LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME UND DATA MINING

Einführung in JavaFX

SEP 2018

Tobias Lingelmann

2018-04-17

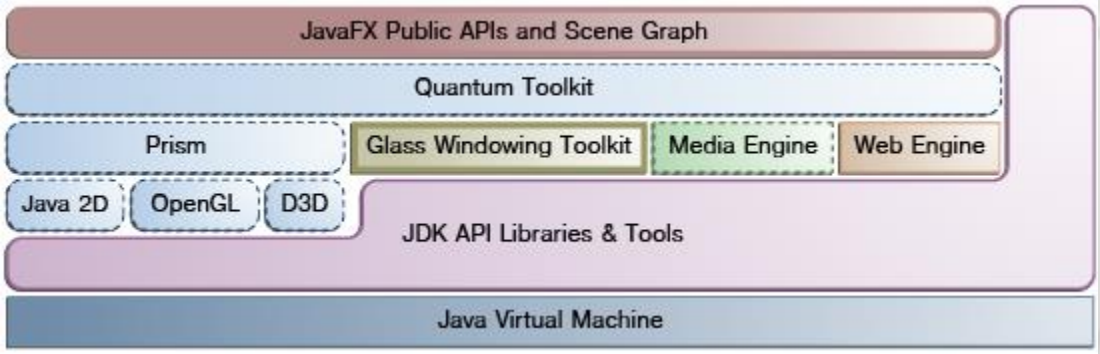
DBS
Wissenschaftliche Betreuer:
Daniel Kaltenthaler, Johannes Lohrer
Verantwortlicher Professor:
Prof. Dr. Peer Kröger

Einführung in JavaFX

- Grundlagen
- Aufbau einer JavaFX-Anwendung
- Kontrollelemente
- Layout
- Weiterführende Themen
  - Styling mit CSS
  - Zeichnen auf einem Canvas
- Events

Grundlagen

- Von Oracle entwickeltes objektorientiertes Framework für GUIs in Java
- Version 1.0 veröffentlicht in 2008
- Integriert in JDK seit JDK 7 Update 6
- Wird mit Java 11 wieder aus dem JDK entfernt.
- Architektur:



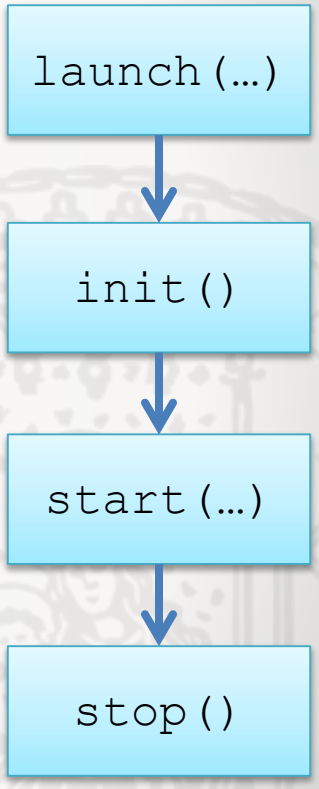
Quelle: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-architecture.htm

JavaFX-Anwendung

- Erben von der abstrakten Klasse Application
- Beim Starten wird ein erstes Fenster erstellt
- Anwendung wird (normalerweise) implizit mit dem Schließen des letzten Fensters beendet
- Änderungen an den gezeigten Elementen nur aus dem JavaFX-Thread heraus!

