

ENU

## Registrierung

- New feature 1
- New feature 2
- New feature 3
- Kleinere Verbesserungen
- Fehlerbehebungen

Aktualisieren

— 20.0 +



①  
INFO

8:47

09-GUI-1

Objektorientierte Programmierung | Matthias Tichy

# Lernziele

- Kurzeinführung Graphische Benutzeroberflächen und JavaFX
- Widgets
- Layout

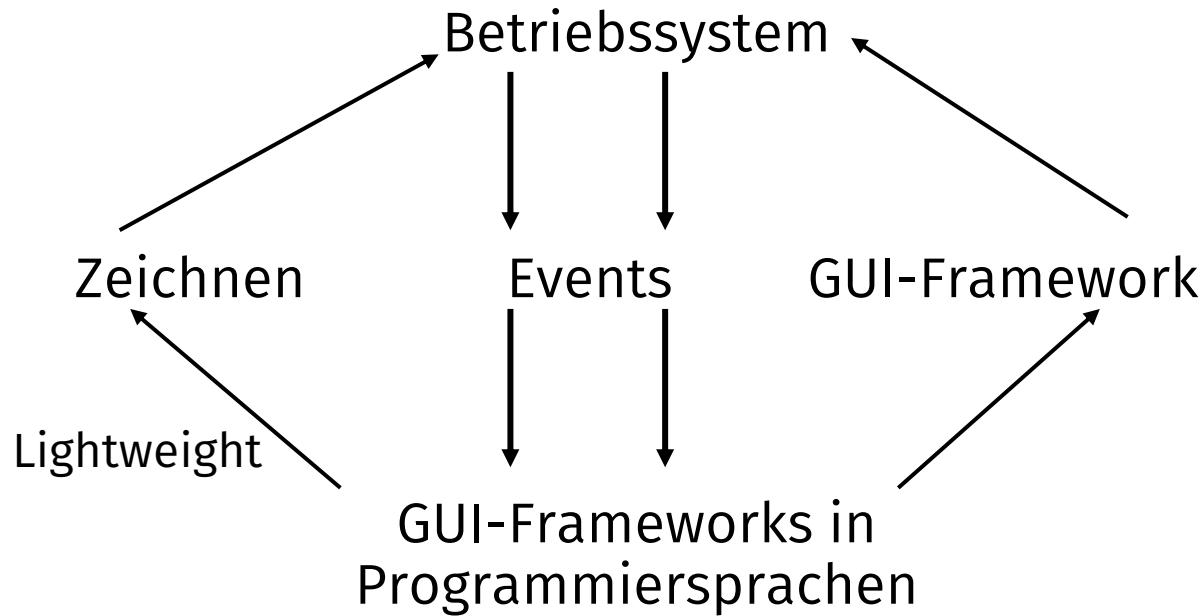
# Graphische Benutzeroberflächen (GUIs)

- Arten von Benutzerschnittstellen (UI)
- Unterschiede zwischen Text und GUI? Vor- und Nachteile?

