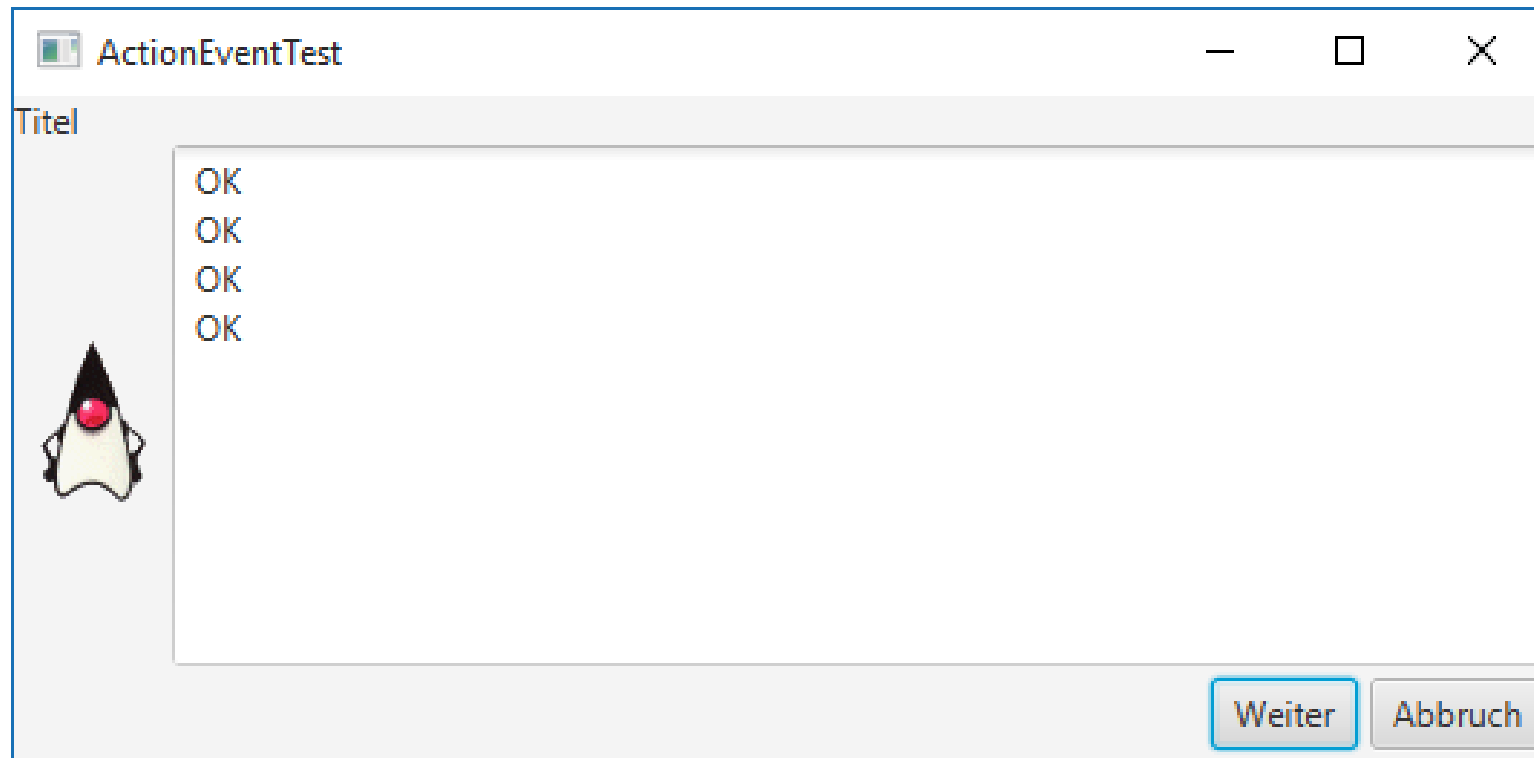


# Grafische Oberflächen und Animationen mit JavaFX



## AWT

Seit dem JDK 1.x gibt es das AWT (Abstract Window Toolkit) zur Erzeugung graphischer Benutzungsoberflächen.

### Nachteile des AWT:

- Alle Fenster- und Dialogelemente werden von dem darunterliegenden Betriebssystem zur Verfügung gestellt → schwierig, plattformübergreifend ein einheitliches Look-and-Feel zu realisieren.
- Die Eigenarten jedes einzelnen Fenstersystems waren für den Anwender unmittelbar zu spüren.
- Im AWT gibt es nur eine Grundmenge an Dialogelementen, mit denen sich aufwändige grafische Benutzeroberflächen nicht oder nur mit sehr viel Zusatzaufwand realisieren ließen.

## Swing

Einführung der Swing-Klassen als Bestandteil der JFC (Java Foundation Classes) mit Java 2.

- Swing-Komponenten benutzen nur Top-Level-Fenster sowie grafische Grafikoperationen des Betriebssystems.
- Alle anderen GUI-Elemente werden von Swing selbst gezeichnet.
- Vorteile:
  - Plattformspezifische Besonderheiten fallen weg → einfachere Implementierung der Dialogelemente.
  - Einheitliche Bedienung auf unterschiedlichen Betriebssystemen
  - Nicht nur Schnittmenge der Komponenten aller Plattformen verfügbar
  - Pluggable Look-and-Feel: Umschaltung des Aussehens einer Anwendung zur Laufzeit (Windows, Motif, Metal, ...).
- Nachteile:
  - Swing-Anwendungen sind ressourcenhungrig. Das Zeichnen der Komponenten erfordert viel CPU-Leistung und Hauptspeicher (Unterstützung durch DirectDraw, OpenGL).

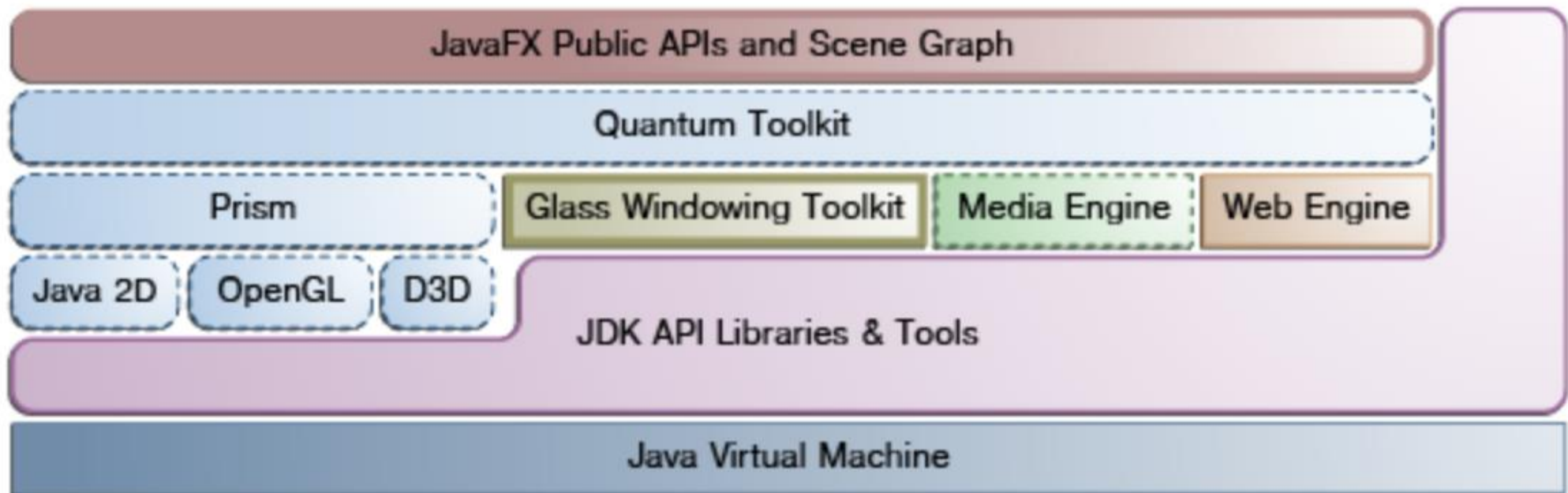
## JavaFX

Wurde ursprünglich eingeführt, um mittels der Skriptsprache JavaFX-Script einfache und schnelle Animationen zu erstellen.

- JavaFX wird Swing ersetzen.
- Die Version 8 bietet noch nicht alle Swing-Funktionen.
- Grafikoperationen usw. sind denen aus Java 2D sehr ähnlich.

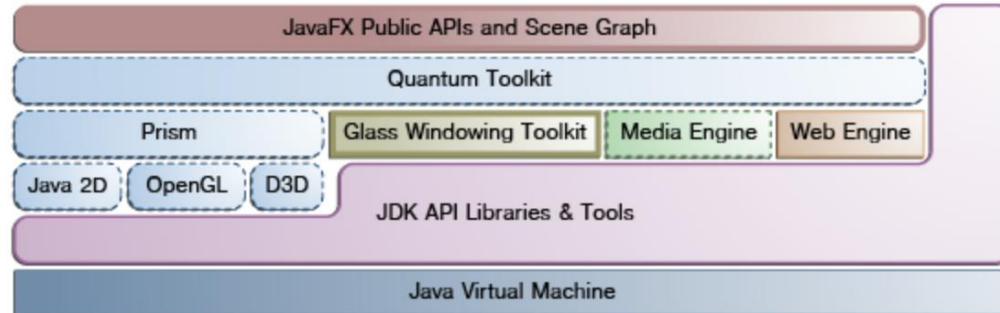
<http://docs.oracle.com/javase/8/javase-clienttechnologies.htm>

# JavaFX Architektur



# JavaFX Architektur

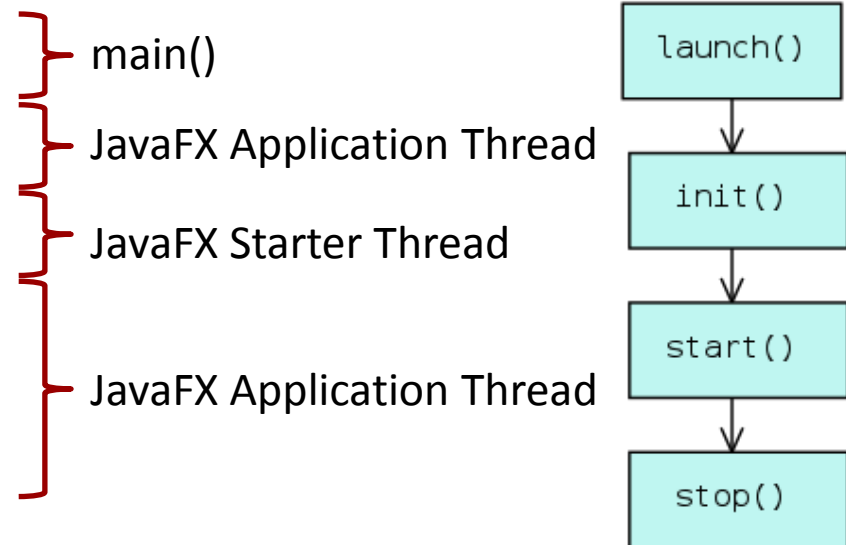
- *Scene Graph:*
  - Baum, der GUI Komponenten abbildet
- *Quantum Toolkit:*
  - Abstraktionsschicht
- *Prism:*
  - Hardware beschleunigtes Rendern der Scene
- *Glass Windowing Toolkit:*
  - Fenster + Timer des OS, Event Queues, User Input, ...
- *Media Engine:*
  - Playback Audio/Video
- *Web Engine:*
  - Web Content (basiert auf WebKit) – inkl. HTML5, CSS3, JS,
- DOM Support



- Jede JavaFX Anwendung ist von *javafx.application.Application* abgeleitet
- JavaFX Runtime erzeugt mehrere Threads, u.a.
  - *JavaFX Launcher*
  - *JavaFX Application Thread*
- Methode *launch()* erzeugt diese beiden Threads

## ■ Ablauf

1. *Application.launch()*
2. *no-args Konstruktor*
3. *init()*
4. *start()*
5. *stop()*





- Lebenszyklus: *launch* => *init* => *start* => *stop*
- Der ursprüngliche Einstiegspunkt für eine Java-Anwendung ist bekanntermaßen die *main*-Methode. Von dieser wird direkt an die **launch-Methode** weitergeleitet und damit beginnt der Lebenszyklus.
- Innerhalb der **init-Methode** können Aufrufparameter der Applikation ausgelesen werden. Diese Methode kann auch leer bleiben. Stage/Scene darf nicht in *init()* erzeugt werden, GUI-Element schon. Wird aufgerufen, wenn Anwendung startet, leere Implementierung in der Basisklasse
- Die **start-Methode** ist der Kernpunkt einer JavaFX-Anwendung. Diese muss überschrieben werden. JavaFX übergibt der Methode ein Objekt vom Typ *Stage* (Bühne). Darunter versteht man eine Art Hauptcontainer.
- Wie lange läuft eine JavaFX-Anwendung? So lange, bis das letzte Fenster der Applikation geschlossen wurde oder die Methode *Platform.exit* aufgerufen wird.
- Die **stop-Methode** wird aufgerufen, wenn die Anwendung beendet wird. Leere Implementierung in der Basisklasse.

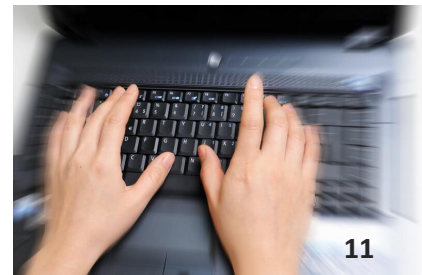
- Der Methode *start* wird ein Parameter der Klasse *Stage* übergeben.
- Innerhalb eines Stages-Objektes können eine oder mehrere *Scene*-Objekte präsentiert werden.
- Die Analogie zu einem Theater ist offensichtlich. Stage ist die Bühne, und auf dieser können eine oder mehrere Szenen spielen.
- Über Eigenschaften kann das Stage-Objekt konfiguriert werden.
- Dazu gehören zum Beispiel die Größe (*setWidth*, *setHeight*) oder der Titel (*setTitle*).