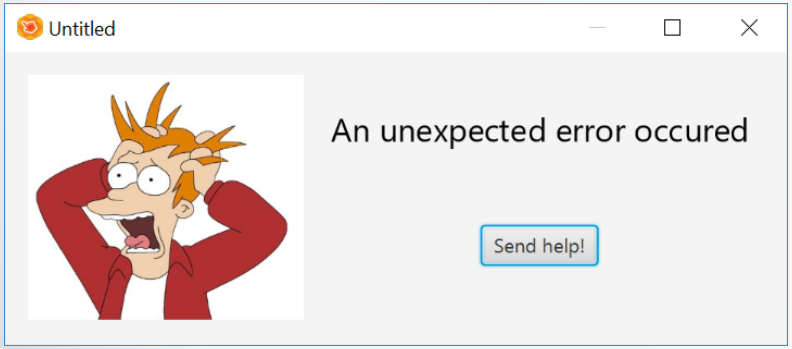
JavaFX-Anwendung

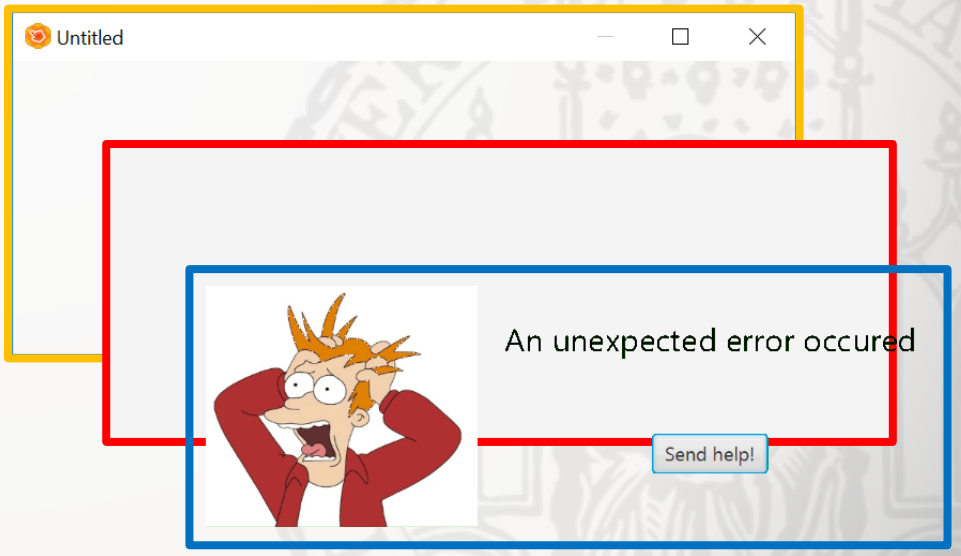
- `launch(String[])`
  - Ruft `init()`, `start(…)` und (nach Programmende) `stop()` auf.
  - Funktioniert gut ohne unsere Hilfe.
- `init()`
  - Zunächst leer.
  - Muss nicht überschrieben werden.
- `start(Stage)`
  - Dieser Methode wird das erste Fenster übergeben.
  - Muss überschrieben werden!
- `stop()`
  - Zunächst leer.
  - Wird ausgeführt, wenn Programm mit `Platform.exit()` beendet wird.
  - Muss nicht überschrieben werden.



JavaFX-Anwendung

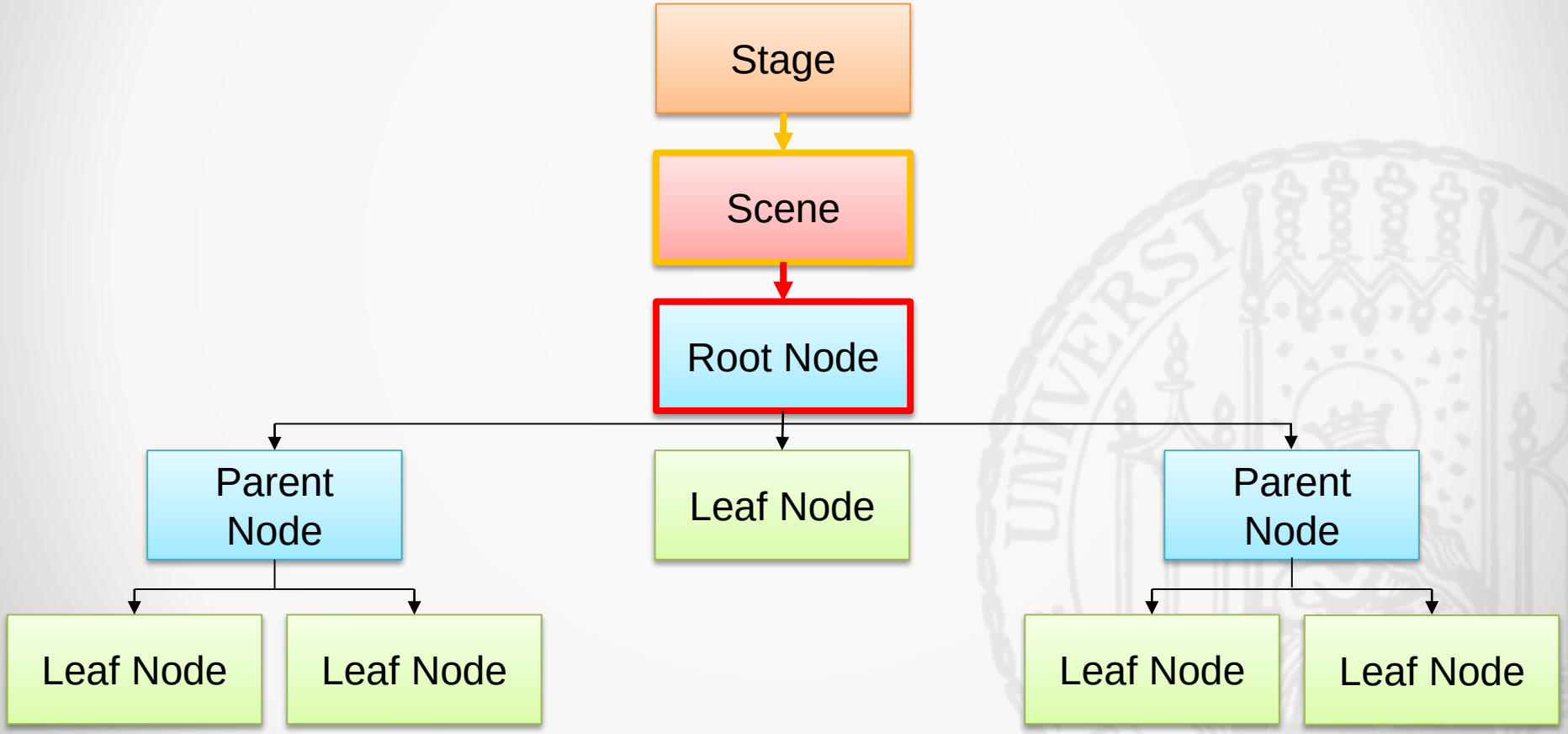
- Stage := „Fenster“
- Scene := „Fensterinhalt“
- Nodes := Objekte in der Scene