# Geschichte

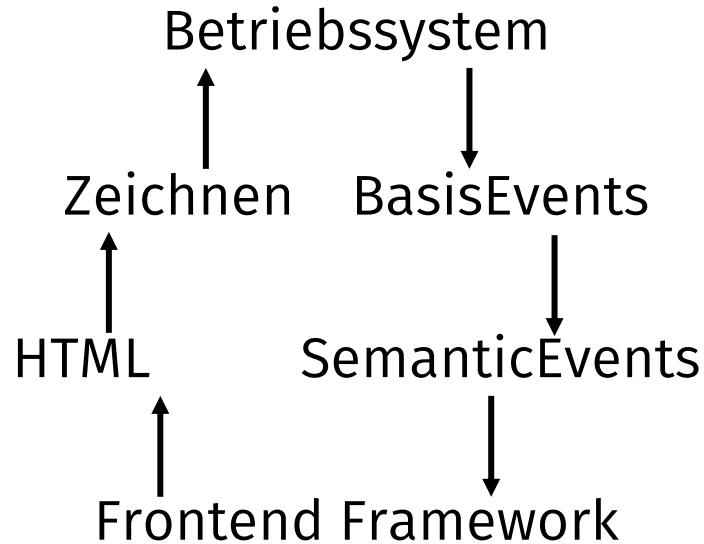


# technische Umsetzung



# technische Umsetzung

Im Browser



- Mehr in Vorlesung: Web Engineering

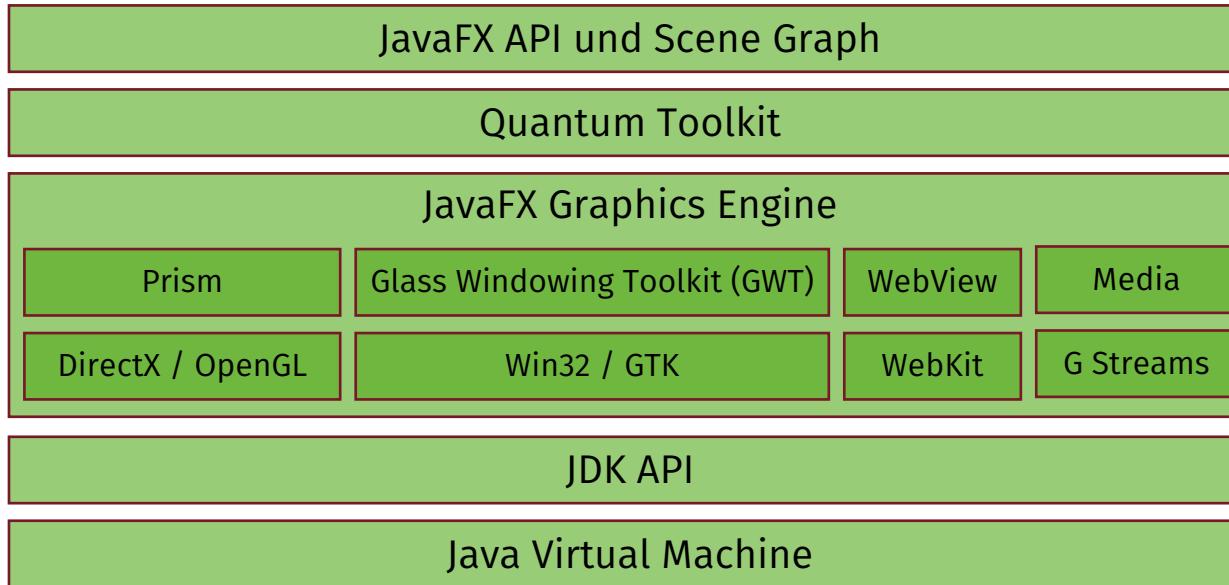
# Beispiele

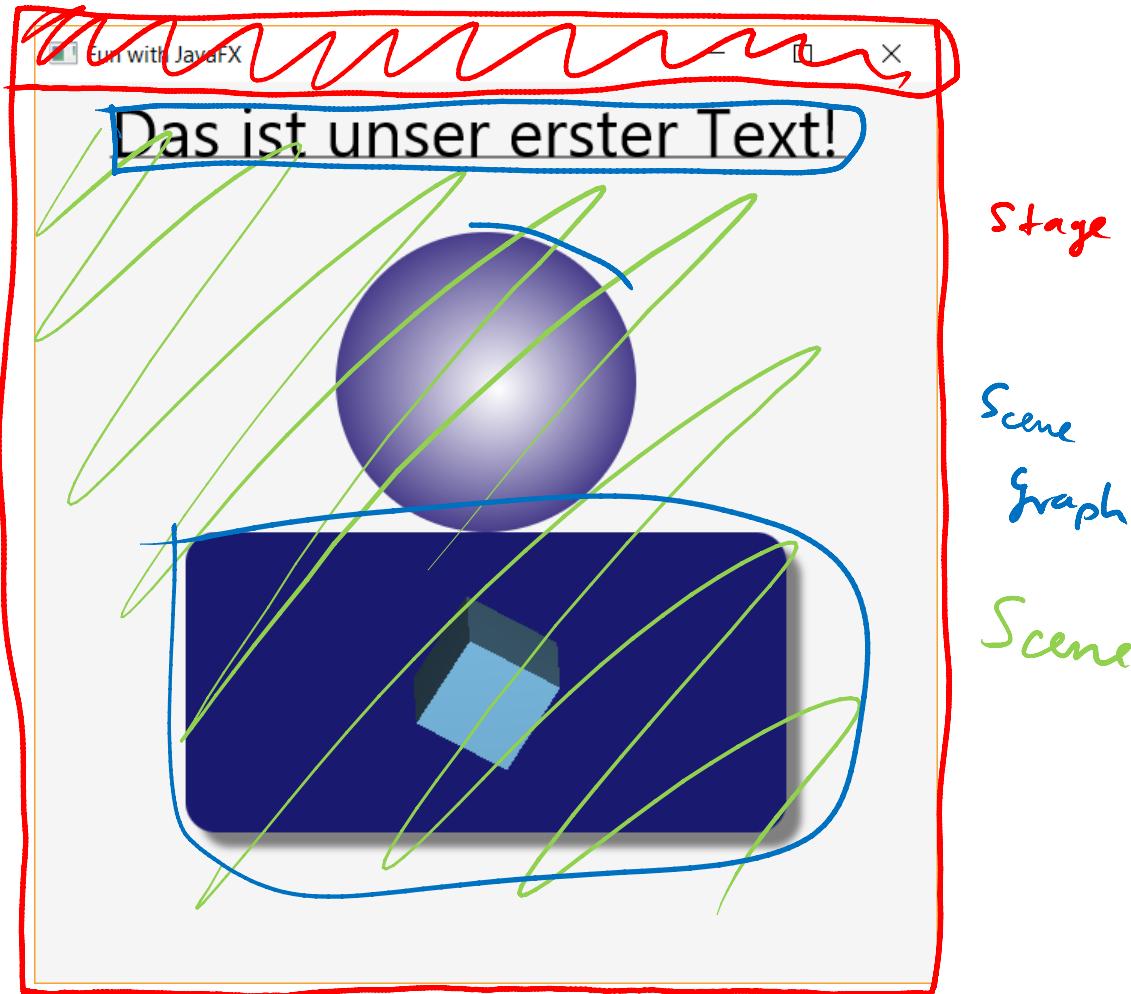
- Java: AWT, Swing, SWT, Java FX
- C++: GTK+, Qt, VCL, wxWidgets
- C#: Windows-Forms (.NET)
- Delphi: Visual Component Library (VCL)
- Objective-C: Cocoa Application Kit