# Stage - anzeigen

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

public class JavaFX01Stage extends Application {
    public static void main(String[] args) {
        Application.launch(args);
    }

    @Override public void start(Stage stage) {
        // Do not write any code here
    }
}
```



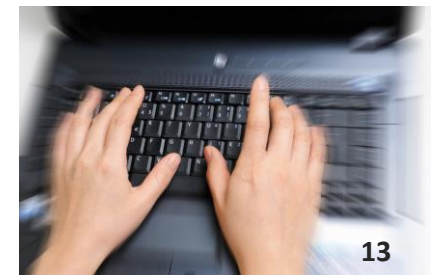
- Starten des Programms:
  - kein Fenster
  - kein Konsolenoutput
  - Taskmanager -> Programm läuft (ewig)
- Warum?
  - JavaFX Programme laufen bis
    - *Platform.exit()* bzw.
    - bis von letzter *stage* Methode *.close()* aufgerufen wird

# Stage anzeigen

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

public class JavaFX02Stage extends Application {
    public static void main(String[] args) {
        Application.launch(args);
    }

    @Override public void start(Stage stage) {
        Platform.exit(); // Exit application
    }
}
```

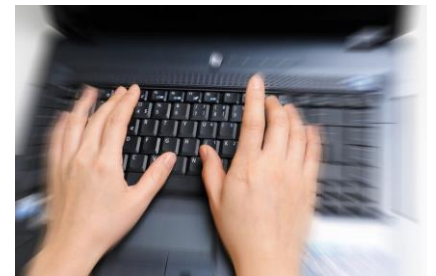


# Stage anzeigen

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

public class JavaFX03Stage extends Application {
    public static void main(String[] args) {
        Application.launch(args);
    }

    @Override public void start(Stage stage) {
        stage.show(); // zeigt stage am Bildschirm an
        stage.close();
    }
}
```



- Bounds
  - Rechteck mit Breite/Höhe
  - Positionsangabe (oberer linker Punkt) mit x/y
- Stage
  - ohne Scene und Größe nicht explizit gesetzt
    - => Position + Größe von Plattform festgesetzt

# Stage - Bounds

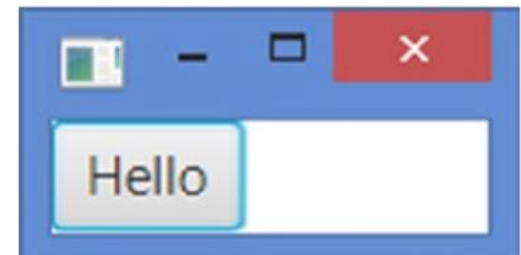
```
import javafx.scene.control.Button;
```

```
public class JavaFX05Stage extends Application {
```

```
...
```

```
@Override
```

```
public void start(Stage stage) {  
    stage.setTitle("Stage mit Button in Scene");  
    Group root = new Group(new Button("Hello"));  
    Scene scene = new Scene(root);  
    stage.setScene(scene);  
    stage.show();  
}
```





# Stage - Bounds

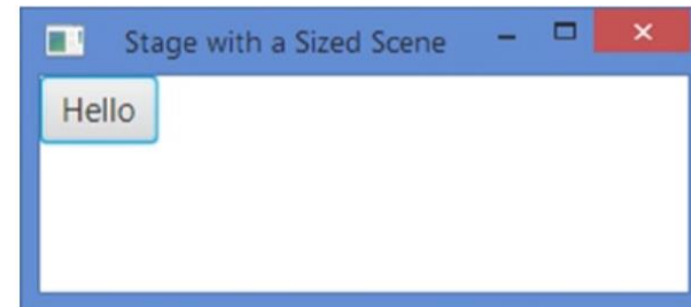
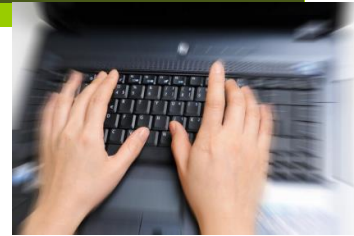
```
import javafx.scene.control.Button;
```

```
public class JavaFX05Stage extends Application {
```

```
...
```

```
@Override
```

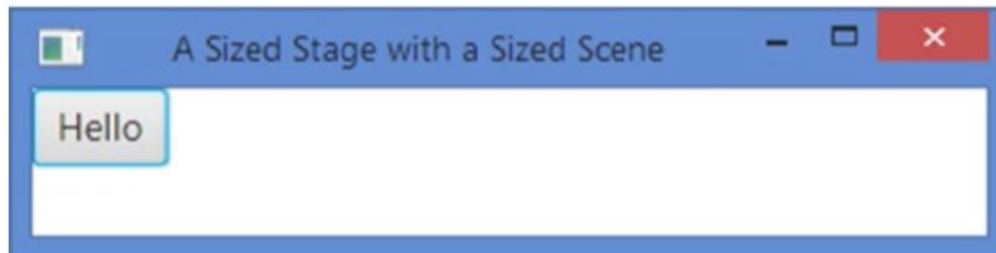
```
public void start(Stage stage) {  
    stage.setTitle("Stage mit sized Scene");  
    Group root = new Group(new Button("Hello"));  
    Scene scene = new Scene(root, 300, 100);  
    stage.setScene(scene);  
    stage.show();  
}
```



# Stage - Bounds

@Override

```
public void start(Stage stage) {  
    stage.setTitle("sized Stage mit sized  
    Scene");  
    Group root = new Group(new Button("Hello"));  
    Scene scene = new Scene(root, 300, 100);  
    stage.setScene(scene);  
    stage.setWidth(400);  
    stage.setHeight(100);  
    stage.show();  
}
```



# Stage - Bounds

- Automatisches Ausrichten

*stage.sizeToScene();*

- Position am Bildschirm

*stage.setX(1000);*

*stage.setY(200);*

- Default:

- stage wird horizontal am Bildschirm zentriert
- y Koordinate (links oben) =  $\frac{1}{3}$  der Bildschirmhöhe – Höhe der Scene

# Stage - Bounds

- Zusammenfassung Stage Bounds
  - keine Scene
    - Bounds von Plattform bestimmt
  - Scene ohne GUI Elemente
    - Bounds von Plattform bestimmt
    - Größe der Scene ist nicht spezifiziert
  - Scene mit GUI Elemente
    - Bounds von GUI Elementen bestimmt
    - Größe der Scene ist nicht spezifiziert
    - Stage am Bildschirm zentriert
  - Scene mit Größenangabe
    - Bounds definiert durch Größenangabe
    - Stage am Bildschirm zentriert

# Stage - Bounds

- Fenster zentrieren am Bildschirm:

```
Rectangle2D bounds = Screen.getPrimary().getVisualBounds();
```

```
double x = bounds.getMinX() +  
           (bounds.getWidth() - stage.getWidth())/2.0;
```

```
double y = bounds.getMinY() +  
           (bounds.getHeight() - stage.getHeight())/2.0;
```

```
stage.setX(x);
```

```
stage.setY(y);
```

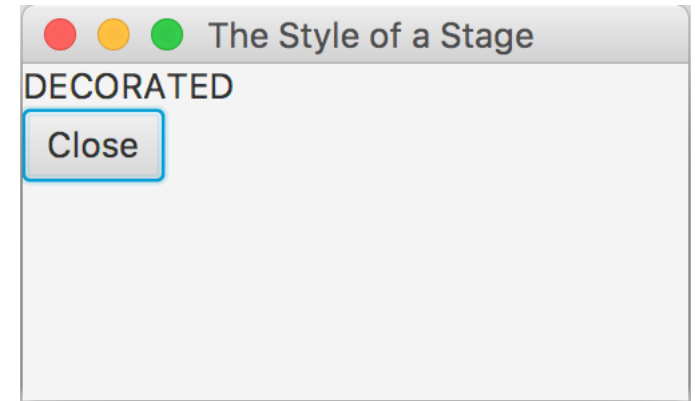
- **ACHTUNG!**

- Darf erst nach *stage.show()* ausgeführt werden
- Vorher hat *stage* noch keine Breite/Höhe!



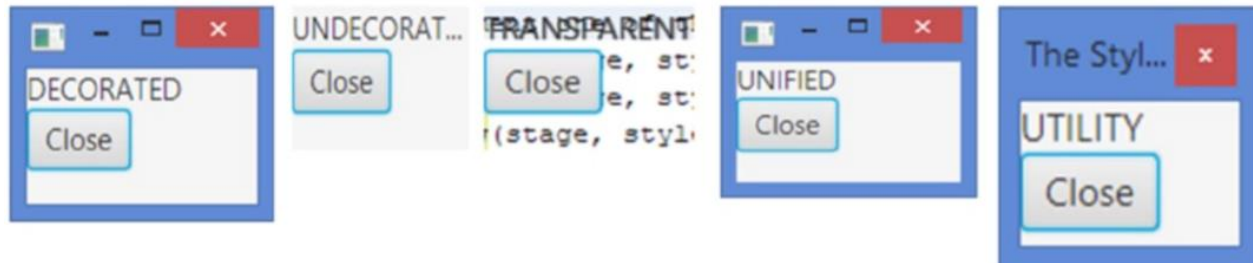
# Stage - Style

- Aufteilung einer Stage:
  - content (Inhalt)
  - decoration (Titel + Rahmen)
- Style einer Stage
  - Decorated (weißer HG + Decorations)
  - Undecorated (weißer HG, keine Decorations)
  - Transparent (Transparenter HG, keine Decorations)
  - Unified (Decorations + kein Rahmen zw. Content und Decorartion)
  - Utility (weißer HG + minimale Decorations)



# Stage – Style

- Style kann mit *initStyle (StageStyle style)* -Klasse Stage geändert werden
  - VOR der .show()
  - Parameter:
    - StageStyle.DECORATED
    - StageStyle.UNDECORATED
    - StageStyle.TRANSPARENT
    - StageStyle.UNIFIED
    - StageStyle.UTILITY



# Stage – Modal / Modeless

- Generell in GUIs – 2 Arten von Fenster
  - Modale – Nicht Modale
    - Modales Fenster: User kann nicht mit anderen Fenstern arbeiten, solange modales Fenster aktiv
    - Nicht Modales Fenster: User kann zwischen Fenster umschalten
  - In JavaFX - 3 Typen (*javafx.stage*):
    - *Modality.NONE*
    - *Modality.WINDOW\_MODAL*
    - *Modality.APPLICATION\_MODAL*

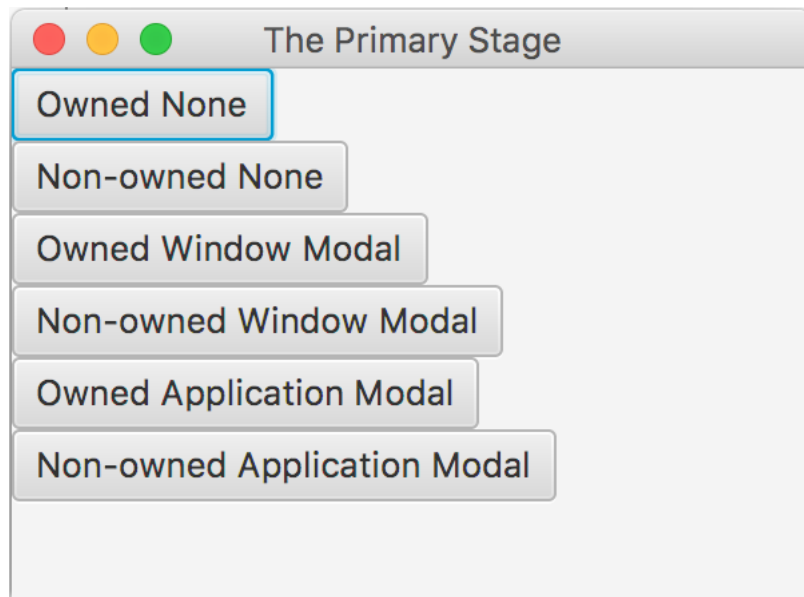


# Stage – Modal/Modeless

- Setzen der Modalität
  - `stage.initModality(Modality.WINDOW_MODAL);`
  - Aufruf VOR `.show()`
- Stage kann Besitzer (Owner) haben
  - wird Fenster minimiert, dann werden auch Fenster minimiert, die “owned” sind
  - `.initOwner (Window owner)`
  - Owner kann auch `NULL` sein
  - Aufruf VOR `.show()`
- *WINDOW\_MODAL: Fenster ist modal für Owner Hierarchie*
- *APPLICATION\_MODAL: Fenster ist modal für ganze Anwendung*

# Stage – Modal / Modeless

- Aufgabe: Programm mit 6 Buttons
  - Je Auswahl soll neue Stage erzeugt werden



# Stage – Modal / Modeless

```
public void start(Stage stage) {  
    ...  
    Button ownedNoneButton = new Button("Owned None");  
    ownedNoneButton.setOnAction(e -> showDialog(stage,  
                                                Modality.NONE));  
    Button nonOwnedNoneButton = ...  
    ...  
  
    VBox root = new VBox();  
    root.getChildren().addAll(ownedNoneButton,  
                             nonOwnedNoneButton,  
                             ownedWinButton,  
                             nonOwnedWinButton,  
                             ownedAppButton,  
                             nonOwnedAppButton);  
    ...  
}
```

# Stage – Modal / Modeless

```
private void showDialog(Window owner, Modality modality) {           // Create a Stage
with specified owner and modality           Stage stage = new Stage();
    stage.initOwner(owner);
    stage.initModality(modality);
    ...
```

# Stage - opacity

- Opacity = Durchsichtigkeit:
  - Werte von 0.0-1.0
  - betrifft auch Decorator
  - *Stage stage = new Stage();*
  - *stage.setOpacity(0.5); // A half-translucent stage*



# Stage - resizing

- Per default darf User stage vergrößern/verkleinern
  - konfigurierbar über
    - `.setResizable(boolean resizable)`
    - `.setMinWidth()`, `.setMinHeight()`
    - `.setMaxWidth()`, `.setMaxHeight()`
  - Scene maximieren:



**`/* Set the position and size of the stage equal to the position and size of the screen */`**

**`Rectangle2D visualBounds = Screen.getPrimary().getVisualBounds();`**

**`stage.setX(visualBounds.getMinX());`**

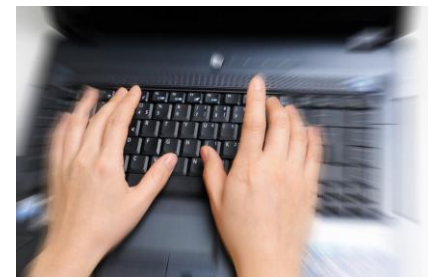
**`stage.setY(visualBounds.getMinY()); stage.setWidth(visualBounds.getWidth());`**

**`stage.setHeight(visualBounds.getHeight());`**

- Konzept von MessageBox bzw. Dialogen gibt es in JavaFX nicht
- *stage.showAndWait();*
  - Programmfluss in aufrufender stage wird erst fortgeführt, sobald (alle) aufgerufenen stages beendet sind

# Stage - Dialoge

- Aufgabe:
  - Primary Stage hat einen Button (open)
  - Klick darauf erzeugt eine neue Stage mit *showAndWait()*, Parameter: *stageNumber*
  - Neue Stage hat 2 Buttons:
    - „Open“: öffnet neue Stage
    - „Hello“: gibt *stageNumber* aus





# Stage - Dialoge

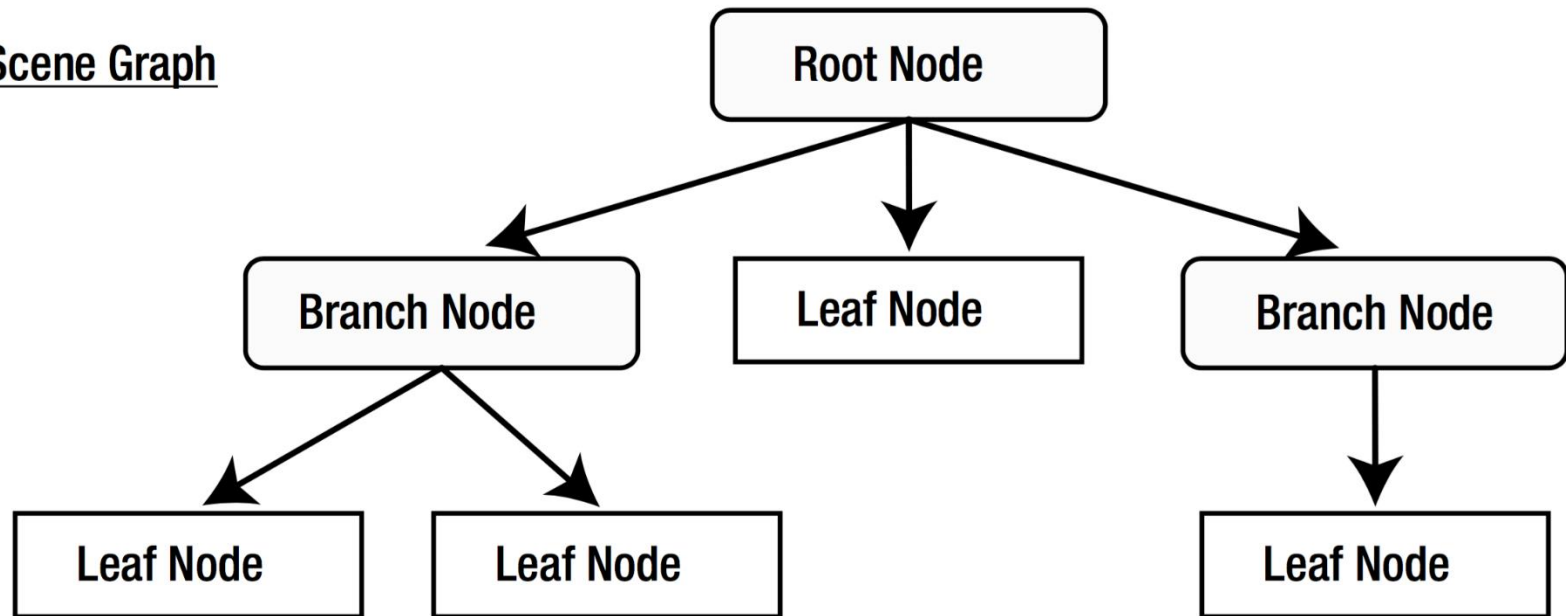
```
public void start(Stage stage) {  
    ...  
    Button openButton = new Button("Open");  
    openButton.setOnAction(e -> open(++counter));  
    ...  
  