JavaFX-Anwendung

JavaFX Scene Graph



JavaFX-Anwendung

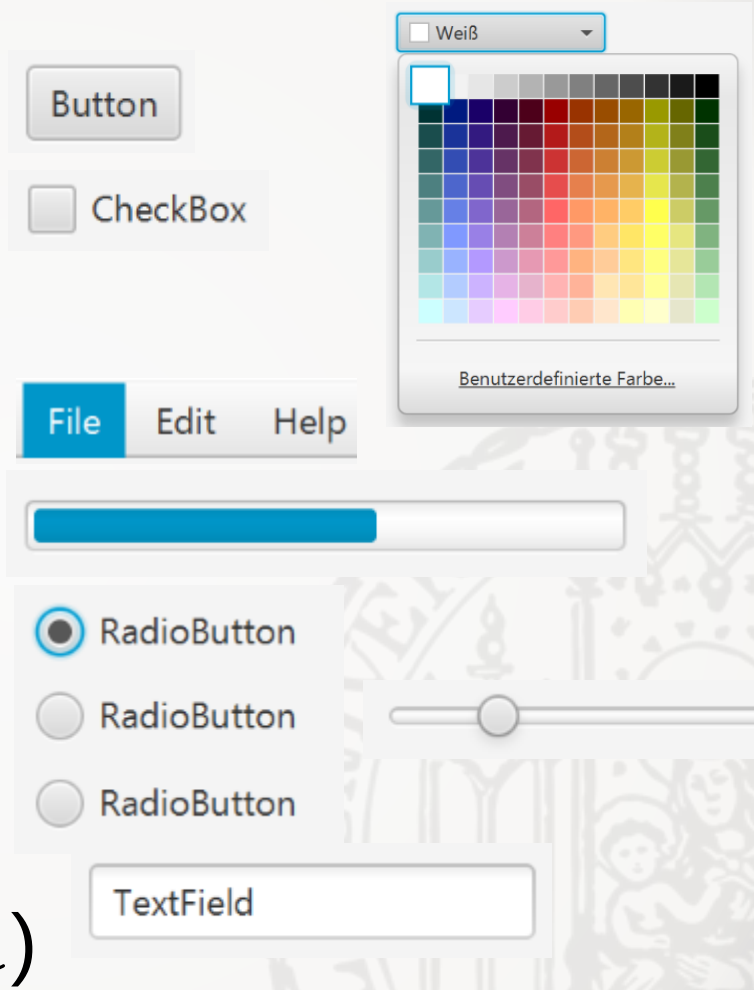
JavaFX Scene Graph:

- Beschreibt die Struktur der Elemente einer `Scene`
- Ist vorwärtsgerichtet und zyklenfrei
- Enthält Wurzelknoten (Typ Parent)
- Parent Nodes können ein oder mehrere Kinder haben, die selber auch Parent Nodes sein können. (`Group`, `Region`)
- Alle Zweige im Scene Graph sind Parent Nodes
- Leaf Nodes enthalten selber keine Kinder mehr.
- Aktiver Scene Graph darf nur durch den JavaFX Application Thread modifiziert werden!