- Javascript: HTML/SVG/(React|Angular...)/(MaterialUI|...)
- oft auch Wrapper für andere Sprachen für z.B. GTK+, Qt

# JavaFX

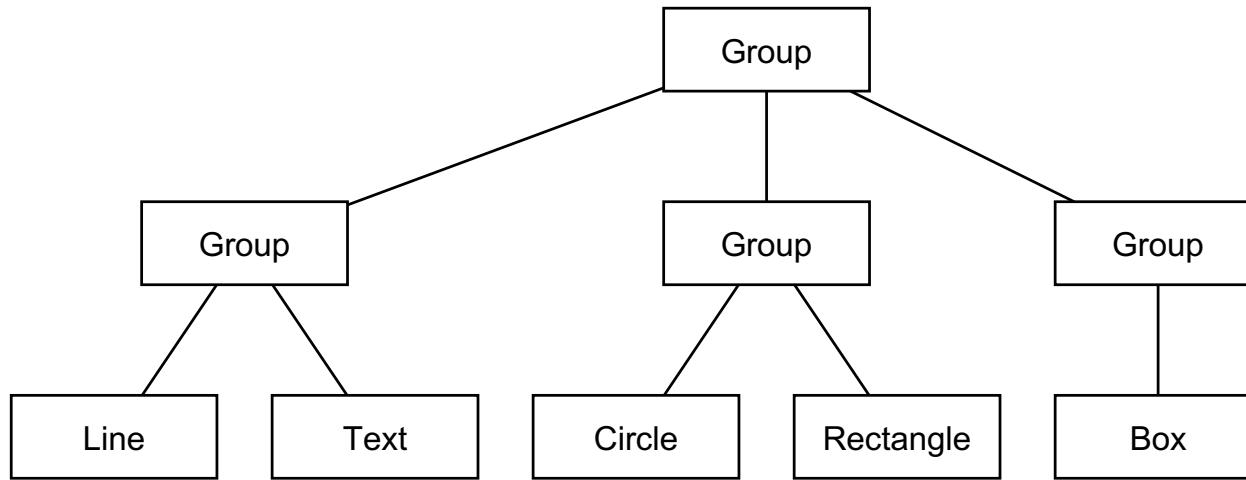
- Lightweight
- viele vordefinierte UI-Komponenten
- CSS Unterstützung (analog zum Web)
- Unterstützung für alternative Eingabemöglichkeiten (z.B. Multitouch)
- Canvas API zum Zeichnen von beliebigen Grafiken
- 3D-Grafik Unterstützung
- Unterstützung von Internationalisierung
- ....