    private void open(int stageNumber) {  
        ...  
        Button sayHelloButton = new Button("Say Hello");  
        ...  
        Button openButton = new Button("Open");  
        ...  
        VBox root = new VBox();  
        root.getChildren().addAll(sayHelloButton, openButton);  
    }  
}
```

- Ein Objekt der Klasse *Scene* ist wiederum hierarchisch aufgebaut.
  - Es handelt sich um einen Baum (Tree), der aus einzelnen Knoten besteht (Nodes).
  - Knoten wiederum können so genannte Parent-Nodes enthalten. Diese sind visuell nicht sichtbar (zum Beispiel *BorderPane*, *HBox*, *VBox*) und nehmen weitere Elemente (Kinder) auf.
  - Kind-Elemente sind die typisch sichtbaren Elemente eines UI, wie *Buttons*, *Textfelder* usw.
- Auf diese Weise können mit JavaFX ganze Szenen erstellt werden.
- Eine Szene wird innerhalb der Stage-Klasse zur Anzeige gebracht.
- Auch JavaFX verwendet eine relative Anordnung der Elemente. Wenn Größenänderungen am Fenster erfolgen, erfolgen die Anpassungen in der Szene automatisch.

- Scene Graph:
  - Baumartige Datenstruktur, deren Elemente man Nodes nennt
  - Nodes sind in Eltern-Kind-Beziehung angeordnet
  - Klasse: *javafx.scene.Node*
  - root node: oberster Knoten
  - branch node: kann Kinderknoten haben
  - leaf node: Endknoten

# Scene - SceneGraph

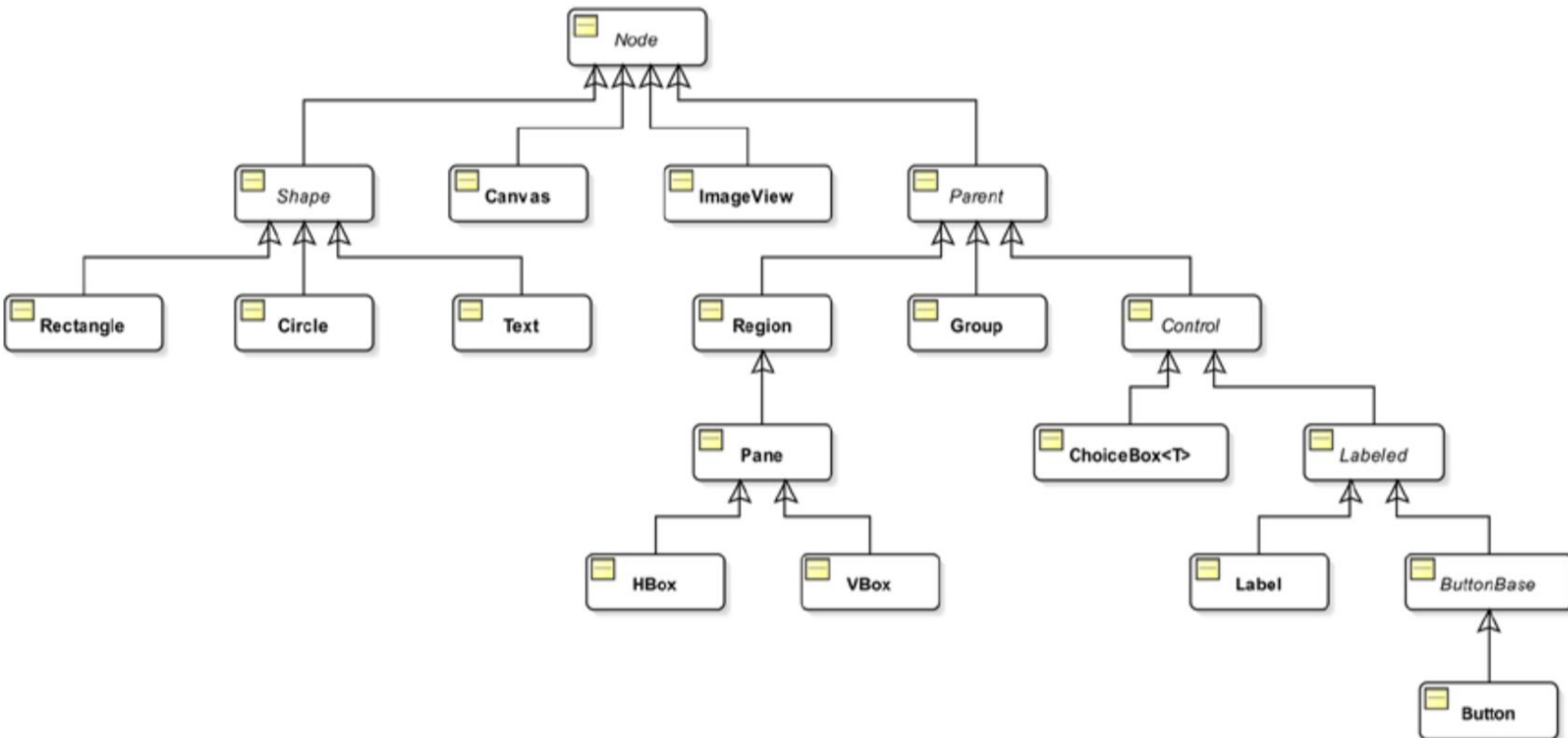
## A Scene Graph



- root node
  - scene hat immer eine root node
  - resizable (z.B. Region oder Control)
    - wird scene vergrößert -> root node wird angepasst
  - non-resizable (z.B. Group)
    - wird scene-Größe geändert -> Inhalt wird an Group angepasst und gegebenenfalls beschnitten
- Branch node: von Parent abgeleitet (abstrakt)
  - muss konkrete Unterklasse, wie z.B. Group, Pane, HBox, VBox, ..., verwenden

- nur EIN node der Scene kann „focus owner“ sein
    - Aber:
      - mehrere Fenster -> mehrere Scenes -> mehrere “focus owner”
      - nur „focus owner“ im aktiven Fenster hat den Fokus
- ```
Node focusOwnerNode = scene.getFocusOwner();  
if (focusOwnerNode == null) {  
    // The scene does not have a focus owner  
} else if (focusOwnerNode.isFocused()) {  
    // The focus owner is the one that has the focus  
} else {  
    // The focus owner does not have the focus  
}
```

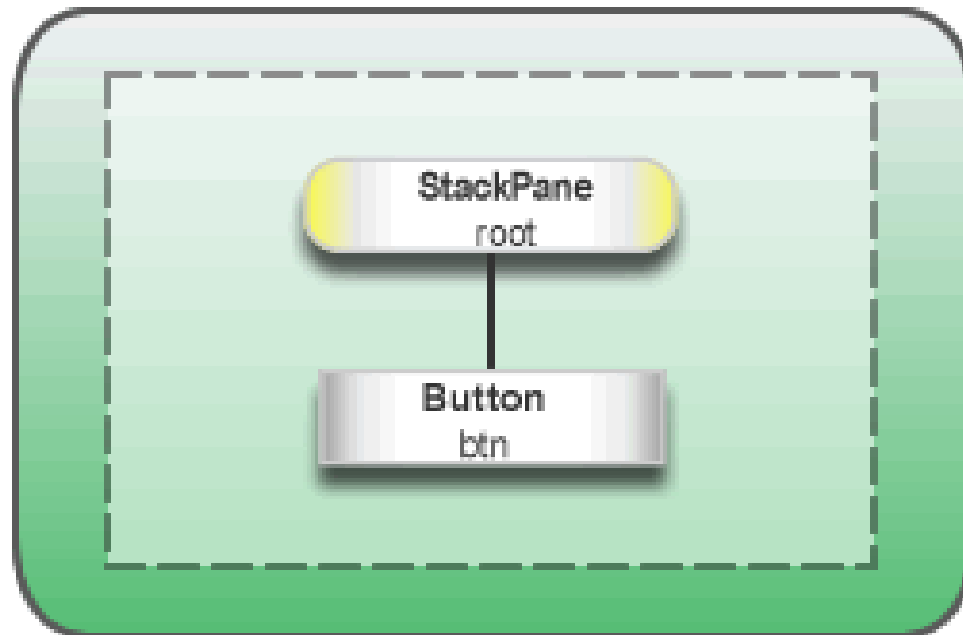
# Scene - Nodes



# Scene - Hierarchie

**Stage** javafx.stage (window)

**Scene** javafx.scene



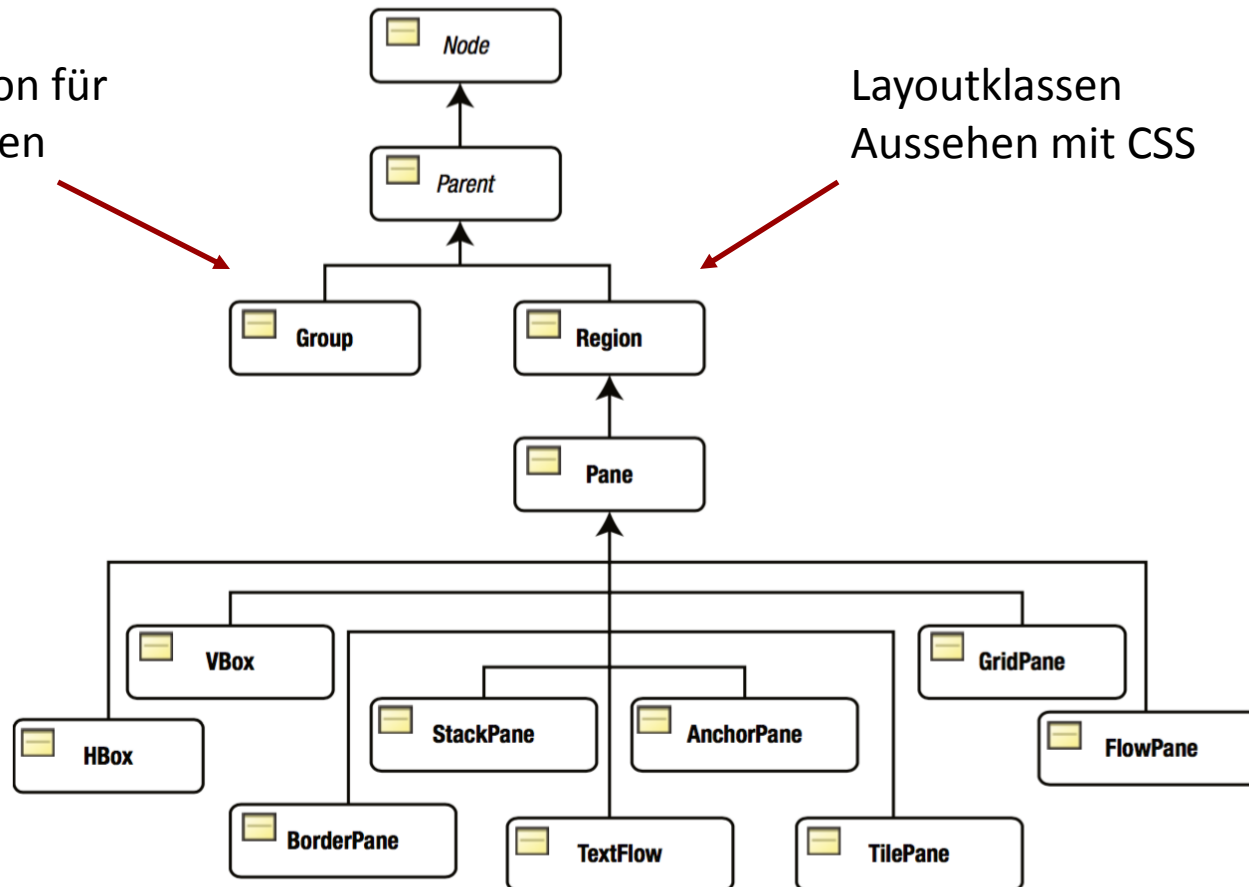


- Unterscheide:
  - Statisches Layout
    - Positionen + Größe der Knoten werden 1x berechnet
    - Bleiben bei Fenstergrößenänderung
  - Dynamisches Layout
    - Position + Größe der Knoten wird laufend berechnet
- Layout Pane:
  - Berechnet Position eines Knotens innerhalb des Elternelements
  - Berechnet Größe des Knotens

# Layout Panes - Klassendiagramm

Effekte +  
Transformation für  
alle Kindknoten

Layoutklassen  
Aussehen mit CSS



# Layout Panes - Überblick

| Container Klasse | Beschreibung                                                                                                                  |
|------------------|-------------------------------------------------------------------------------------------------------------------------------|
| Group            | Effekte und Transformationen an alle Kinder                                                                                   |
| Pane             | Absolutes Positionieren der Kinder                                                                                            |
| HBox             | Horizontales Anordnen in einer Reihe                                                                                          |
| VBox             | Vertikales Anordnen in einer Spalte                                                                                           |
| FlowPane         | Horizontales/Vertikales anordnen in Reihe/Spalte – falls ein Element nicht mehr Platz hat, werden sie automatisch umgebrochen |
| BorderPane       | Layout wird unterteilt in top, right, bottom, left und center                                                                 |
| StackPane        | Stapel – Back-to-Front (übereinander)                                                                                         |
| TilePane         | Raster mit einheitlicher Zellengröße                                                                                          |
| GridPane         | Raster mit variabler Zellengröße                                                                                              |
| AnchorPane       | „Verankern“ der Kanten eines Knoten mit denen des Elternelements                                                              |
| TextFlow         | Rich Text – Inhalt kann aus mehreren Textknoten bestehen                                                                      |

# Layout Panes – Knoten hinzufügen

- Kindknoten einer Layout Pane zuordnen:
  - bei Erzeugung der Pane (Konstruktor)
  - über *.getChildren()* Methode

# Layout Panes

**// Create two buttons**

```
Button okBtn = new Button("OK");  
Button cancelBtn = new Button("Cancel");
```

**// Create an HBox with two buttons as its children**

```
HBox hBox1 = new HBox(okBtn, cancelBtn);
```

**// Create an HBox with two buttons with 20px horizontal spacing  
// between them**

```
double hSpacing = 20;  
HBox hBox2 = new HBox(hSpacing, okBtn, cancelBtn);
```

**// Create an empty HBox, and afterwards, add two buttons to it**

```
HBox hBox3 = new HBox();  
hBox3.getChildren().addAll(okBtn, cancelBtn);
```

# Layout Panes - Group

- Gruppe ist kein richtiger Container, mehr eine Sammlung von Knoten
  - Rendern: in Reihenfolge wie Elemente hinzugefügt werden
  - Kinder werden nicht positioniert (0/0)
  - Kinder werden auf Standardgröße gebracht
  - Gruppe hat keine Größe – sie wird durch die Kinder bestimmt

# Layout Panes - Group

- Erzeugen:

## // Variante 1