Noch nicht genug? https://docs.oracle.com/javafx/2/scenegraph/jfxpub-scenegraph.htm

Kontrollelemente

- Button
- CheckBox
- ColorPicker
- MenuBar
- ProgressBar
- RadioButton
- Slider
- TextField/Area
- Dialogfenster (`Alert`)
- Und viele andere (Tipp: Scenebuilder)

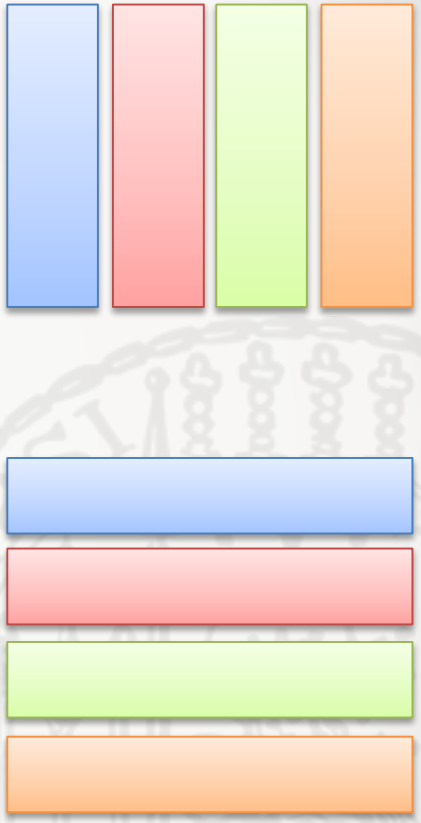


Und natürlich auch hier: https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/ui_controls.htm

Layout in JavaFX

HBox und VBox

- Horizontale Anordnung (nebeneinander)
- Vertikale Anordnung (untereinander)
- Hinzufügen über `.getChildren().add()`



Javadoc: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/HBox.html
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/VBox.html

Layout in JavaFX

GridPane

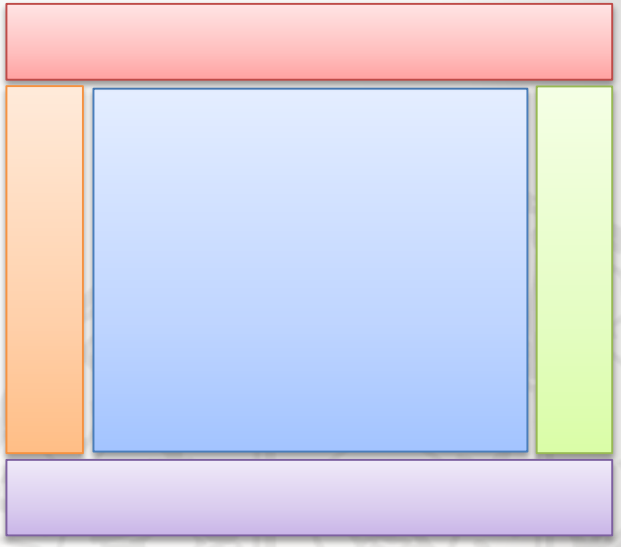
- Anordnung in tabellarischer Form möglich
- Hinzufügen über `.add(Node, spaltenIndex, zeilenIndex)`

Javadoc: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/GridPane.html

Layout in JavaFX

BorderPane

- 5 bestimmte Regionen: top, bottom, left, right, center
- Hinzufügen über z.Bsp. `.setCenter(Node)`