# Architektur





# Scene Graph



# JavaFX Anwendung

- Klasse leitet von **Application** ab
- drei Methoden:
  - **init()**
  - **start(Stage primaryStage)**
  - **stop()**
- statische Methode **launch(String args[])** erstellt Instanz der Klasse, ruft **init()** auf, dann **start(..)** und kurz vor Ende **stop()**
- in **start(..)** wird Scene Graph aufgebaut

# JavaFX Scene Graph

- Schritte für Scene Graph:

1. Objekt erstellen

```
Text text = new Text();
```

2. Eigenschaften einstellen

```
text.setFont(new Font(45));  
text.setX(50);  
text.setY(50);  
text.setText("Das ist unser erster Text!");
```

3. in Baum einfügen

```
underlinedText.getChildren().add(text);
```

# JavaFX Scene Graph

- Die schlechte Nachricht:  
Man muss die ganzen Eigenschaften kennen und kombinieren können
- Die gute Nachricht:  
Sehr viele Ähnlichkeiten zwischen verschiedenen Frameworks → Wiederverwendbarkeit des Wissens

# Literatur

- Packages:  
`javafx.*`  
`javafx.scene.*`
- Externe Bibliothek seit Java 17 → Gradle/Maven
- Doku/Tutorials:
  - <https://openjfx.io/openjfx-docs/>
  - <https://openjfx.io/javadoc/18/>
  - <https://dzone.com/refcardz/javafx-8-1>
  - <http://code.makery.ch/library/javafx-8-tutorial/>

# Controls (Auswahl)

This is a Button!

This is a Label!

TextField with PromptText

TextArea with some text

ComboBox option 1  
ComboBox option 2  
ComboBox option 3

Enable Checkbox 1  
Enable Checkbox 2  
Enable Checkbox 3

Either Radiobutton 1 or ...  
... Radiobutton 2 or ...  
... Radiobutton 3.

ListView item 1  
ListView item 2  
ListView item 3  
ListView item 4

ScrollPane

First Col Second Col

TableView (0,0)	Example (1,0)
TableView (0,1)	Example (1,1)
TableView (0,2)	Example (1,2)
TableView (0,3)	Example (1,3)

root node

- ▼ Treelitem 1
  - Treelitem 1/1
  - Treelitem 1/2
  - Treelitem 1/3
- Treelitem 2
- Treelitem 3

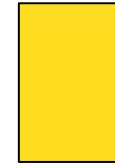
# GUI-Komponenten

## Die GUI-Komponenten

... waren mir bekannt



... habe ich schon einmal gehört



... praktisch neu



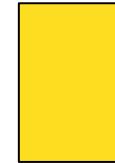
# Welche GUI?

Auswahl von Dateien, die kopiert werden sollen

... Checkboxen



... Liste



... RadioButtons



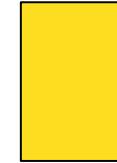
# Welche GUI?

