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

# Einführung in JavaFX

SEP Sommersemester 2019

Nicolas Brauner

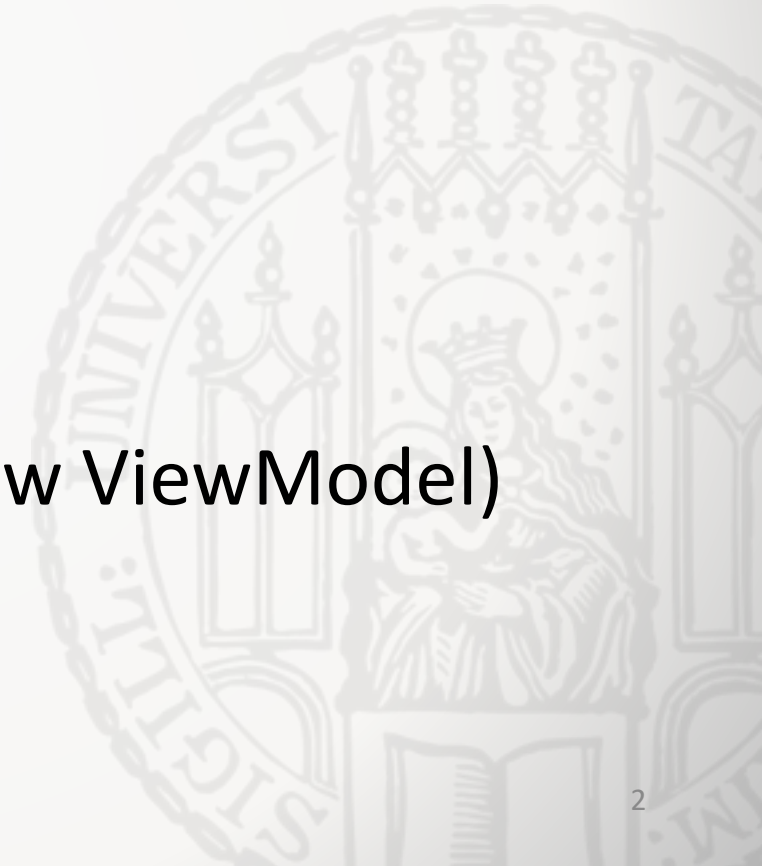
23.04.2019

DBS

Wissenschaftlicher Betreuer:
Maximilian Hünemörder, Ludwig Zellner
Verantwortlicher Professor:
**Prof. Dr. Peer Kröger**

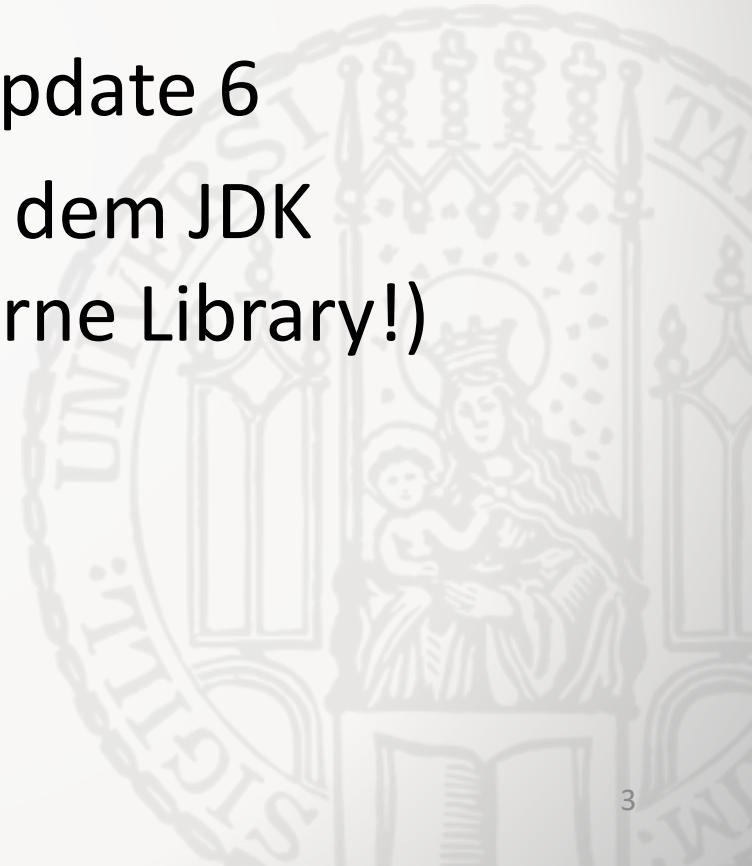
# Übersicht

- Grundlagen
- Aufbau einer JavaFX-Anwendung
- Kontrollelemente
- Layout
- Styling mit CSS
- Events
- Ausblick: MVVM (Model View ViewModel)