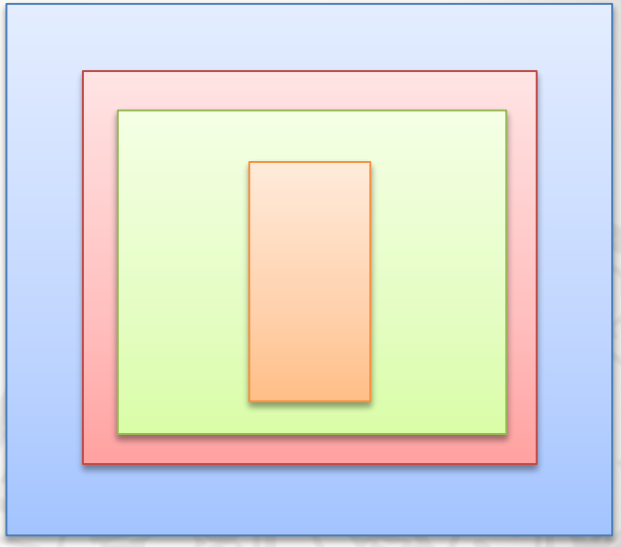


Javadoc: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/BorderPane.html

Layout in JavaFX

StackPane

- Nodes sind automatisch zentriert und werden übereinander gelegt.
- Hinzufügen über `.getChildren().add()`

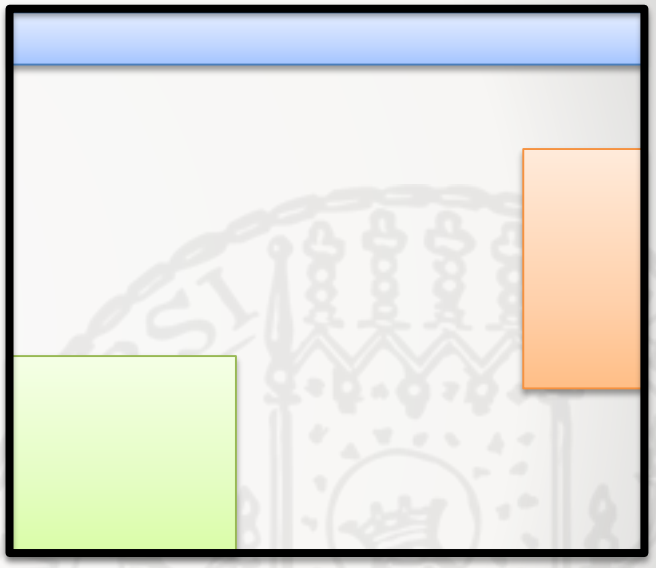


Javadoc: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/StackPane.html

Layout in JavaFX

AnchorPane

- Nodes können an (mehreren) Rändern verankert werden.
- Hinzufügen über `.getChildren().add()`
- Verankern über z.Bsp. `.setTopAnchor(Node, dist)`

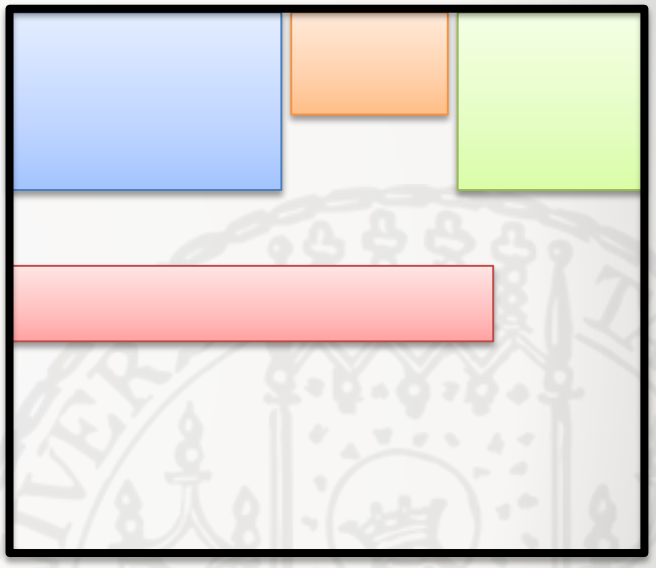


Javadoc: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/AnchorPane.html

Layout in JavaFX

FlowPane

- Nodes werden automatisch in eine neue Reihe/Spalte gesetzt, wenn vorgegebene Breite/Höhe erreicht
- Hinzufügen über `.getChildren().add()`