Eine Option aus einer größeren Liste auswählen

... RadioButton



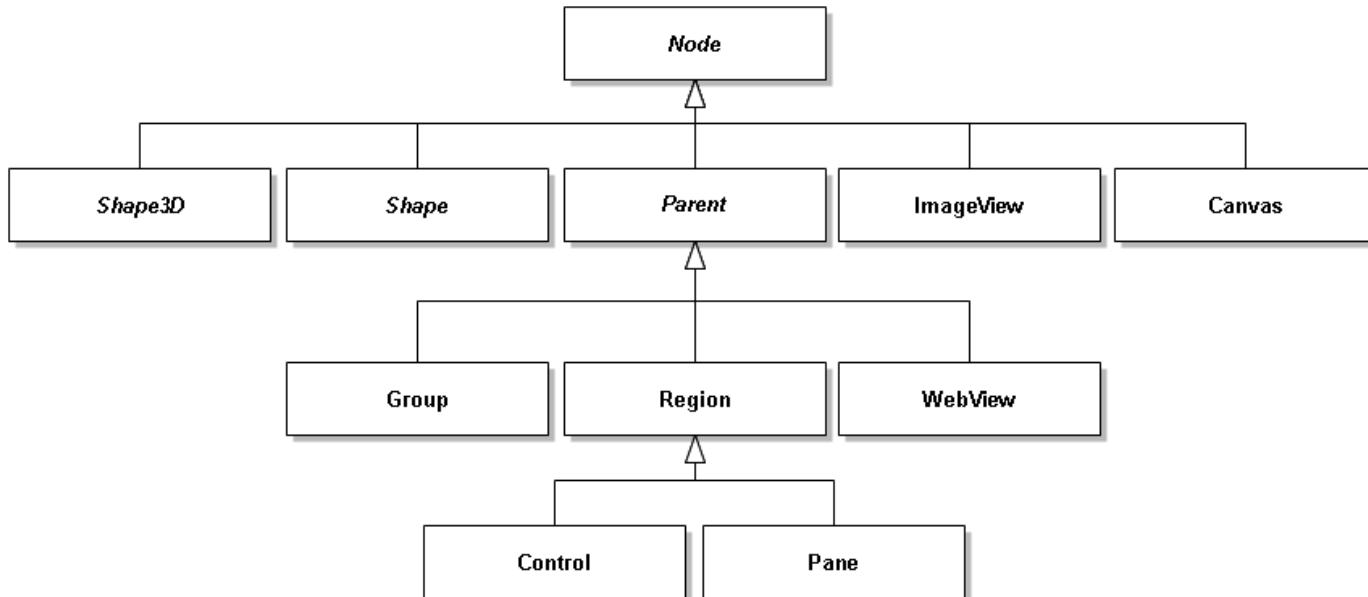
... ComboBox



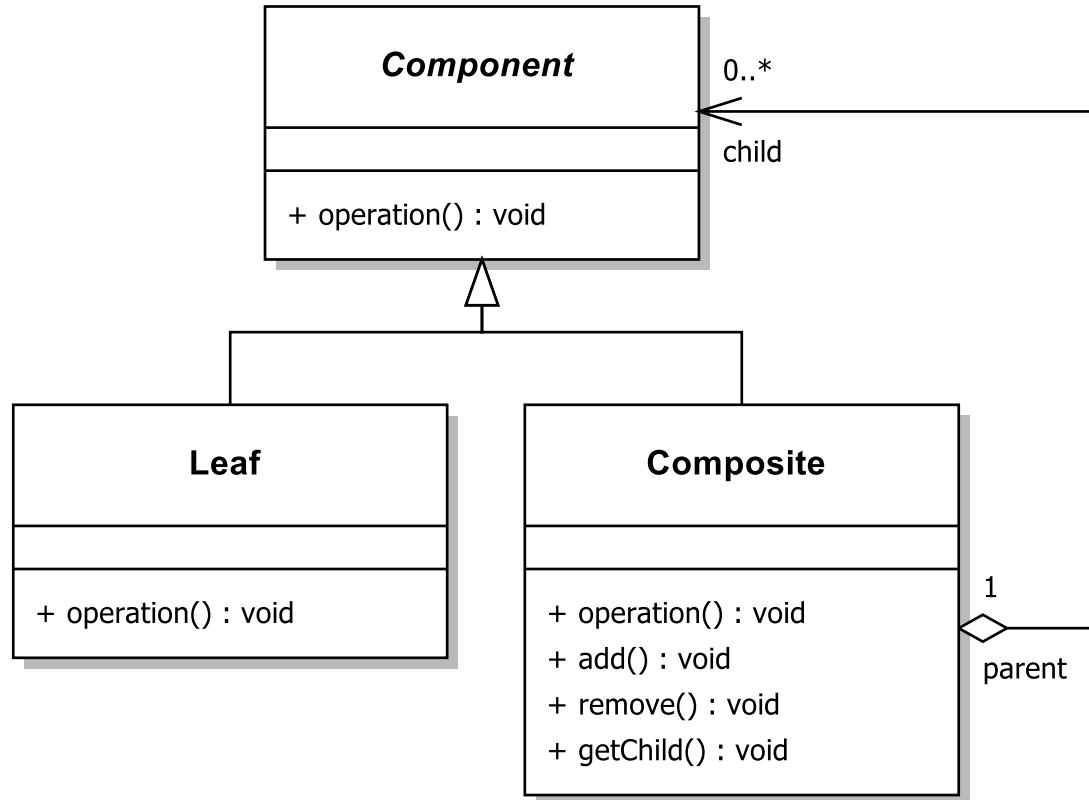
... CheckBoxen



# Hierarchie



# Composite Pattern



# Composite Pattern

- einfache Repräsentation verschachtelter Strukturen
- gleiche Behandlung von Blatt- und Zwischenknoten → vereinfachte Anwendung
- Operationen werden automatisch durchgereicht
- Flexibilität bzgl. Erweiterungen
- was steht in "operation()"?

