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

Einführung in JavaFX

SEP

Felix Zenz

16.10.2017

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

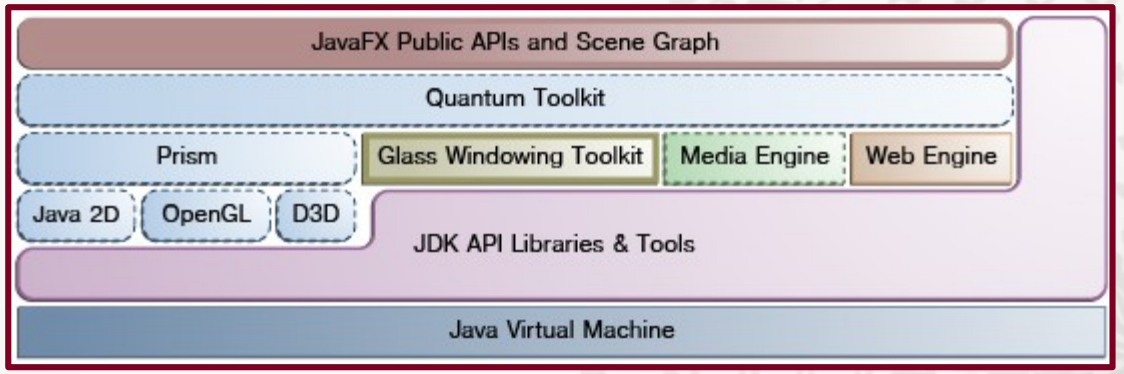
DBS

Einführung in JavaFX

- Grundlegendes
- Aufbau einer JavaFX Application
  - Application Class
  - Stage & Scene
  - Scene Graph
- Kontrollelemente
- Layout
- Weiterführende Themen
  - Canvas
  - CSS Styling

Grundlegendes

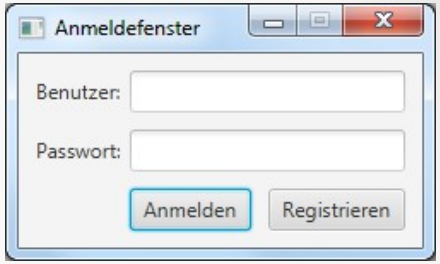
- JavaFX ist ein von Oracle entwickeltes Framework zur Erstellung von GUIs in Java
- Die Version JavaFx 8 wird von der JDK 8 unterstützt, die Bibliothek ist integriert
- JavaFX ist eine Neuentwicklung nach AWT und Swing
- Architektur:



Oracle

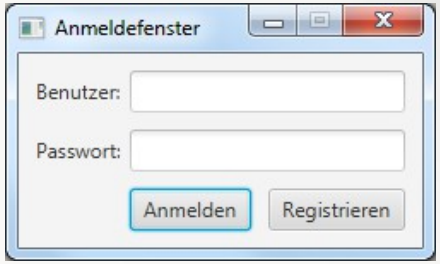
Aufbau einer JavaFX Application

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

public class LoginApp extends Application {


            public static void main (String[] args){
                        Launch(args);
            }

            @Override
            public void start(Stage primaryStage) throws Exception {

                        LoginView loginContent = new LoginView(); //selbstgeschriebene Klasse, enthält Scene
                        primaryStage.setScene(loginContent.getScene());

                        primaryStage.setTitle("Anmeldefenster");
                        primaryStage.sizeToScene(); //die Stage soll die Größe der Scene annehmen
                        primaryStage.show();

            }

}
```



Aufbau einer JavaFX Application

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

public class LoginApp extends Application {


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

        @Override
        public void start(Stage primaryStage) throws Exception {

                LoginView loginContent = new LoginView(); //selbstgeschriebene Klasse, enthält Scene
                primaryStage.setScene(loginContent.getScene());

                primaryStage.setTitle("Anmeldefenster");
                primaryStage.sizeToScene(); //die Stage soll die Größe der Scene annehmen
                primaryStage.show();

        }

}
```