# 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. (Einbinden als externe Library!)



# JavaFX-Anwendung

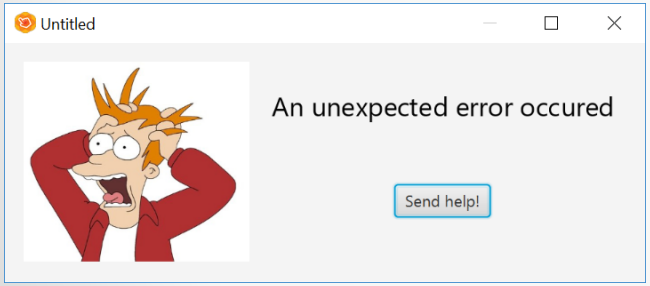
- Erben von der abstrakten Klasse Application
- Beim Starten wird ein erstes Fenster erstellt
- Anwendung wird (normalerweise) implizit mit dem Schließen des letzten Fensters beendet
- Änderungen an den gezeigten Elementen nur aus dem JavaFX-Thread heraus!

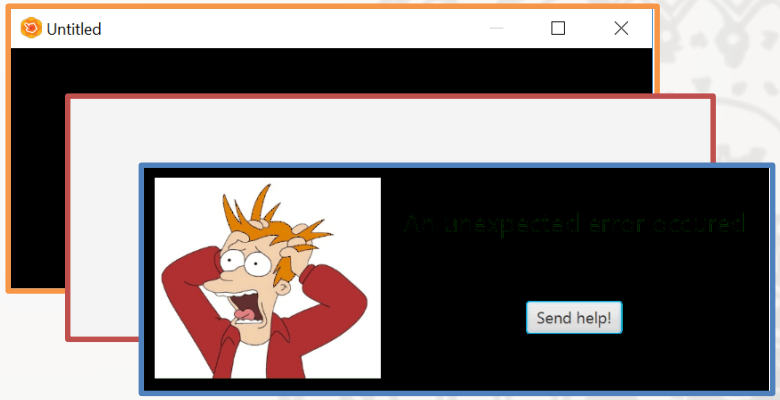
# JavaFX-Anwendung

- 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

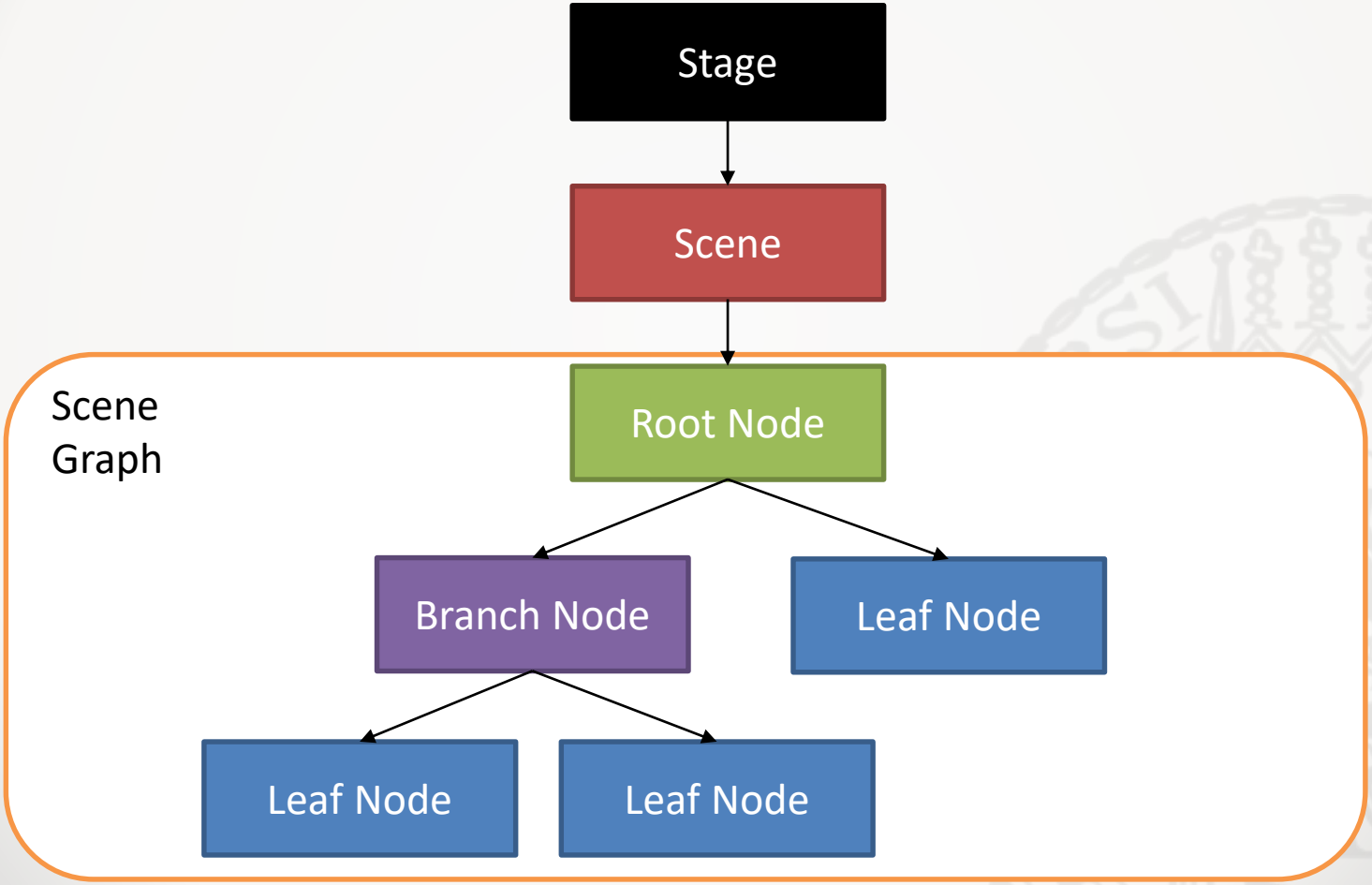
- Stage := „Fenster“
- Scene := „Fensterinhalt“
- Nodes := Objekte in der Scene



# 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
- LeafNodes enthalten selber keine Kinder mehr.
- Aktiver Scene Graph darf nur durch den JavaFX Application Thread modifiziert werden! (Bei Multithreading: **Platform.runLater(…)**)

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

# Layout in JavaFX

- Es gibt verschiedene Objekte, in denen sich andere JavaFX Nodes anordnen lassen:
  - **HBox** (horizontale Anordnung nebeneinander)
  - **VBox** (vertikale Anordnung untereinander)
  - Panes:
    - **GridPane** (Anordnung in tabellarischer Form und über Indizes)
    - **BorderPane** (5 regionen: top, bottom, left, right, center)
    - **TilePane** (Jeder Node bekommt die gleiche Fläche)

    …
  - Die meisten Objekte lassen sich über **.getChildren().add()** hinzufügen. (Für genaueres siehe in den jeweiligen Dokumentationen)

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

# 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)

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

# 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

Eigenschaften eines Events:

- Event-Typ
  - Beispiel: KeyEvent.KEY_PRESSED
  - Hierarchische Ordnung
- Ereigniskette von Quelle zum Ziel der Bearbeitung durch Hierarchien des JavaFX Scene Graphs (Stage → source node)
  - Quelle („source“)
    - Ort an dem das Event generiert wird (z.B. Maus)
  - Ziel („target“)
    - Node am Ende der Ereigniskette (z.B. Button)

# 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

# MVVM (Model View ViewModel)

- Trennung zwischen Ansicht, Daten und deren Verarbeitung
- Genaueres nächste Woche!