# Node

- Benutzerdefinierte Eigenschaften
- Transformationen
- Eventhandling
- Drag&Drop
- Clipping

# Region

- |  |   |
|--|---|
| <ul style="list-style-type: none"><li>• Background</li><li>• Borders</li></ul> | <ul style="list-style-type: none"><li>• CSS-Styling</li><li>• Basic layouting</li></ul> |
|--|---|

# Control

- Tooltips
- Kontextmenü
- viel weitere Funktionalität

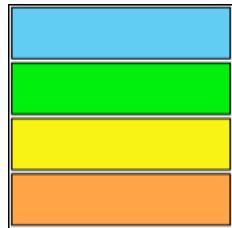
# Pane

- Basisklasse für Layoutmanager

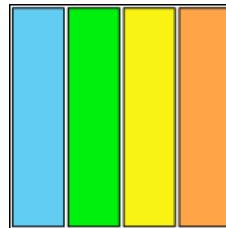
# LayoutManager – Wozu?

- Unterschiedliche Bildschirmauflösungen, Fontgrößen, Fenstergrößen
- GUI sollte sich möglichst selbst anpassen
- Verzicht auf absolute Positionen
- geschickter Einsatz von Layoutmanagern

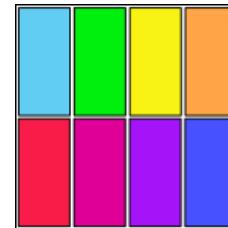
# LayoutManager – Überblick



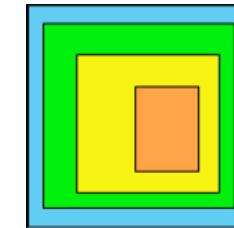
VBox



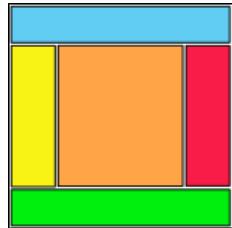
HBox



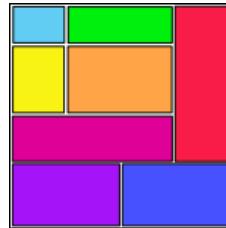
TilePane



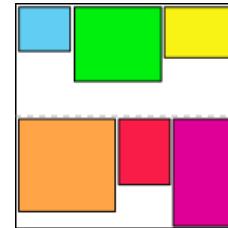
StackPane



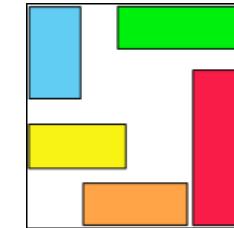
BorderPane



GridPane



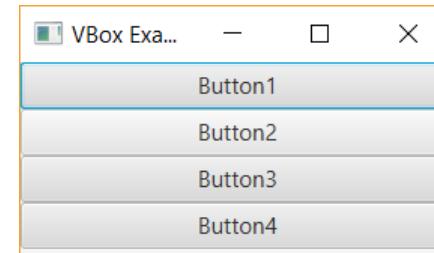
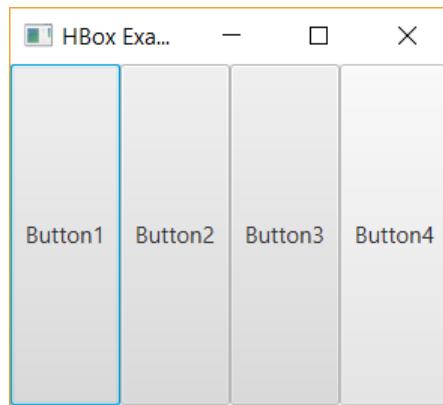
FlowPane