```
Group emptyGroup = new Group();  
Button smallBtn = new Button("Small Button");  
Button bigBtn = new Button("This is a big button");  
Group group1 = new Group(smallBtn, bigBtn);
```

## // ODER Variante 2

```
List<Node> initailList = new ArrayList<>();  
initailList.add(smallBtn);  
initailList.add(bigBtn);  
// Create a Group with all Nodes in the initialList as its children  
Group group2 = new Group(initailList);
```

# Layout Panes - Group

- Gruppen werden in der Reihenfolge, in der sie hinzugefügt werden gerendert

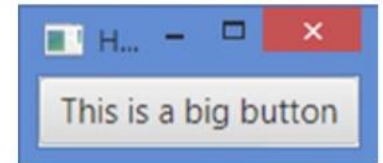
```
Button smallBtn = new Button("Small button");
```

```
Button bigBtn = new Button("This is a big button");
```

```
Group root = new Group();
```

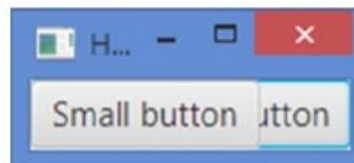
```
root.getChildren().addAll(smallBtn, bigBtn);
```

```
Scene scene = new Scene(root);
```



**// Ändere folgende Zeile:**

```
root.getChildren().addAll(bigBtn, smallBtn);
```





# Layout Panes - Group

- Positionieren in einer Gruppe:

**// Set the location of the OK button**

```
okBtn.setLayoutX(10);
```

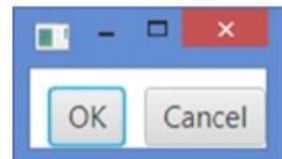
```
okBtn.setLayoutY(10);
```

**// Set the location of the Cancel button relative to the OK button**

```
NumberBinding layoutXBinding = okBtn.layoutXProperty()  
    .add(okBtn.widthProperty().add(10));
```

```
cancelBtn.layoutXProperty().bind(layoutXBinding);
```

```
cancelBtn.layoutYProperty().bind(okBtn.layoutYProperty());
```



# Layout Panes – Panes

- Panes
  - Können verwendet werden, wenn absolute Positionierung benötigt wird
  - Position muss gesetzt werden, sonst (0/0)
  - Größe der Kinder wird auf default Größe gesetzt

# Layout Panes - Panes

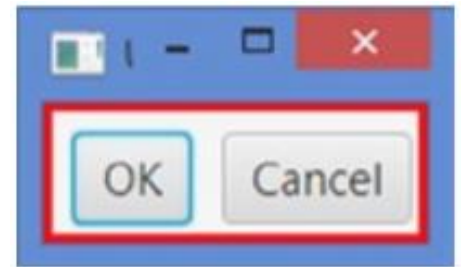
```
Button okBtn = new Button("OK");  
Button cancelBtn = new Button("Cancel");
```

**// Positioniere:**

```
okBtn.relocate(10, 10);  
cancelBtn.relocate(60, 10);  
Pane root = new Pane();  
root.getChildren().addAll(okBtn, cancelBtn);
```

**// Rahmen um Pane herum**

```
root.setStyle("-fx-border-style: solid inside;" +  
              "-fx-border-width: 3;" +  
              "-fx-border-color: red;");
```



# Layout Panes - HBox

- HBox stapelt alle Kinder in einer horizontal ausgerichteten Box
- Standard Abstand: 0px
- Breite passt sich automatisch an
- Position der Kinder kann nicht definiert werden



# Layout Panes - HBox

- Erzeugen einer HBox:

**// leere Hbox mit default spacing (0px)**

```
HBox hbox1 = new HBox();
```

**// Hbox mit 10px spacing**

```
HBox hbox2 = new HBox(10);
```

**// Hbox mit 2 Buttons + 1px spacing**

```
Button okBtn = new Button("OK");
```

```
Button cancelBtn = new Button("Cancel");
```

```
HBox hbox3 = new HBox(10, okBtn, cancelBtn);
```

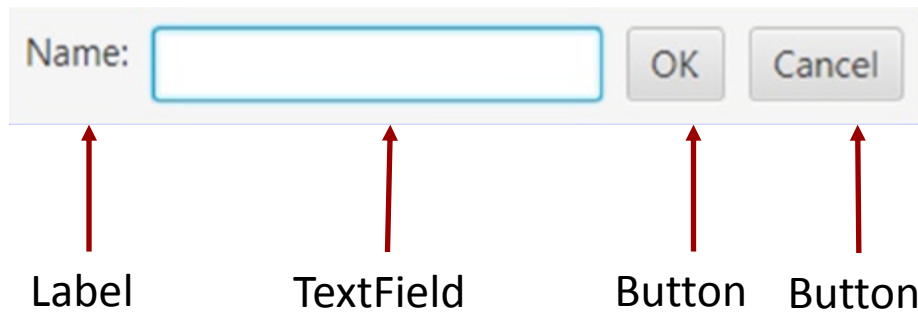
# Layout Panes - HBox

- Hinzufügen von Elementen in bestehende HBox:

```
HBox root = new HBox(10); // 10px spacing  
root.getChildren().addAll(okBtn, ....);
```

# Layout Panes - HBox

- Erstelle folgendes Layout (10px spacing):



- Füge folgende Zeile ein:

```
root.setStyle("-fx-padding: 20;");
```

# Layout Panes - HBox

## ■ HBox Properties:

- `hbox.setAlignment(...)`
  - `Pos.TOP_LEFT` (default)
  - `Pos.BOTTOM_RIGHT`
  - `Pos.BASELINE_CENTER`
  - `Pos.CENTER`
  - ...
- `hbox.setFillHeight(true/false)`
- Ändere vorher die max. Höhe
  - `cancelBtn.setMaxHeight(1000);`
- `.setHgrow()`  
`root.setHgrow(nameFld, Priority.ALWAYS);`

Deaktivieren:

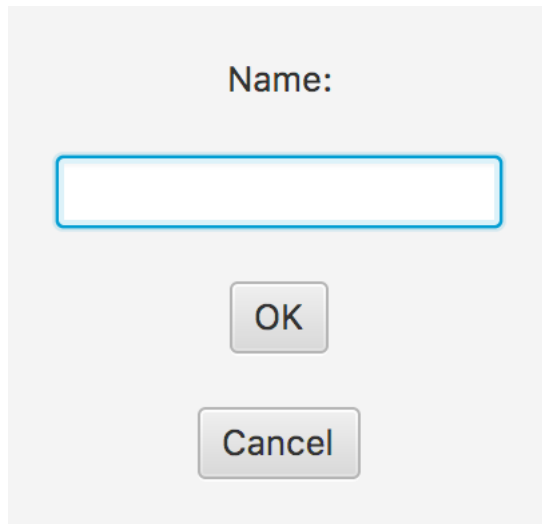
`root.setHgrow(nameFld, null);`





# Layout Panes – VBox

- VBox analog zu HBox nur VERTIKALE Anordnung
- Ändere im vorigen Programm HBox auf VBox



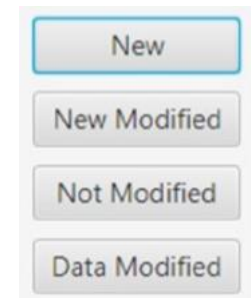
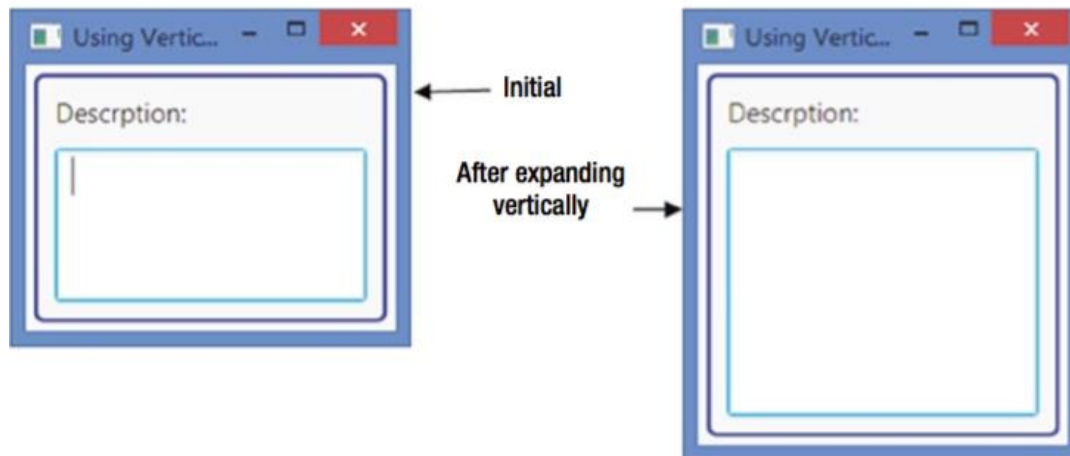
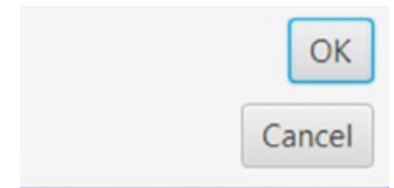
A dialog box with a light gray background. It contains a label 'Name:' at the top. Below the label is a rectangular text input field with a blue border. At the bottom of the dialog are two buttons: 'OK' and 'Cancel', both with a light gray background and a thin border.

VBox



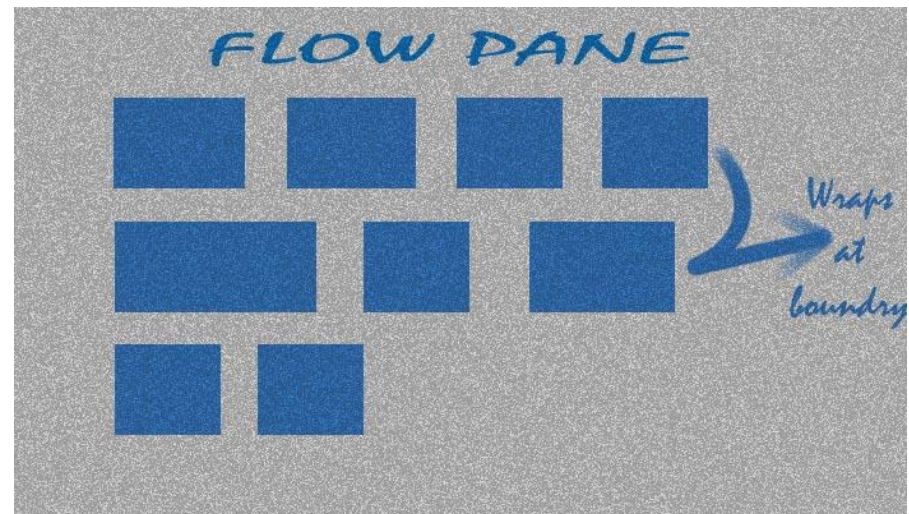
# Layout Panes – VBox

- Properties:
  - `vbox.setAlignment (POS....)`
  - `vbox.setFillWidth(true/false);`
  - `vbox.setVgrow(btn, Priority.ALWAYS);`



# Layout Panes – FlowPane

- FlowPane:
  - Anordnung in Spalten/Zeilen, Umbruch bei definierter Höhe/Breite
  - (vgl. Text in Paragraph/HTML)



# Layout Panes – FlowPane

- Erzeugung

**// Create an empty horizontal FlowPane with 0px spacing**

**FlowPane fpane1 = new FlowPane();**

**// Create an empty vertical FlowPane with 0px spacing**

**FlowPane fpane2 = new FlowPane(Orientation.VERTICAL);**

**// Create an empty horizontal FlowPane with 5px hor. and 10px vertical spacing**

**FlowPane fpane3 = new FlowPane(5, 10);**

**// Create an empty vertical FlowPane with 5px hor. and 10px vertical spacing**

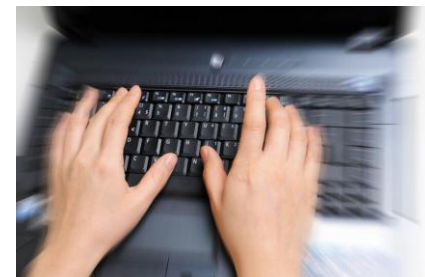
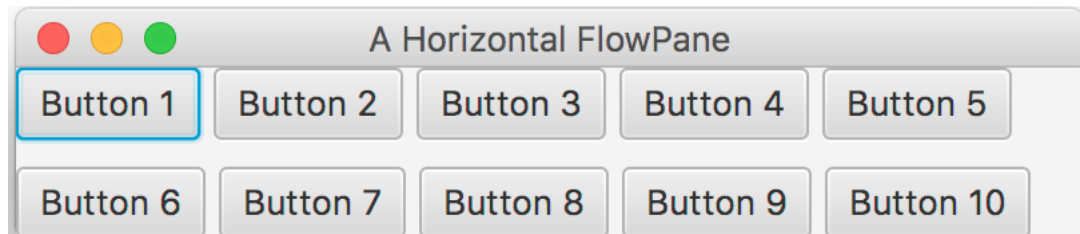
**FlowPane fpane4 = new FlowPane(Orientation.VERTICAL, 5, 10);**

**// Create a horizontal FlowPane with two Buttons and 0px spacing**

**FlowPane fpane5 = new FlowPane(new Button("Button 1"), new Button("Button 2"));**

# Layout Panes – FlowPane

- FlowPane:
  - 10 Buttons erzeugen (Schleife) mit Bezeichnung “Button1 - 10”

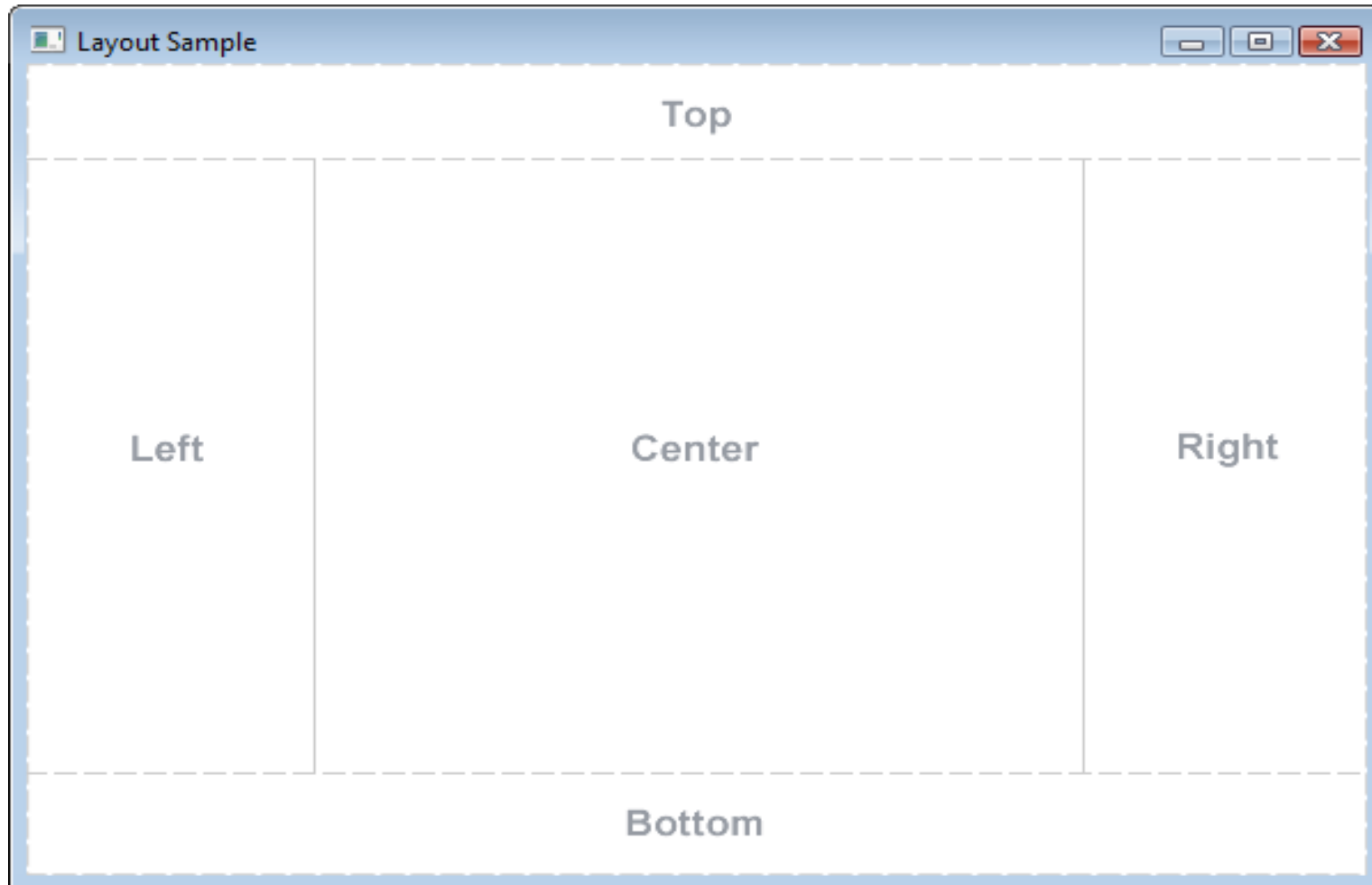


# Layout Panes – FlowPane

- Properties

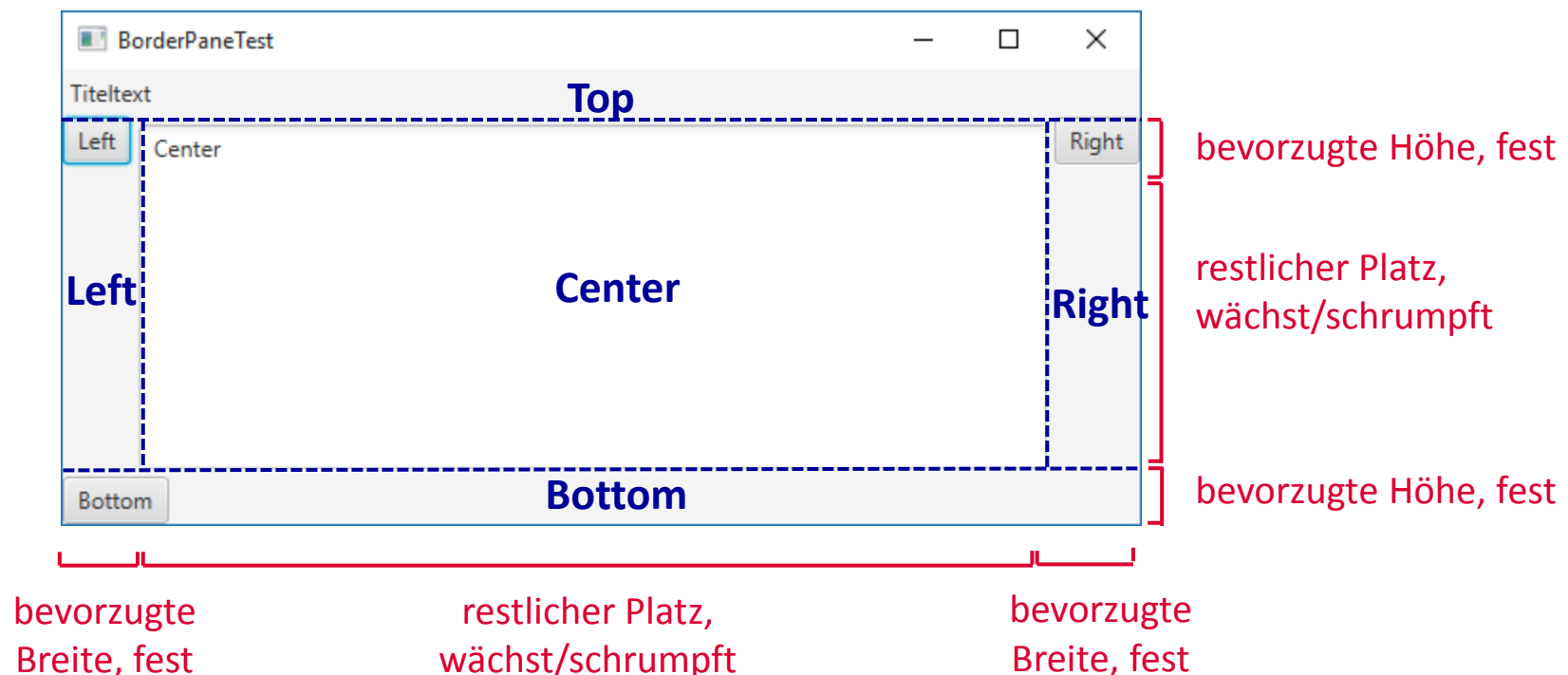
| Property         | Beschreibung                                                                                            |
|------------------|---------------------------------------------------------------------------------------------------------|
| alignment        | Ausrichtung der Spalten/Reihen (default: Pos.TOP_LEFT)                                                  |
| rowValignment    | Vertikale Ausrichtung in horizontaler FlowPane                                                          |
| columnHalignment | Horizontale Ausrichtung in vertikaler FlowPane                                                          |
| hgap,vgap        | Horizontale/vertikale Spalte zwischen Elementen                                                         |
| orientation      | Ausrichtung der FlowPane: HORIZONTAL (default)/VERTICAL                                                 |
| prefWrapLength   | Breite/Höhe, wann umgebrochen werden soll (default: 400)<br>wird benötigt um Breite der FP zu berechnen |

# Layout Panes - BorderPane



### BorderPane: Basis für Standarddialoge

- **BorderPane** besitzt 5 Bereiche, in die jeweils ein Widget eingetragen werden kann:





- Einfügen von Widgets in eine **BorderPane**:
  - **setTop(Control contr)**:
    - **contr**: einzufügendes Widget (z.B. **Label**, **Button**, ...)
  - Statt **setTop** gibt es auch die Varianten für die anderen Positionen (wie z.B. **setBottom**).
- Ist der Bereich größer als der Platz, den das Widget benötigt, dann kann die Position des Widgets festgelegt werden:
  - Taste soll zentriert werden: **BorderPane.setAlignment(button, Pos.CENTER);**
  - Die weiteren Positionsangaben sind ebenfalls statische Attribute der Klasse **Pos**.
- Es kann auch ein leerer Rand um ein Widget erzeugt werden, Beispiel:  
**BorderPane.setMargin(button, new Insets(4.0, 4.0, 4.0, 4.0));**
- **Insets** nimmt die vier Abstände im Uhrzeigersinn auf (oben, rechts, unten, links).

- Beispiel (Datei **BorderPaneTest.java**):

...

