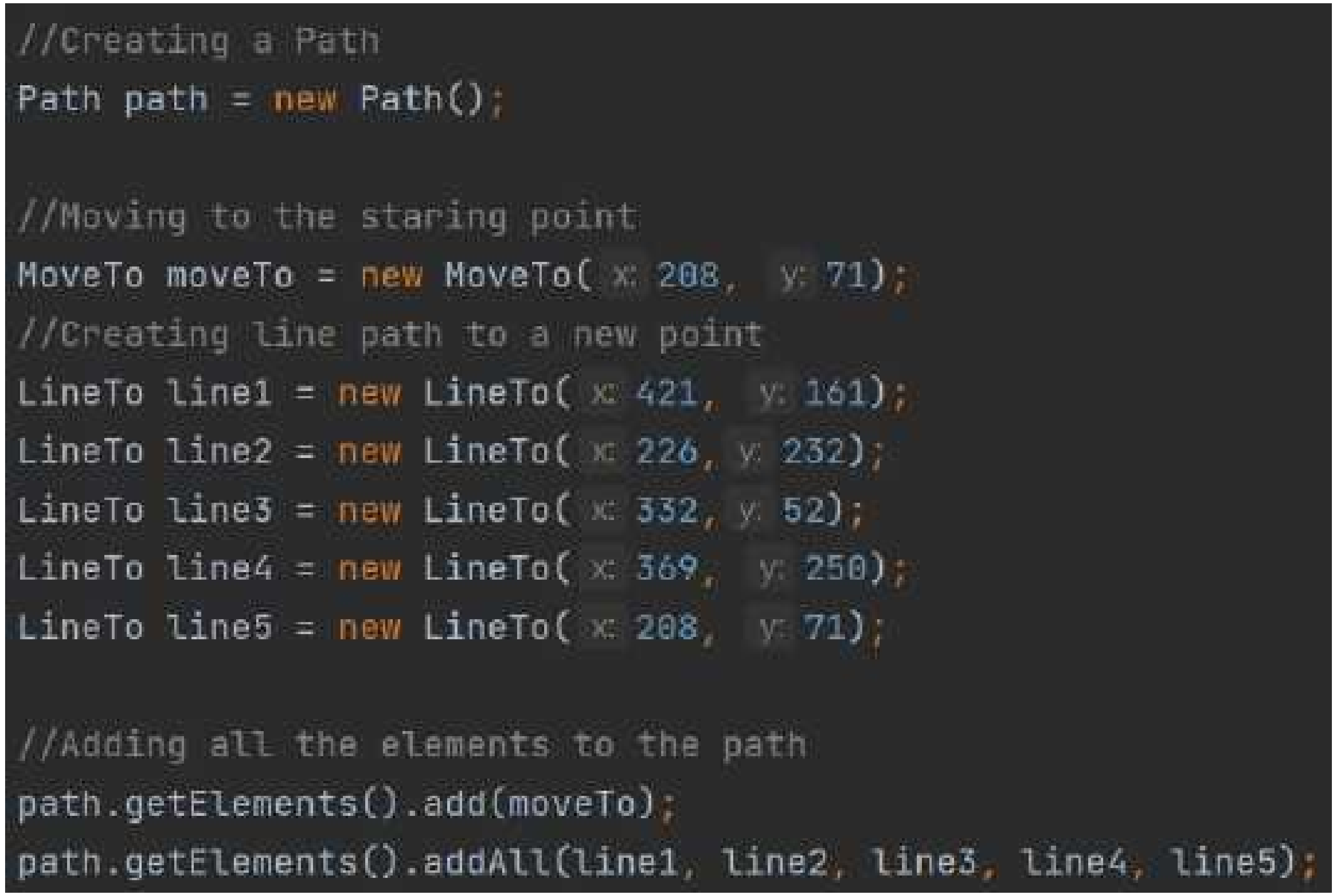


Full example: https://www.javatpoint.com/javafx-reflection-effect

Example: Reflection Effect

JavaFX Animation Example

Creating Path

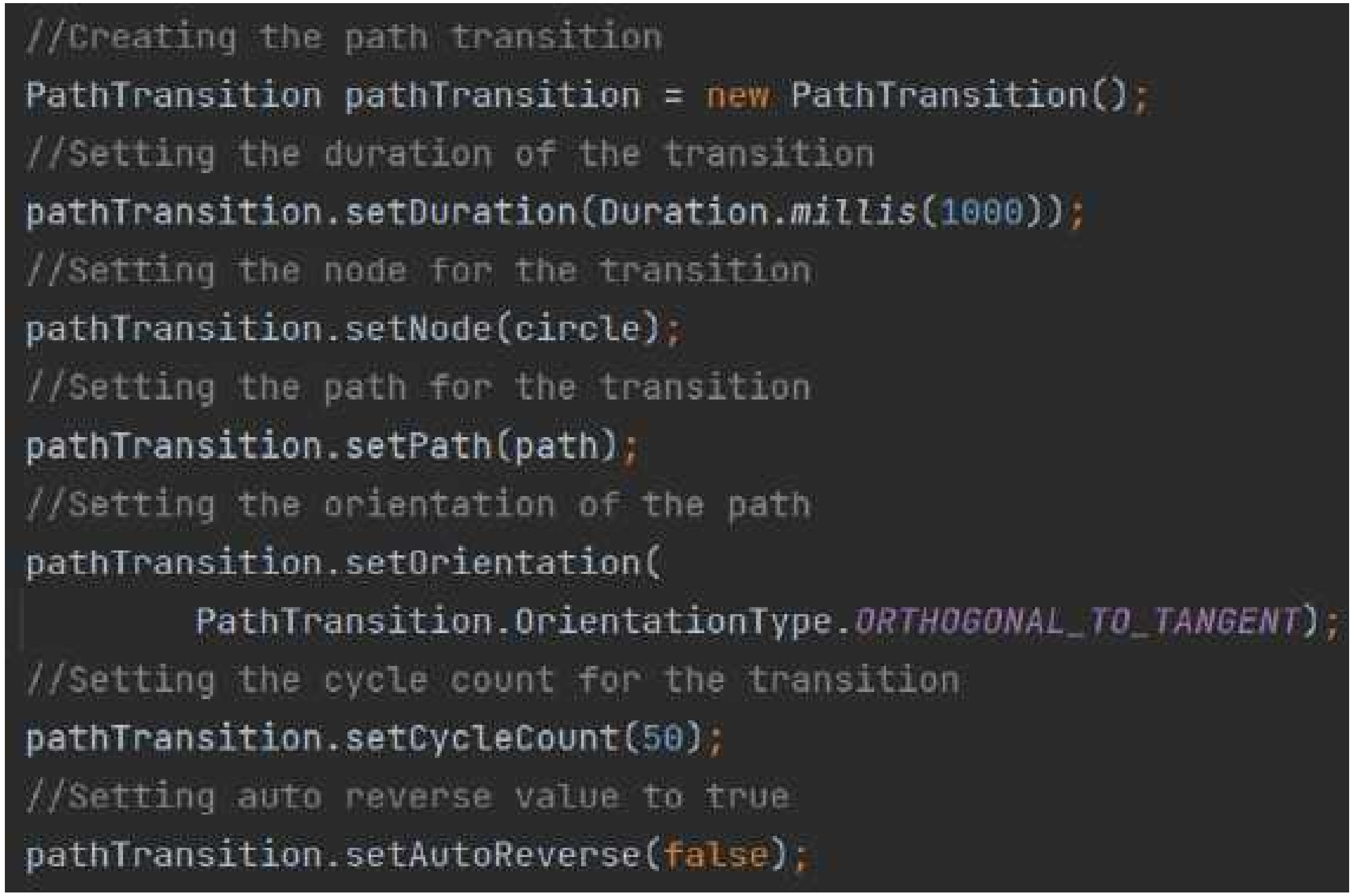


```
//Creating a Path
Path path = new Path();

//Moving to the staring point
MoveTo moveTo = new MoveTo( x: 208,  y: 71);
//Creating line path to a new point
LineTo line1 = new LineTo( x: 421,  y: 161);
LineTo line2 = new LineTo( x: 226, y: 232);
LineTo line3 = new LineTo( x: 332, y: 52);
LineTo line4 = new LineTo( x: 369,  y: 250);
LineTo line5 = new LineTo( x: 208,  y: 71);

//Adding all the elements to the path
path.getElements().add(moveTo);
path.getElements().addAll(line1, line2, line3, line4, line5);
```

Full example:
https://www.tutorialspoint.com/javafx/javafx_event_handling.htm

Creating Path Transition Animation

Allows the node to animate through a specified path over the specified duration

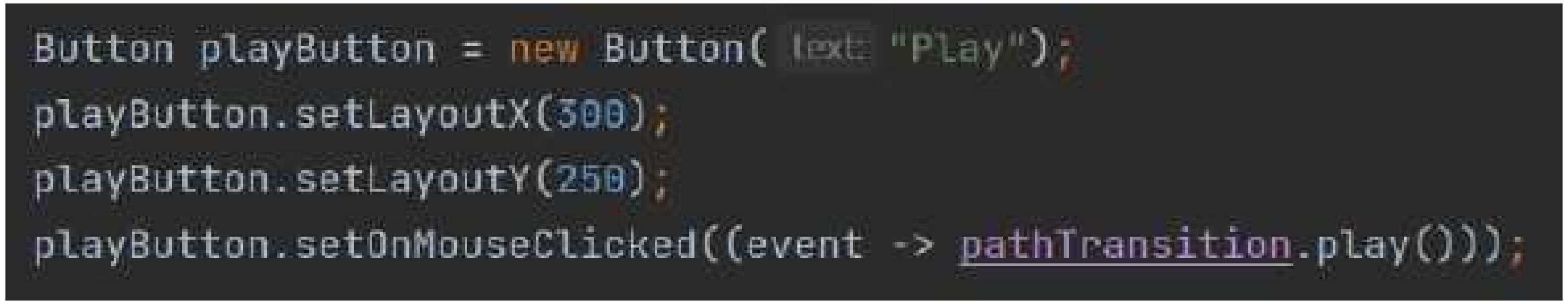


```java
//Creating the path transition
PathTransition pathTransition = new PathTransition();
//Setting the duration of the transition
pathTransition.setDuration(Duration.millis(1000));
//Setting the node for the transition
pathTransition.setNode(circle);
//Setting the path for the transition
pathTransition.setPath(path);
//Setting the orientation of the path
pathTransition.setOrientation(
        PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT);
//Setting the cycle count for the transition
pathTransition.setCycleCount(50);
//Setting auto reverse value to true
pathTransition.setAutoReverse(false);
```

Full example:
https://www.tutorialspoint.com/javafx/javafx_event_handling.htm

Add the Animation Event

When button is clicked, play the animation



```
Button playButton = new Button( text: "Play");
playButton.setLayoutX(300);
playButton.setLayoutY(250);
playButton.setOnMouseClicked((event -> pathTransition.play()));
```

Full example:
https://www.tutorialspoint.com/javafx/javafx_event_handling.htm

Next Lecture

- Design Patterns