AnchorPane

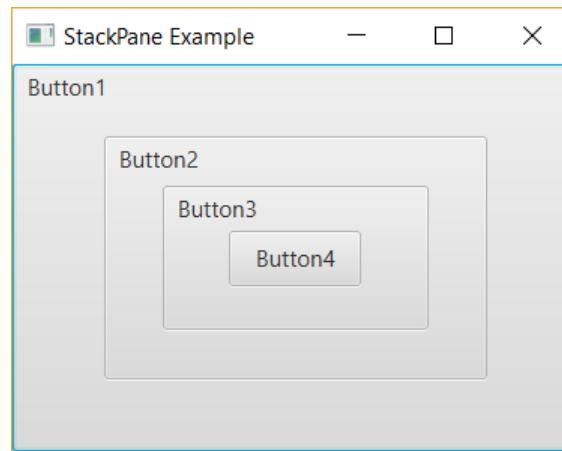
# Hbox, VBox

- die Komponenten werden horizontal bzw. vertikal angeordnet



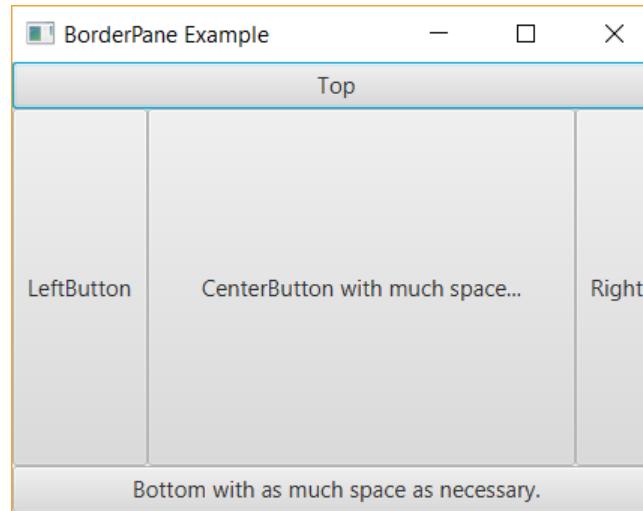
# StackPane

- die Komponenten werden übereinander angeordnet
- absolute Positionierung oder relative Ausrichtung



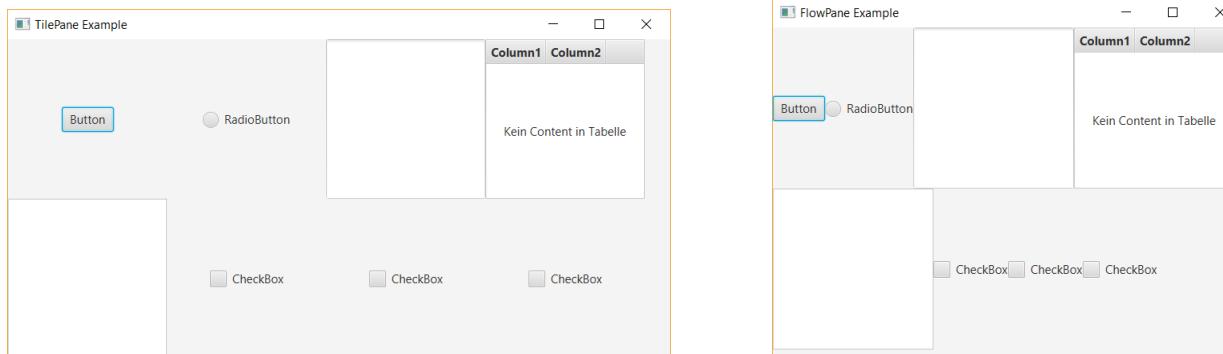
# BorderPane

- die Komponenten werden in bestimmten Bereichen eingeordnet (top, left, right, bottom, center)