```
// Inhalt des Fensters anordnen
BorderPane borderPane = new BorderPane();

// Widgets zur BorderPane hinzufügen
Label topLabel = new Label("Titeltext");
BorderPane.setAlignment(topLabel, Pos.CENTER_LEFT);
BorderPane.setMargin(topLabel, new Insets(4.0, 4.0, 4.0, 4.0));
borderPane.setTop(topLabel);

borderPane.setLeft(new Button("Left"));
borderPane.setRight(new Button("Right"));

TextArea centerText = new TextArea("Center");
BorderPane.setAlignment(centerText, Pos.CENTER);
BorderPane.setMargin(centerText, new Insets(4.0, 4.0, 4.0, 4.0));
borderPane.setCenter(centerText);

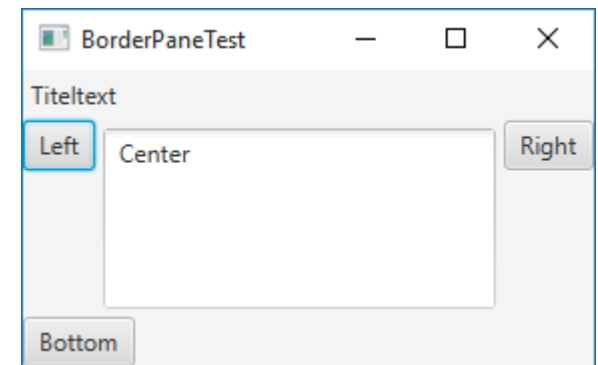
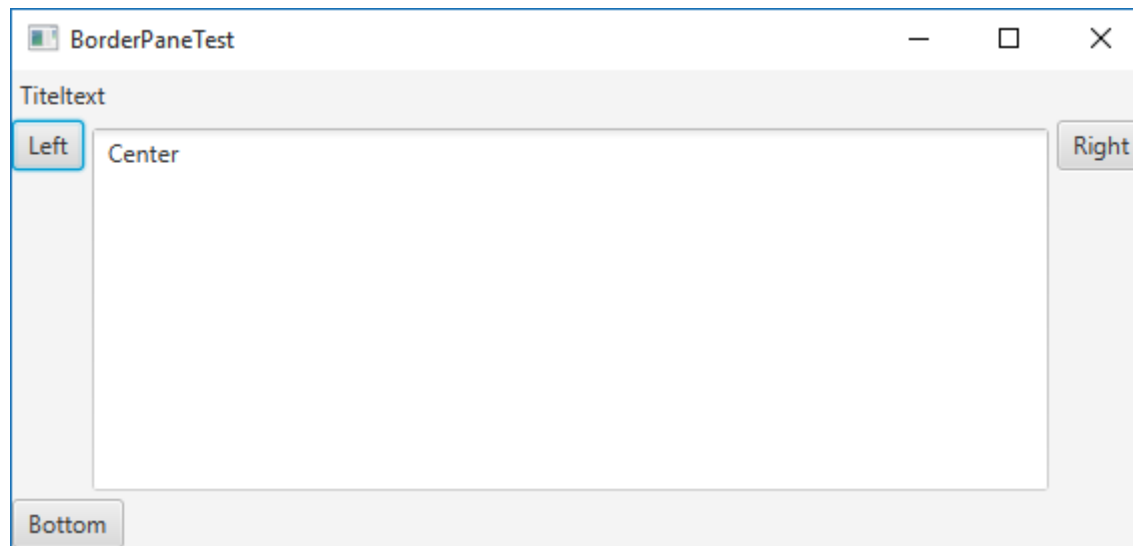
borderPane.setBottom(new Button("Bottom"));
```

...

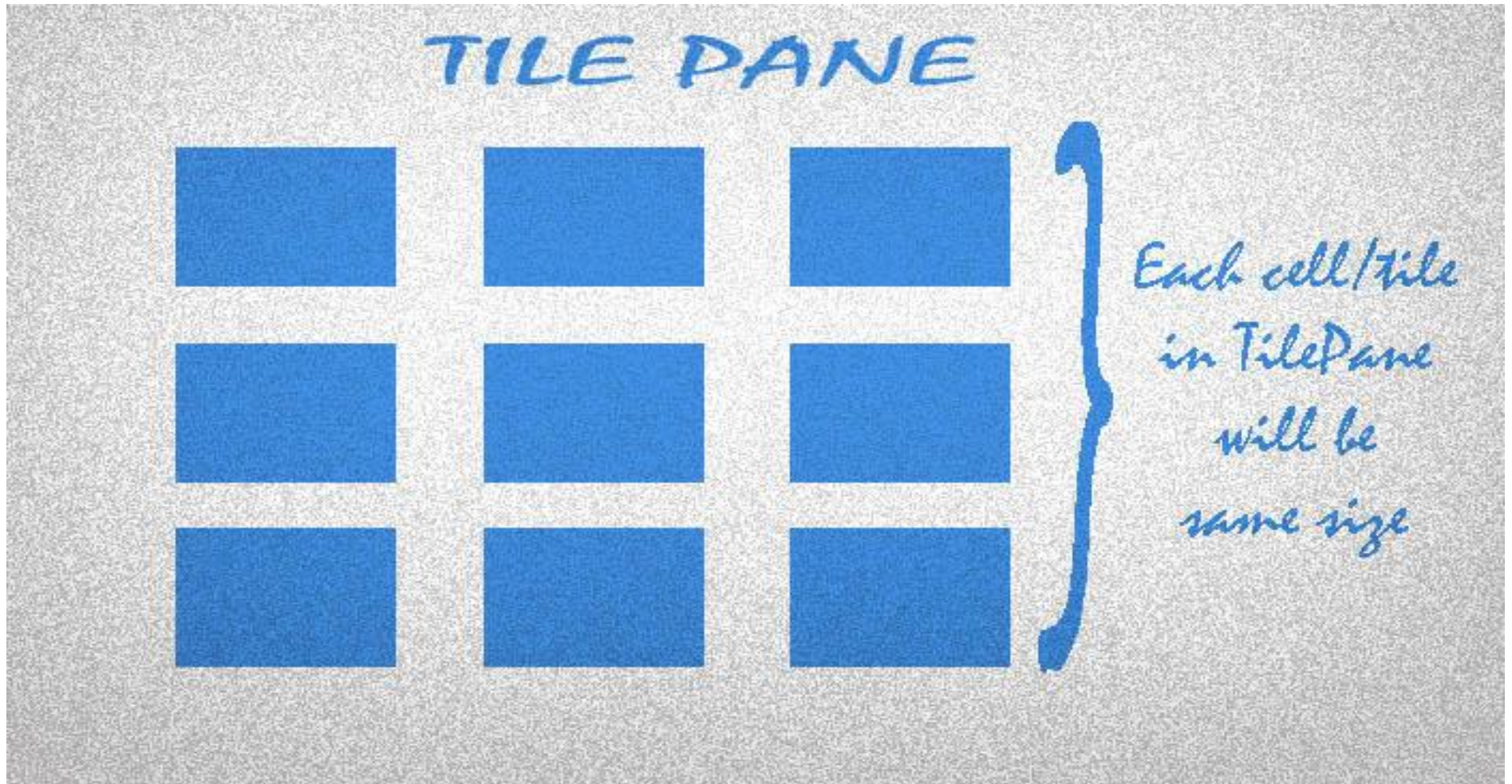
# Layouts

## BorderPane

- Ausgabe (normal und verkleinert):



# Layout Panes - Tile Pane



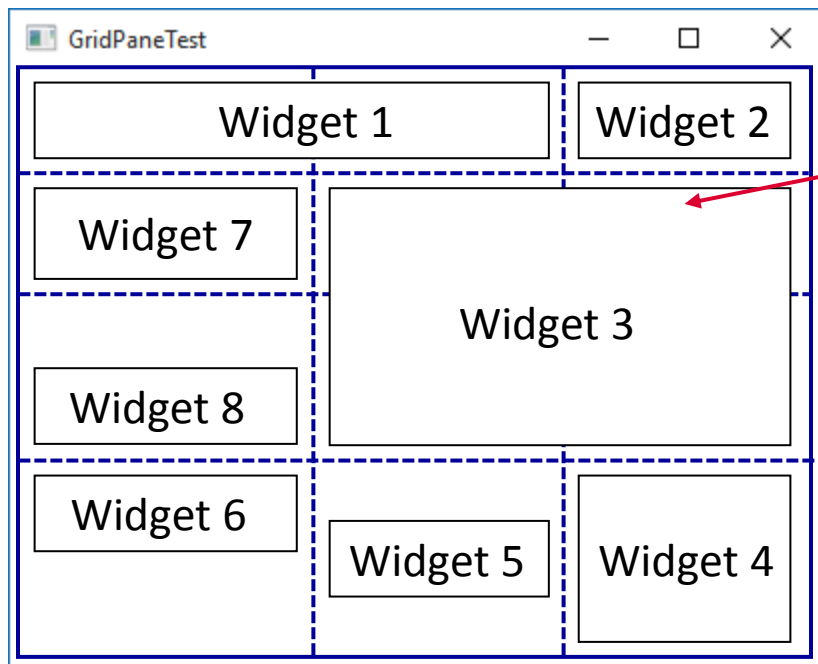
# Layout Panes - StackPane

- Stapelt jeden Knoten der hinzugefügt wird auf dem anderen
- Beispiel:
  - Blaues Rechteck
  - Button
  - „?“



## Flexible Gestaltung aufwändiger Dialoge

- **GridPane** ist der flexibelste und komplexeste Standardlayoutmanager in JavaFX.
- **GridPane** platziert die Komponenten in einem Raster aus Zeilen und Spalten.

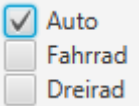
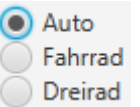

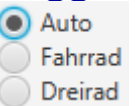
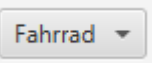


Widgets dürfen sich über mehrere Zeilen und/oder Spalten erstrecken.

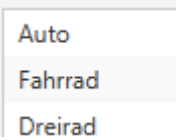


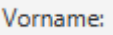

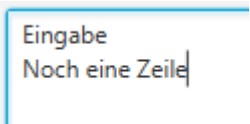
Zeilen dürfen unterschiedliche Höhen, Spalte unterschiedliche Breiten besitzen.

Die Größen der Zellen werden aus den bevorzugten Größen der Widgets berechnet.

# Widgets



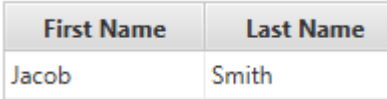
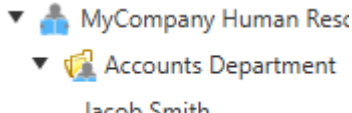
| Name                                                                                                       | Verhalten                                                                                                            |
|------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| <b>CheckBox</b><br>       | Kann selektiert und deselektiert werden. Der Selektionszustand bleibt bis zur Änderung erhalten.                     |
| <b>RadioButton</b><br>    | Kann selektiert werden. Der Selektionszustand bleibt erhalten, bis ein anderer Button derselben Gruppe gewählt wird. |
| <b>Button</b><br>         | Kann gedrückt werden. Der Selektionszustand bleibt nicht erhalten.                                                   |
| <b>ToggleButton</b><br> | Kann selektiert werden. Verhält sich wie eine <b>CheckBox</b> oder ein <b>RadioButton</b> .                          |
| <b>ComboBox</b><br>     | Zeigt einen ausgewählten Eintrag sowie eine Liste weiterer möglicher Einträge an.                                    |

# Widgets

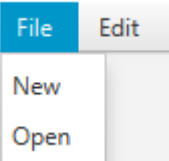

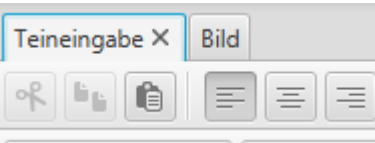
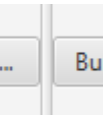
| Name                                                                                                    | Verhalten                                                                                         |
|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| <b>ListView</b><br>    | Zeigt einen ausgewählten Eintrag sowie eine Liste weiterer möglicher Einträge an.                 |
| <b>Slider</b><br>      | Kann horizontal oder vertikal verschoben werden und an bestimmten Markierungen einrasten.         |
| <b>Hyperlink</b><br>   | Auswahl einer URL                                                                                 |
| <b>Label</b><br>      | kein Verhalten, Bezeichnung und/oder Bild (z.B. für eine anderes Widget)                          |
| <b>TextField</b><br> | einzeiliges Freitexteingabefeld mit Cursorsteuerung und Selektion, keine variablen Textattribute  |
| <b>TextArea</b><br>  | mehrzeiliges Freitexteingabefeld mit Cursorsteuerung und Selektion, keine variablen Textattribute |



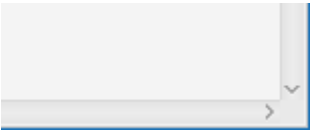
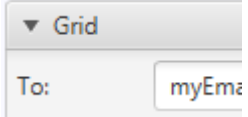
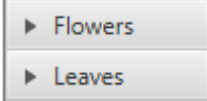
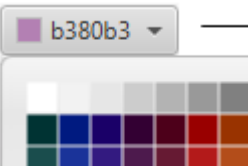
# Widgets

| Name                                                                                                      | Verhalten                                                                                                                                    |
|-----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>HTMLEditor</b><br>    | mehrzeiliges Freitexteingabefeld mit Cursor-Steuerung und Selektion, variable Textattribute, das Datenmodell im Hintergrund basiert auf HTML |
| <b>PasswordField</b><br> | Eingegebene Zeichen werden z.B. durch * dargestellt.                                                                                         |
| <b>TableView</b><br>    | Darstellung bzw. Eingabe tabellarischer Daten                                                                                                |
| <b>TreeView</b><br>    | Baum mit auf- und zuklappbaren Knoten (wie z.B. beim Explorer)                                                                               |


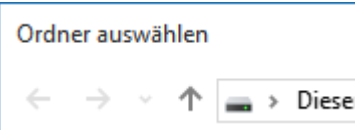
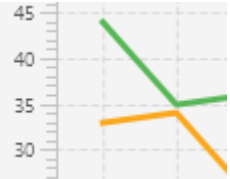
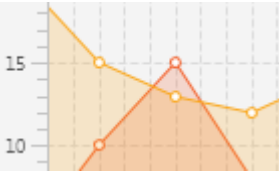
# Widgets

| Name                                                                                                    | Verhalten                                                  |
|---------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <b>Menu</b><br>        | Menu mit Einträgen und Untermenüs zum Starten von Aktionen |
| <b>ToolBar</b><br>     | Buttonleiste für Aktionsauswahl                            |
| <b>TabPane</b><br>    | Umschaltung zwischen mehreren Teildialogen                 |
| <b>SplitPane</b><br> | Variable Darstellung mehrerer Komponenten                  |

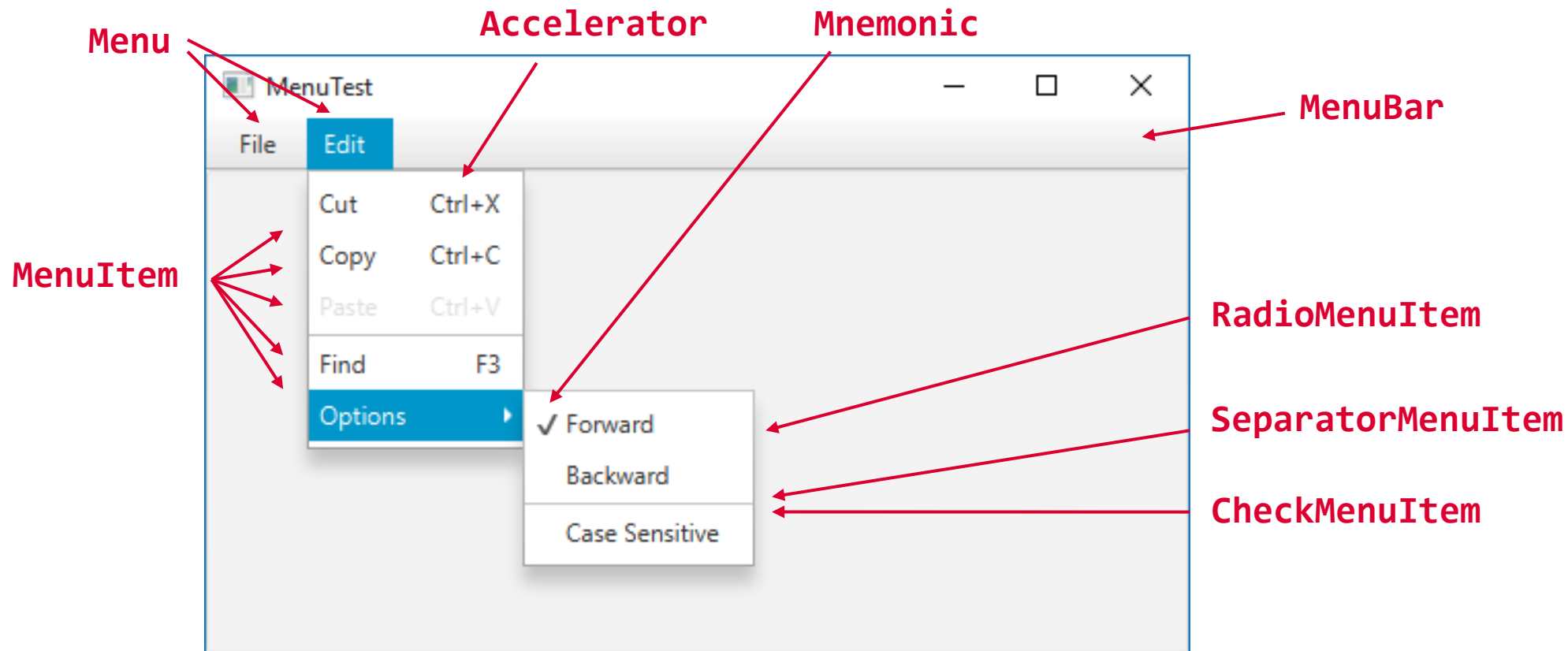
# Widgets

| Name                                                                                                      | Verhalten                                                  |
|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <b>ScrollPane</b><br>    | Verschieben und Darstellung eines Ausschnittes             |
| <b>TitledPane</b><br>    | auf- und zuklappbarer Teildialog                           |
| <b>Accordion</b><br>    | auf- und zuklappbarer Teildialog, bestehend aus TitlePanes |
| <b>ColorPicker</b><br> | Farbauswahldialog                                          |

# Widgets

| Name                                                                                                         | Verhalten                |
|--------------------------------------------------------------------------------------------------------------|--------------------------|
| <b>FileChooser</b><br>      | Dateiauswahldialog       |
| <b>DirectoryChooser</b><br> | Verzeichnisauswahldialog |
| <b>LineChart</b><br>       | Liniendiagramm           |
| <b>AreaChart</b><br>      | Diagramm                 |

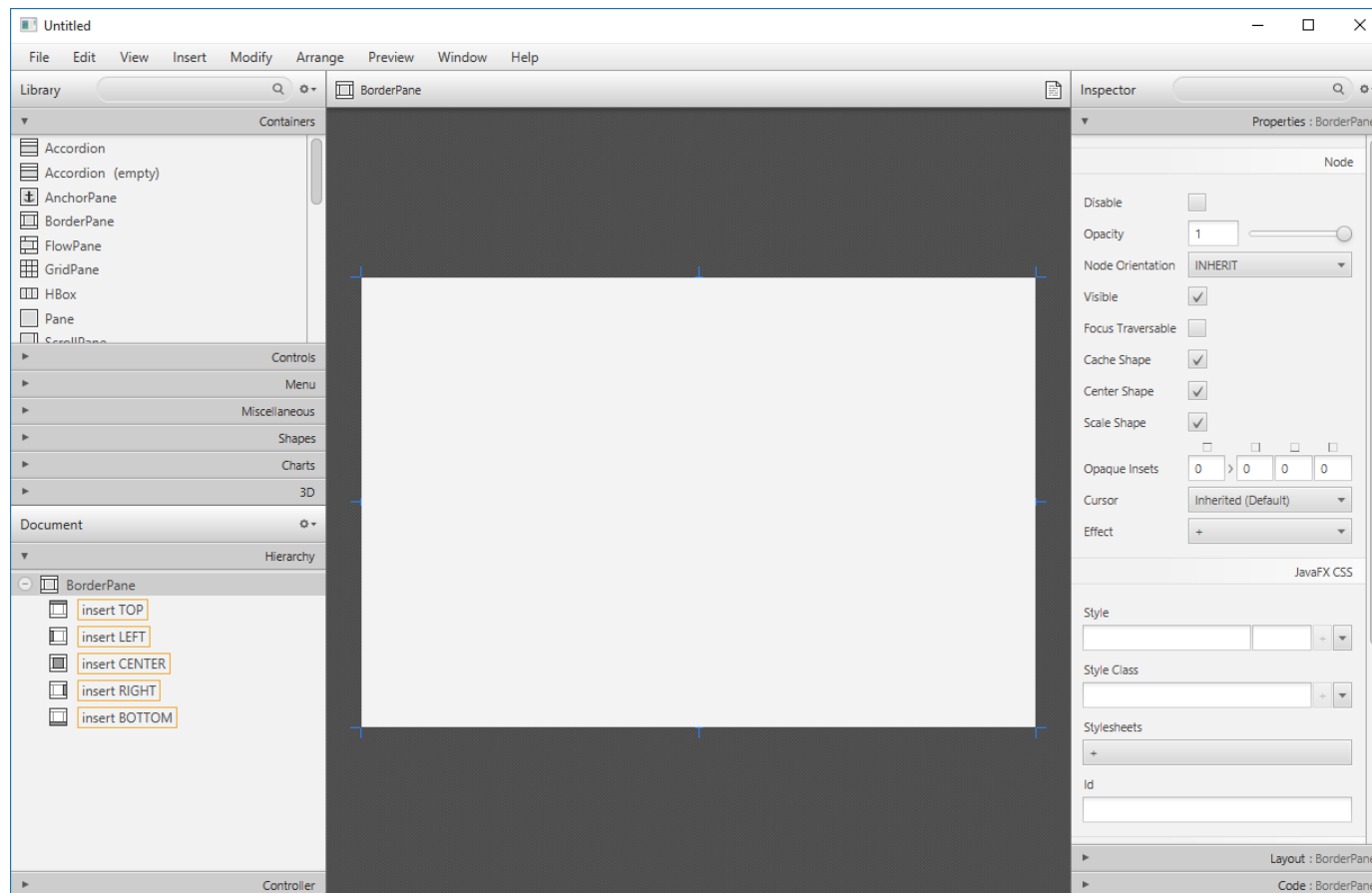