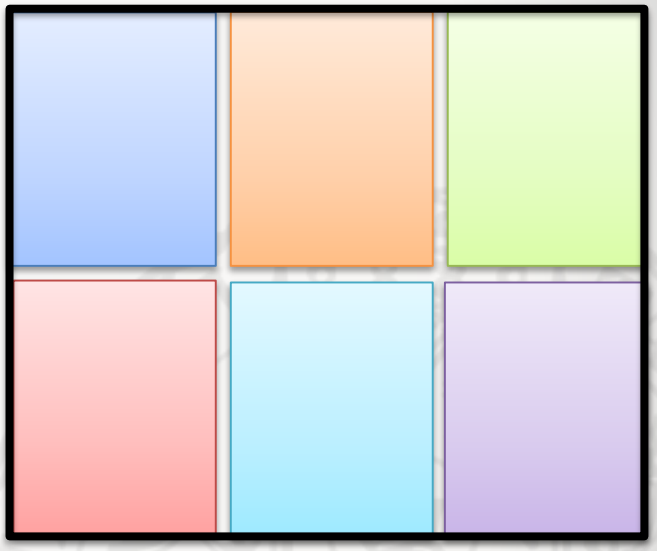


Javadoc: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/FlowPane.html

Layout in JavaFX

TilePane

- Jeder Node bekommt genau die gleiche Fläche zur Verfügung gestellt.
- Hinzufügen über `.getChildren().add()`



Javadoc: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/TilePane.html

Layout in JavaFX

- Inhalte von Panes (und anderen Parents) lassen sich per .setAlignment() ausrichten.
- Abstände zwischen Elementen meist anpassbar (spacing, gap)
- Bei einer BorderPane müssen nicht alle 5 Regionen genutzt werden.
- Schachtelung mehrerer Panes nicht unüblich!

Mehr Infos (insbesondere zum Anpassen): https://docs.oracle.com/javafx/2/layout/jfxpub-layout.htm

Weiterführende Themen

Styling mit CSS

- Parents, Scenes und SubScenes können Stylesheets zugewiesen werden.
- Zuweisen von „Style-Klassen“-Namen mit .getStyleClass().add(String)
- Inline-Styles mit .setStyle(String) (unschön)
- Beispiel im SEPStartProject ☺

Alle Details: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html

Weiterführende Themen

Zeichnen auf einem `Canvas`

- Prinzipiell ein manipulierbares Bild
- Nicht objekt- sondern pixelbasierend
- Zeichnen mittels `Canvas.getGraphicsContext2D()`
- Unter Umständen performanter als tausende Objekte, aber Verzicht auf viele Features von Shapes.
- Beispiel: Siehe Anfangsprojekt auf der Website

Mehr Infos: https://docs.oracle.com/javase/8/javafx/graphics-tutorial/canvas.htm

Events

- Physikalische Events
  - `InputEvent` extends `Event`
  - `MouseEvent` extends `InputEvent`
  - `ScrollEvent` extends `InputEvent`
  - `KeyEvent` extends `InputEvent`
- Logische Events
  - `ActionEvent` extends `Event`
  - `WindowEvent` extends `Event`

(Liste nicht vollständig! Siehe auch https://docs.oracle.com/javafx/2/events/convenience_methods.htm)

Events

Ablauf Ereignisbehandlung

- Zielauswahl (Aktueller Fokus; Mauszeiger)
- Wegwahl durch SceneGraph (vom root/Stage zum targetNode)
- „Event Capturing“: Event wird vom root zum targetNode geschickt. (→ EventFilter)
- „Event Bubbling“: Event kehrt zurück zum root (→ EventHandler)
- Aufhalten/Abfangen durch `.consume()`

Events

Eigenschaften eines Events

- Event-Typ
  - Beispiel: `KeyEvent.KEY_PRESSED`
  - Hierarchische Ordnung
- Quelle („source“)
  - Node auf dem das Event derzeit behandelt wird
  - Ändert sich!
- Ziel („target“)
  - Node am Ende der Ereigniskette

Events

Beispiele

- Handler:
  - `.setOnKeyPressed(e -> … )`
  - `.addEventHandler(KeyEvent.KEY_PRESSED, e -> …)`
  - Natürlich auch mit fxml!
- Filter:
  - `.addEventFilter(KeyEvent.ANY, e -> …)`

Mehr Infos: https://docs.oracle.com/javafx/2/events/jfxpub-events.htm

Linksammlung

- Siehe auch auf den einzelnen Folien ☺
- Oracles Sammlung: https://docs.oracle.com/javase/8/javase-clienttechnologies.htm
- Scenebuilder: http://gluonhq.com/products/scene-builder/
- Tutorial zu JavaFX inkl. Scenebuilder: http://code.makery.ch/library/javafx-8-tutorial/
- Lauter kurze Codebeispiele, u.a. zu EventFilter: http://www.java2s.com/Tutorials/Java/JavaFX/1120__JavaFX_Event_Filter.htm