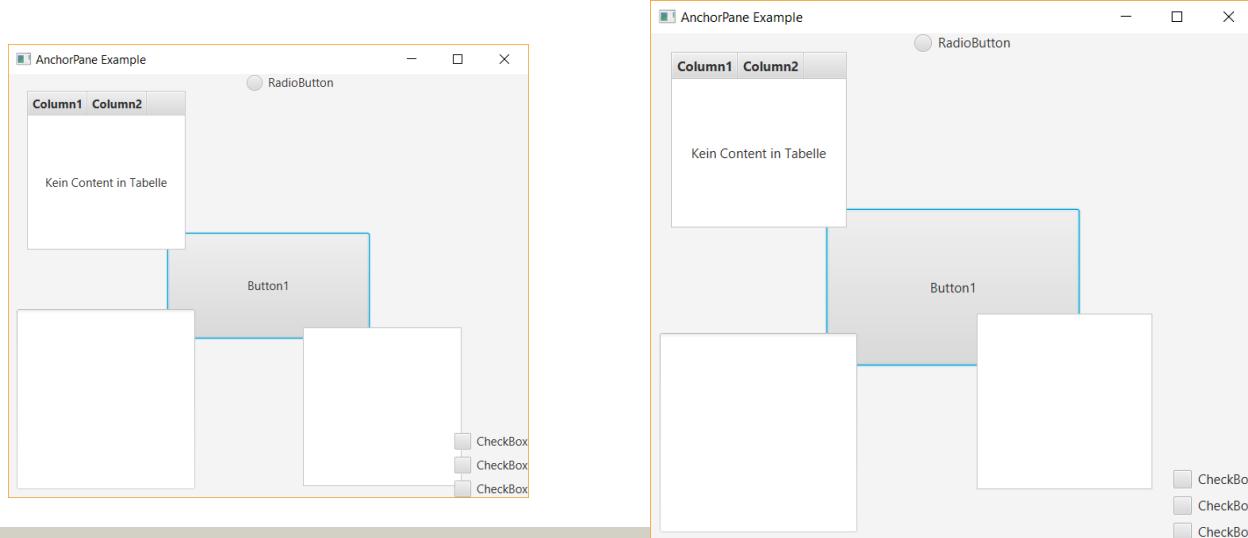
# TilePane / FlowPane

- die Komponenten werden von links nach rechts und nach unten auffüllend angeordnet
- TilePane: Jede Komponente hat so viel Platz wie die größte
- FlowPane: Jede Komponente bekommt nur so viel Platz, wie sie braucht.



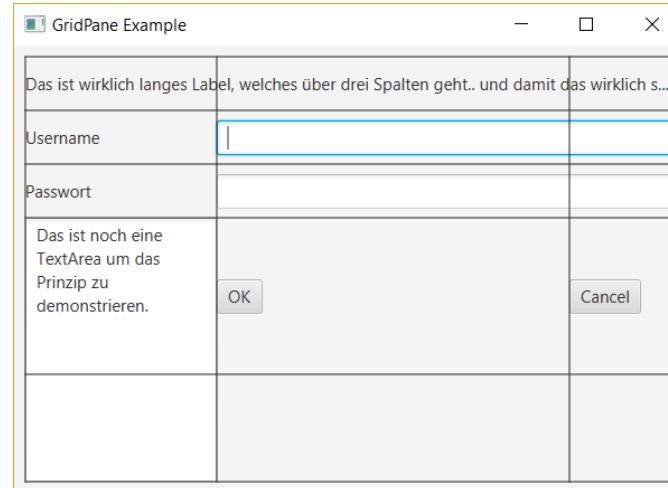
# AnchorPane

- absolute Positionierung/Größe oder
- relative Positionierung mit Größenanpassung zu umgebenden Rändern



# GridPane

- Gitternetz (Grid) mit Angaben für jede Komponente in welche Zelle sie kommt
- Überspannung mehrere Zeilen oder Spalten möglich.



# Layouting - Anmerkungen

- Verschachtelung der Panes möglich
- gute Kombination nicht einfach
- Normalerweise Controls automatisch so groß, das Text sichtbar ist (auch abhängig von Font)
- MinSize            Vorgabe  
PrefSize          meist berechnet  
MaxSize          Vorgabe oder gleich PrefSize
- kein Resize:    MinSize = PrefSize = MaxSize
- Buttonanpassung an Umgebung → setMaxSize auf Double.MAX\_VALUE

# Layouting - Anmerkungen

- weitere spezielle Panes:
  - ButtonBar (mehrere gleich große Buttons nebeneinander)
  - Accordion (Auf- und Zuklappen von Bereichen)
  - Toolbar
  - TabbedPane
  - SplitPane
  - ScrollPane
  - teilweise auch von Drittherstellern

# Aligning (Ausrichtung)

- bei manchen Panes vorgegeben (HBox, VBox, ...)
- bei anderen Angabe über setAlignment(Pos)
- Pos ist Enum mit Konstanten
  - CENTER\_LEFT
  - CENTER
  - CENTER\_RIGHT
  - TOP\_LEFT
  - TOP\_CENTER
  - TOP\_RIGHT
  - ...

# Lernziele

- Kurzeinführung Graphische Benutzeroberflächen und JavaFX
- Widgets
- Layout