### Aufbau einer Menüstruktur



# Layouts

## Scene Builder

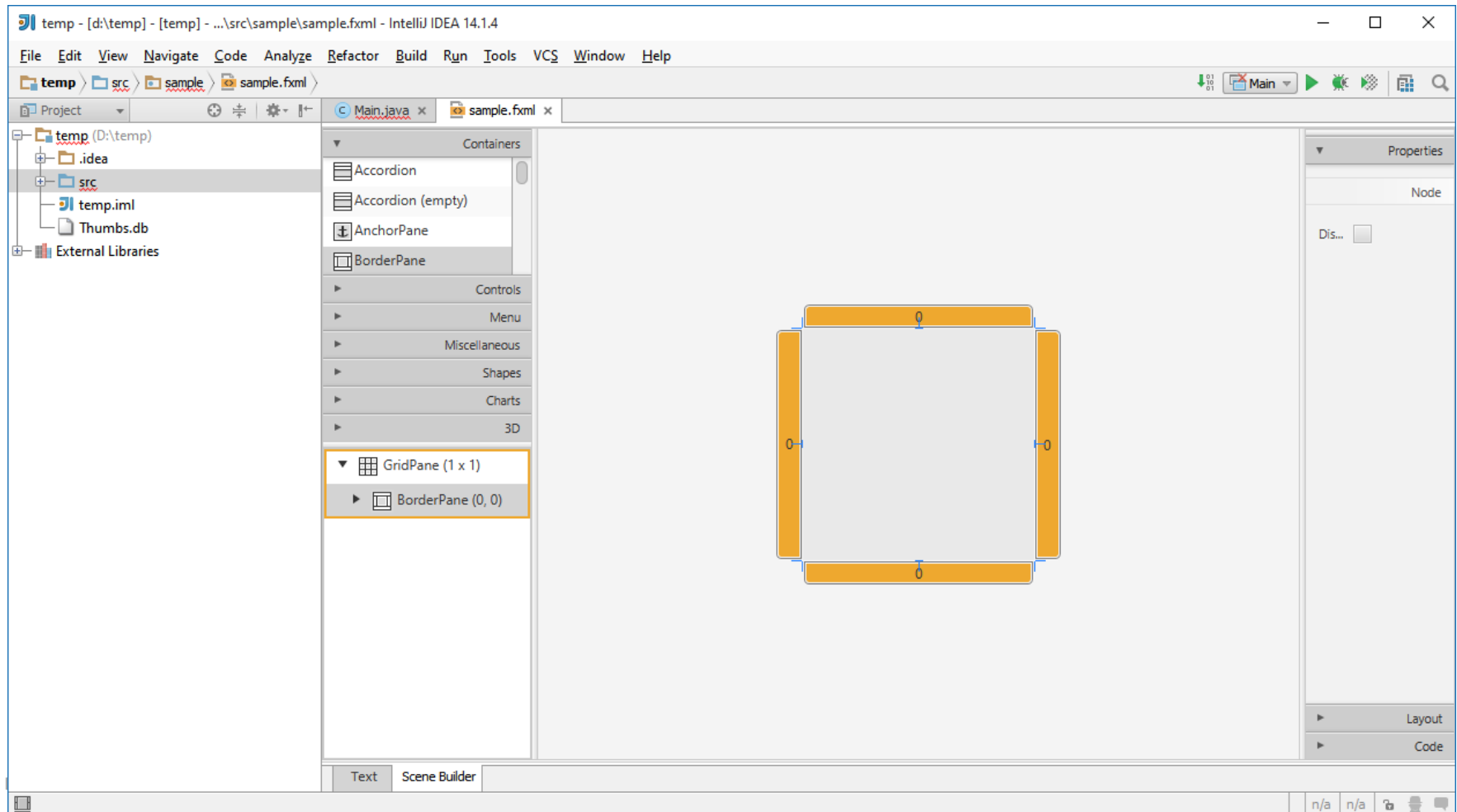
- Wer das manuelle Programmieren der Oberflächen nicht mag, nimmt einen interaktiven GUI-Builder wie den Scene Builder:



# Layouts

## IntelliJ IDEA

- In IntelliJ IDEA (Scene Builder) ist der interaktive GUI-Editor bereits integriert:



# Layout Panes

- Baue das Schülerstammblatt mittels Kombination mehrerer Layout Panes nach





# Layout Panes

| HTBLA WELS                                                                                                                    |                       | <input type="checkbox"/> Höhere Abteilung<br><input type="checkbox"/> Fachschule                                                  |  | für                         |                                                                              | Chemie                                                                  |          | 20 .. / .. |                   |         |                       |
|-------------------------------------------------------------------------------------------------------------------------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------|--|-----------------------------|------------------------------------------------------------------------------|-------------------------------------------------------------------------|----------|------------|-------------------|---------|-----------------------|
| <b>Schülerdaten:</b><br><div></div>                                                                                           |                       | Familienname                                                                                                                      |  |                             | Schulpflicht erfüllt                                                         | <input type="checkbox"/> ja<br><input type="checkbox"/> nein            |          | Klasse     | Schuljahr         | Kat.Nr. | Aufstiegs-klausel in: |
|                                                                                                                               |                       | Vorname                                                                                                                           |  |                             | Staatsbürger - schaft                                                        | <input type="checkbox"/> Österreich<br><input type="checkbox"/> deutsch |          |            |                   |         |                       |
|                                                                                                                               |                       | geboren am                                                                                                                        |  |                             | Muttersprache                                                                | <input type="checkbox"/> röm. -kath.<br><input type="checkbox"/> .....  |          |            |                   |         |                       |
|                                                                                                                               |                       | in(Ort)                                                                                                                           |  |                             | Bekenntnis                                                                   |                                                                         |          |            |                   |         |                       |
|                                                                                                                               |                       | Bezirk des Wohnortes                                                                                                              |  |                             | zuständiges Gemeindeamt                                                      |                                                                         |          |            |                   |         |                       |
|                                                                                                                               |                       | in(Land)                                                                                                                          |  |                             | SV-Nr.                                                                       |                                                                         |          |            |                   |         |                       |
|                                                                                                                               |                       | Anschrift                                                                                                                         |  | Postleitzahl, Ort           |                                                                              | Straße, Platz Nr.                                                       |          | Tel. - Nr. |                   |         |                       |
|                                                                                                                               |                       |                                                                                                                                   |  | E-Mail Adresse des Schülers |                                                                              |                                                                         |          |            |                   |         |                       |
| <b>Personaldaten aller</b><br>gesetzlichen Vertreter bei Minderjährigen.<br>Bei Volljährigkeit Daten des nächsten Verwandten. | Vertretung als        | <input type="checkbox"/> Vater <input type="checkbox"/> Mutter <input type="checkbox"/> Vormund<br><input type="checkbox"/> ..... |  | Beruf                       |                                                                              |                                                                         |          |            |                   |         |                       |
|                                                                                                                               | Familienname          |                                                                                                                                   |  | Vorname                     |                                                                              |                                                                         | Familien |            |                   |         |                       |
|                                                                                                                               | Anschrift             | Postleitzahl, Ort                                                                                                                 |  | Straße, Platz Nr.           |                                                                              | Tel. - Nr.                                                              |          |            |                   |         |                       |
|                                                                                                                               | Dienstgeber           |                                                                                                                                   |  |                             |                                                                              |                                                                         |          |            |                   |         |                       |
|                                                                                                                               | Vertretung als        | <input type="checkbox"/> Vater <input type="checkbox"/> Mutter <input type="checkbox"/> Vormund<br><input type="checkbox"/> ..... |  | Beruf                       |                                                                              |                                                                         |          |            |                   |         |                       |
|                                                                                                                               | Familienname          |                                                                                                                                   |  | Vorname                     |                                                                              |                                                                         | Familien | stand      |                   |         |                       |
|                                                                                                                               | Anschrift             | Postleitzahl, Ort                                                                                                                 |  | Straße, Platz Nr.           |                                                                              | Tel. - Nr.                                                              |          |            |                   |         |                       |
|                                                                                                                               | Dienstgeber           |                                                                                                                                   |  |                             |                                                                              |                                                                         |          |            |                   |         |                       |
|                                                                                                                               | Eingetreten am:       |                                                                                                                                   |  |                             | Ausgetreten am:                                                              |                                                                         |          |            |                   |         |                       |
|                                                                                                                               | Wiedereingetreten am: |                                                                                                                                   |  |                             | <input type="checkbox"/> MATURA<br><input type="checkbox"/> ABSCHLUSSPRÜFUNG |                                                                         |          |            | abgeschlossen am: |         |                       |
| Wiederausgetreten am:                                                                                                         |                       |                                                                                                                                   |  |                             |                                                                              |                                                                         |          |            |                   |         |                       |

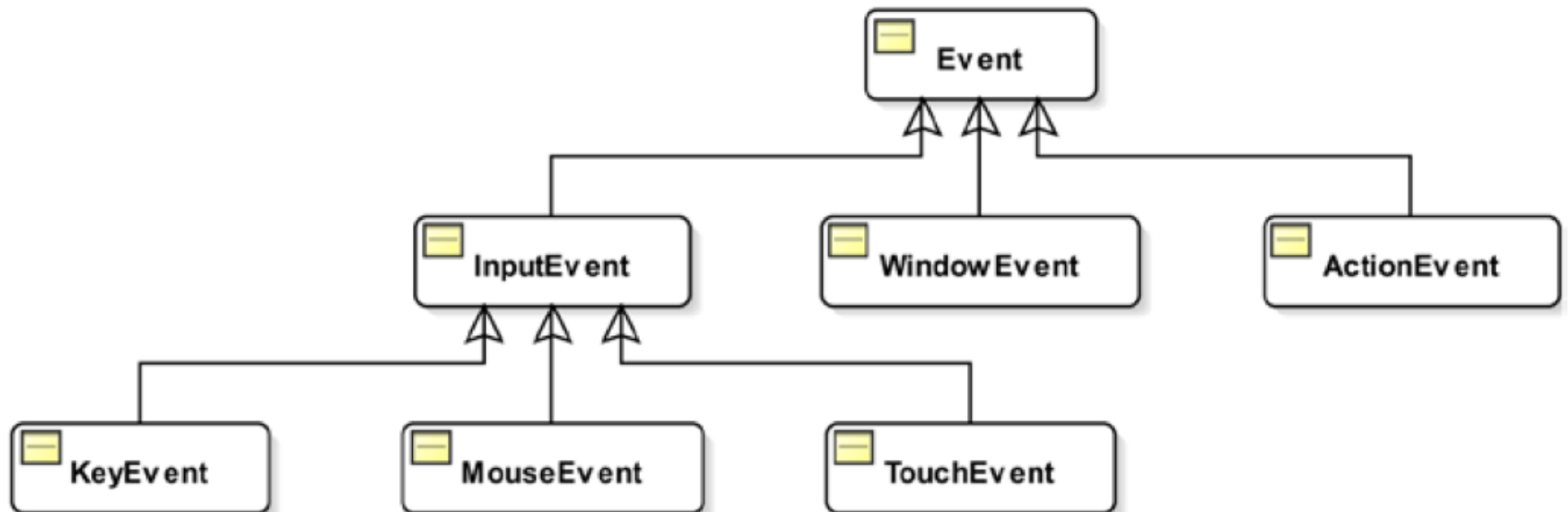
- Event (Ereignis)
  - Auftreten von User Interaktion mit der Anwendung
  - Behandlung des Ereignisses = Event Handling
  - JavaFX: Klasse `javafx.event.Event`

# Event Handling

- Jedes Event hat 3 Eigenschaften:
  - Event Source (Keyboard, Mouse, ...)
  - Event Target (Button, ...)
  - Event Type (Mouse pressed, ...)

| Name                | Class/Interface | Beschreibung                                          |
|---------------------|-----------------|-------------------------------------------------------|
| <i>Event</i>        | Class           | Ereignis                                              |
| <i>EventTarget</i>  | Interface       | Ziel des Events, z.B. Button                          |
| <i>EventType</i>    | Class           | Typ des Events, z.B. Keypressed                       |
| <i>EventHandler</i> | Interface       | Ereignisbehandler -><br>implementiert <i>handle()</i> |

# Event Handling



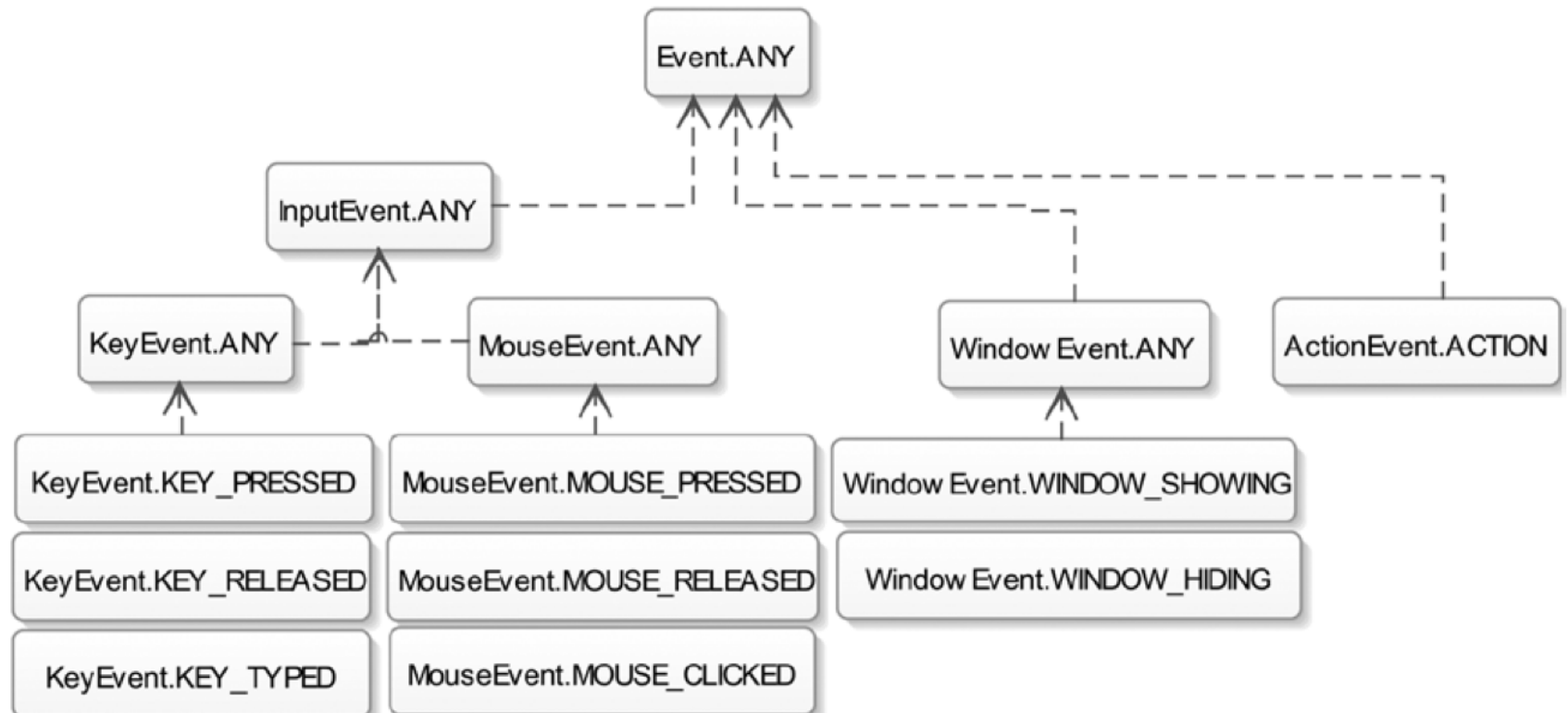
## ■ Event Target

- UI Element, das auf Ereignisse reagieren kann
- muss Interface *EventTarget* implementieren
- z.B. *Window*, *Scene*, *Node* implementieren Interface *EventTarget*

# Event Handling

## ■ Event Types

- dienen dazu die Art des Ereignisses innerhalb eines Event Targets zu unterscheiden



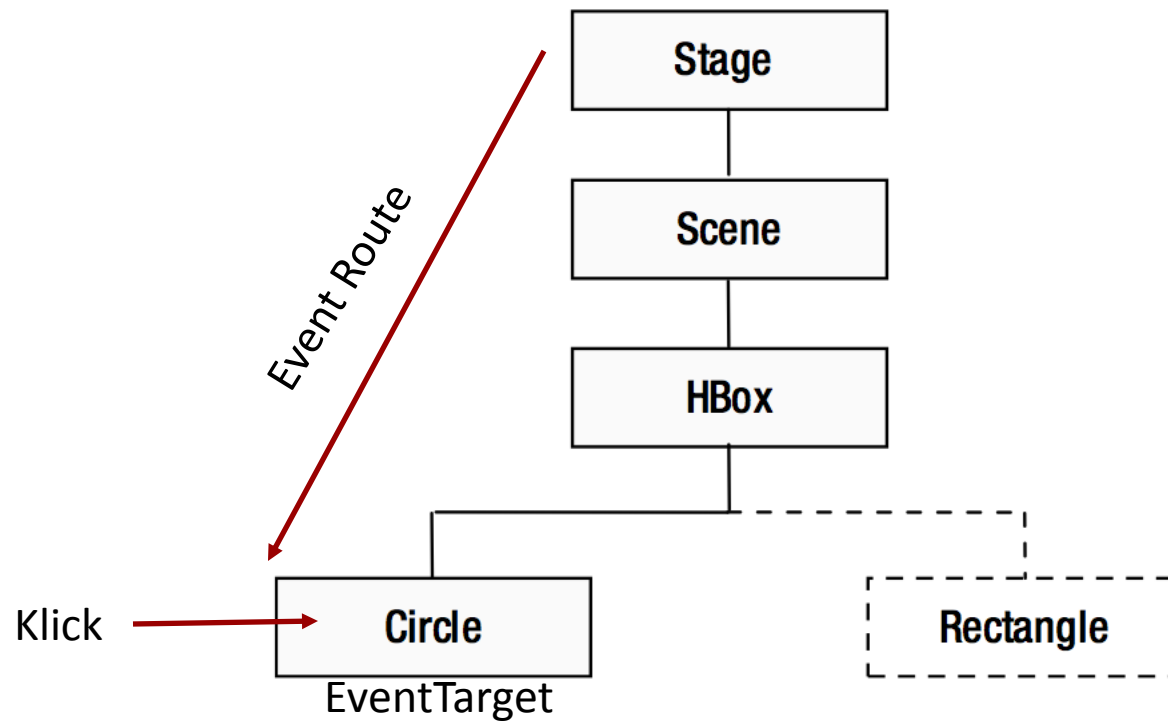
- Event Target Selection

- Wie wird das Event Target ausgewählt?
- Maus Events: oberster Knoten, über dem der Mauszeiger steht
- Key Events: Element mit Fokus

- Event Route
  - Events “wandern” entlang der Event Dispatch Chain („Abarbeitungskette“)
  - Bsp: Circle und Rectangle in Hbox, Hbox ist root node der Scene innerhalb einer Stage
  - Circle wird angeklickt -> Event Target
  - Circle baut Event Route auf (von Stage zu Circle)



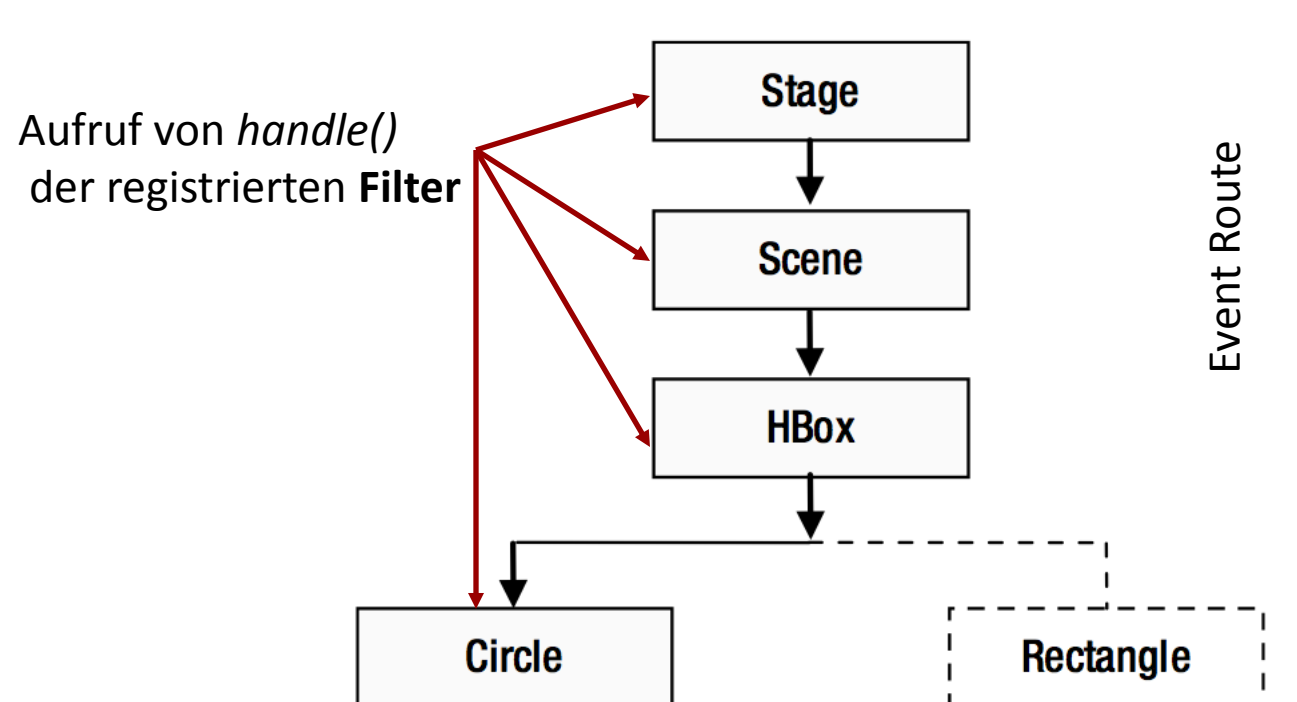
# Event Handling



- Durchlauf der Event Route
  - 2 Schritte:
    - Capture Phase: von oben nach unten
    - Bubbling Phase: von unten nach oben
  - am Weg der Event Route werden alle Nodes über Event informiert und können darauf reagieren
  - setzt voraus, dass Handler/Filter registriert wurden

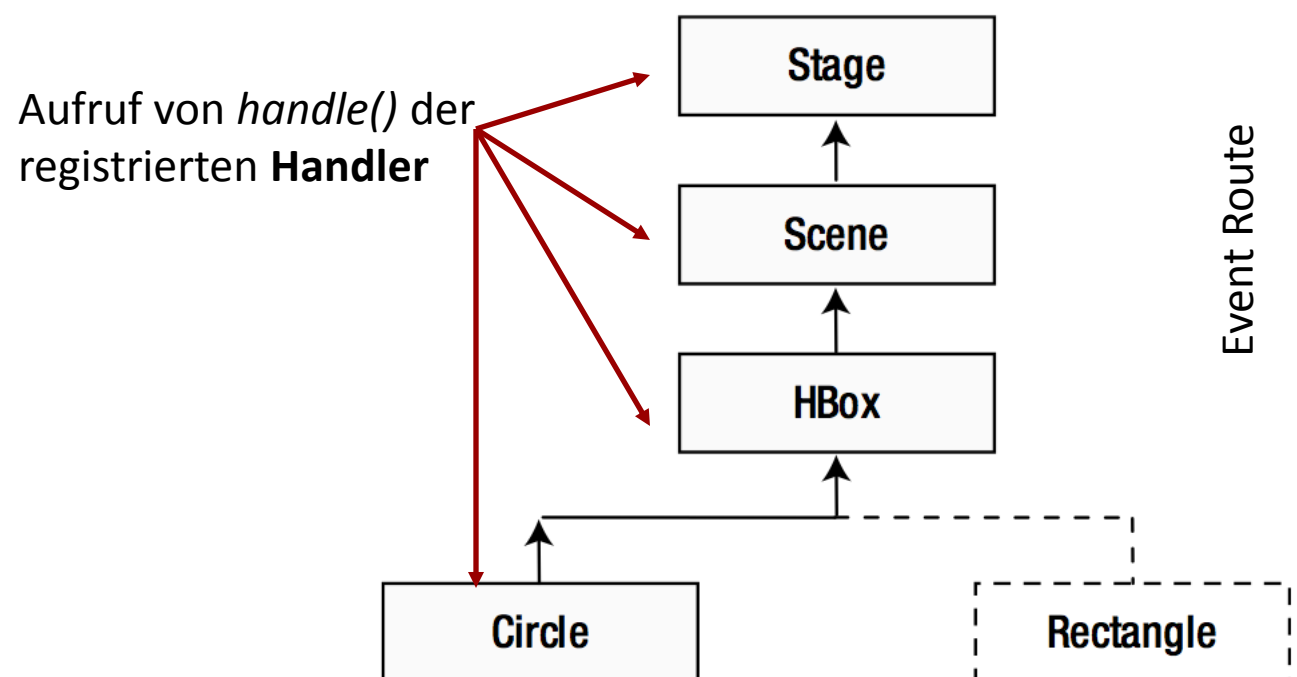
# Event Handling

- Event Capture Phase



# Event Handling

- Event Bubbling Phase



# Event Handling

- 2 Schritte notwendig, um Ereignis zu behandeln:
  - 1. EventHandler implementieren

```
public interface EventHandler<T extends Event> extends EventListener {  
    void handle(T event);  
}
```
  - 2. EventHandler/Filter registrieren

## 1. EventHandler implementieren – V1:

innere Klassen

```
EventHandler<MouseEvent> aHandler = new EventHandler<MouseEvent>() {  
    @Override  
    public void handle(MouseEvent e) {  
        /* Event handling code goes here */  
    }  
};
```

## 2. EventHandler/Filter registrieren – V1:

### **addXXX() / removeXXX() Methoden**

```
<T extends Event> void addEventFilter(EventType<T> eventType,  
                                     EventHandler<? super T> eventFilter)
```

```
<T extends Event> void addEventHandler(EventType<T> eventType,  
                                       EventHandler<? super T> eventHandler)
```

# Event Handling

```
...  
Circle circle = new Circle (100, 100, 50);  
  
// Create a MouseEvent filter  
EventHandler<MouseEvent> mouseEventFilter =  
    e -> System.out.println("Mouse event filter has been called.");  
  
// Create a MouseEvent handler  
EventHandler<MouseEvent> mouseEventHandler =  
    e -> System.out.println("Mouse event handler has been called.");  
  
// Register the MouseEvent filter and handler to the Circle  
// for mouse-clicked events  
circle.addEventFilter(MouseEvent.MOUSE_CLICKED, mouseEventFilter);  
circle.addEventHandler(MouseEvent.MOUSE_CLICKED, mouseEventHandler);
```



# Event Handling

```
// Create a MouseEvent  
  
// EventHandler object  
  
EventHandler<MouseEvent> aHandler =  
    e -> System.out.println ("Mouse event filter or handler has been called.");  
  
// Register the same EventHandler object as the MouseEvent  
  
// filter and handler  
  
// to the Circle for mouse-clicked events circle.addEventFilter(MouseEvent.MOUSE_CLICKED,  
aHandler); circle.addEventHandler(MouseEvent.MOUSE_CLICKED, aHandler);
```

# Event Handling

| EventHandler implementieren                                                                                                                                                                                                                                        | EventHandler registrieren                                                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Anonyme, innere Klasse</b></p> <pre>EventHandler&lt;MouseEvent&gt; aHandler = new<br/>EventHandler&lt;MouseEvent&gt;() {<br/>    @Override<br/>    public void handle(MouseEvent e) {<br/>        /* Event handling code goes here */<br/>    }<br/>};</pre> | <p><b>addXXX() Methoden</b></p> <pre>btn.addEventFilter(MouseEvent.MOUSE_CLICKED,<br/>aHandler);<br/><br/>btn.addEventHandler(MouseEvent.MOUSE_CLICKED,<br/>aHandler);</pre> |
| <p><b>Lamda Expressions</b></p> <pre>EventHandler&lt;MouseEvent&gt; aHandler = e-&gt; {...};</pre>                                                                                                                                                                 | <p><b>setOnXXX() Methoden</b></p> <pre>btn.setOnMouseClicked(eventHandler);</pre> <p><b>Oder Kurzschreibweise:</b></p> <pre>btn.setOnMouseClicked(e-&gt; {...});</pre>       |

# Event Handling

- `removeEventFilter()` / `removeEventHandler()`
  - Filter/Handler können gelöscht werden -> ab diesem Zeitpunkt keine Behandlung mehr

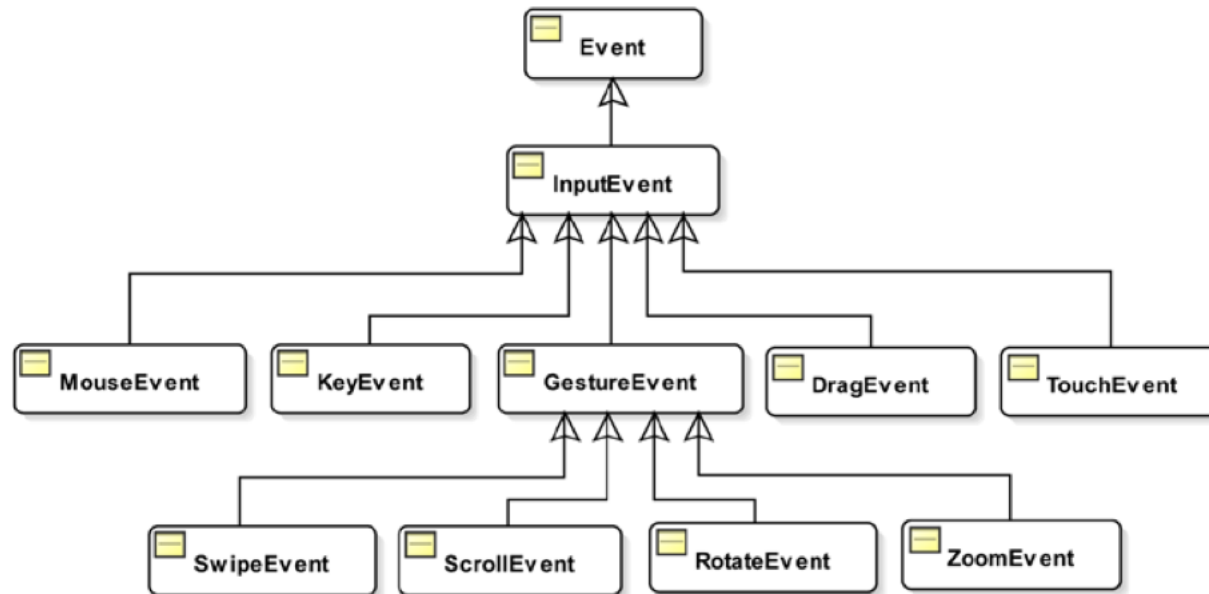
```
circle.removeEventFilter(MouseEvent.MOUSE_CLICKED, handler);  
circle.removeEventHandler(MouseEvent.MOUSE_CLICKED, handler);
```

# Event Handling

- ein Event gilt als konsumiert, wenn die `.consume()` Methode aufgerufen wird
- in diesem Fall ist die Ereignisbehandlung komplett und das Ereignis wird nicht mehr weitergereicht
- üblicherweise wird `.consume()` in der Methode `.handle()` aufgerufen

# Event Handling

- Input Events



# Event Handling

- Mouse Events

| Mouse Event    | Beschreibung                       |
|----------------|------------------------------------|
| ANY            | Alle Mouse Events                  |
| MOUSE_PRESSED  | Mausbutton gedrückt                |
| MOUSE_RELEASED | Mausbutton losgelassen             |
| MOUSE_CLICKED  | Mausbutton gedrückt+losgelassen    |
| MOUSE_MOVED    | Maus bewegt                        |
| MOUSE_ENTERED  | Maus in Node bewegt                |
| MOUSE_EXITED   | Maus aus Node bewegt               |
| MOUSE_DRAGGED  | Maus bewegt mit Maustaste gedrückt |

# Event Handling

- Status von Mouse Buttons (in MouseEvent Objekt gespeichert)

| Mouse Button Status          | Beschreibung                               |
|------------------------------|--------------------------------------------|
| MouseButton getButton()      | MouseButton, der zum Ereignis geführt hat  |
| bool isPrimaryButtonDown()   | true, wenn Primärbutton gedrückt wurde     |
| bool isSecondaryButtonDown() | true, wenn Sekundärbutton gedrückt wurde   |
| bool isMiddleButtonDown()    | true, wenn mittlerer Button gedrückt wurde |

# Event Handling

- Key Events

| Mouse Event  | Beschreibung               |
|--------------|----------------------------|
| ANY          | Alle Key Events            |
| KEY_PRESSED  | Taste gedrückt             |
| KEY_RELEASED | Taste losgelassen          |
| KEY_TYPED    | Taste gedrückt+losgelassen |



# Event Handling

- Status von Tasten Buttons (in KeyEvent Objekt gespeichert)

| Mouse Button Status   | Beschreibung                                         |
|-----------------------|------------------------------------------------------|
| KeyCode getCode()     | KeyCode Enum liefert Tastencode für eingegeben Taste |
| String getText()      | Stringbeschreibung der gedrückten Taste              |
| String getCharacter() | Eingegebene Taste als String                         |