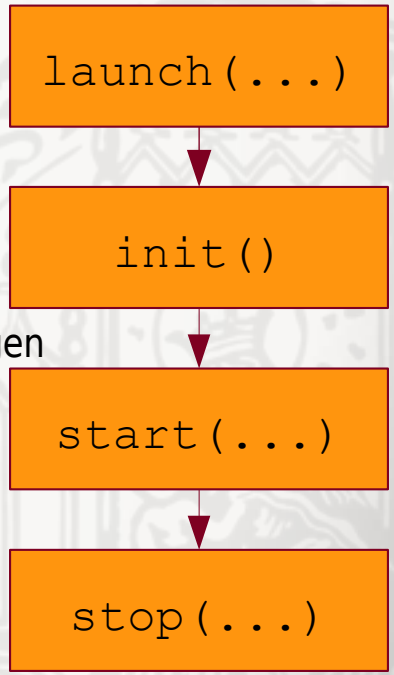


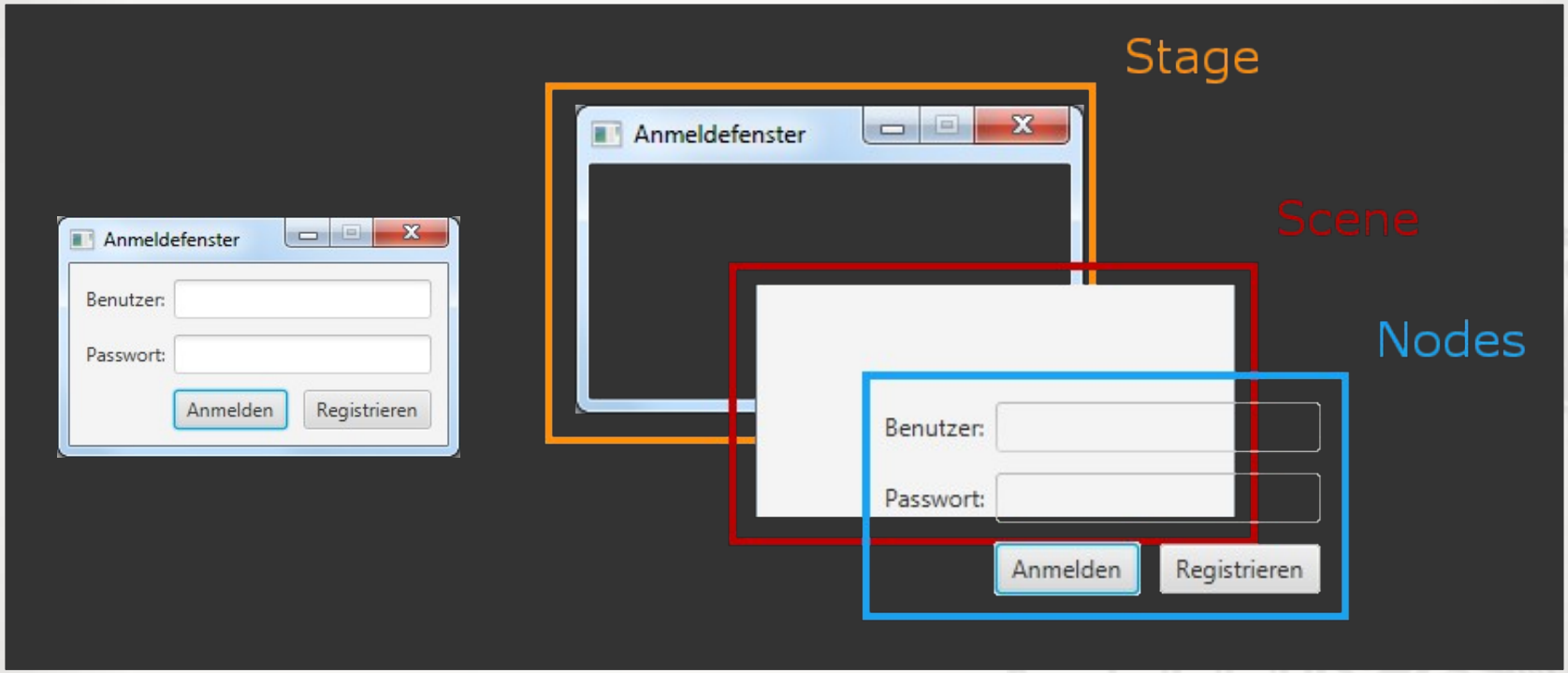
JavaFX Application

Die Application Klasse enthält u.a. die Methoden:

- **`launch(String[])`**
  - Erhält die Programm Parameter
  - Führt erst `init()` dann `start(...)` aus
  - wartet dann auf das Programmende und führt `stop()` aus
  - Sollte nicht überschrieben werden
- **`init()`**
  - Ist zunächst leer
  - Muss nicht überschrieben werden
- **`start(javafx.stage.Stage)`**
  - Wird verwendet um das primäre Anwendungsfenster zu erzeugen
  - Ist abstrakt → Muss überschrieben werden
- **`stop()`**
  - Wird typischer Weise zur Freigabe von Ressourcen verwendet
  - Ist zunächst leer
  - Wird beim beenden der Anwendung automatisch ausgeführt
  - Muss nicht überschrieben werden

Lebenszyklus einer JavaFX Application:



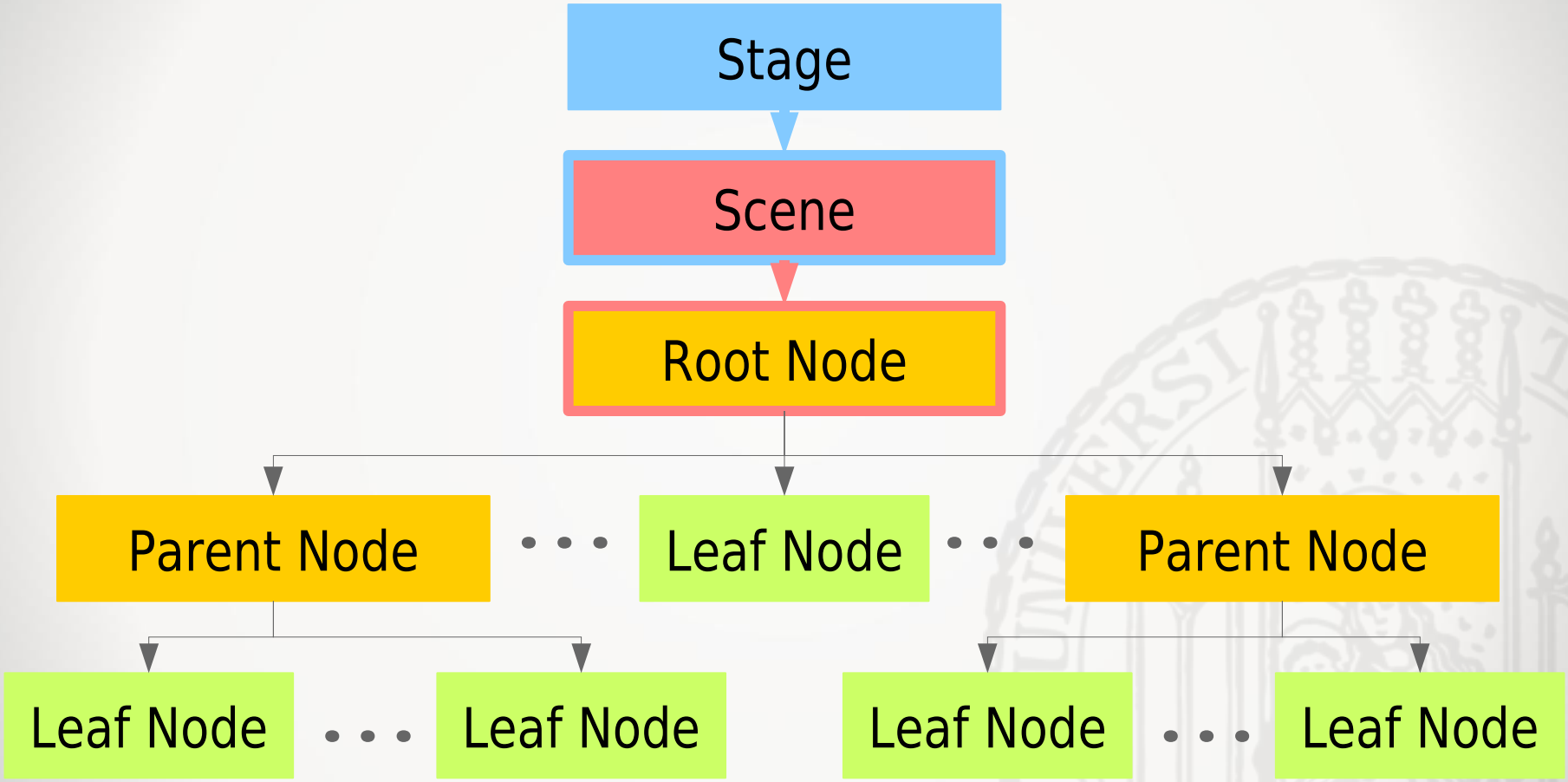

Stage
Anmeldefenster
Scene
Nodes
Benutzer:
Passwort:
Anmelden
Registrieren
Anmeldefenster
Benutzer:
Passwort:
Anmelden
Registrieren

JavaFX Stage & Scene

- Eine `Stage` ist im Prinzip ein Fenster, das als „Bühne“ für unseren Inhalt dient
- Eine `Stage` wird mit Aufruf der Methode `show()` sichtbar
- Zugriff auf einige interne Werte der Stage sind möglich

  z.B.: `Stage.setTitle(String);` oder `Stage.setResizable(Boolean);`
- In der `Stage` enthalten ist ein Objekt der Klasse `Scene`
- Die `Scene` „Szene“ enthält alle Objekte, die in der `Stage` gerendert werden sollen, sowie interne Werte, auf die zugegriffen werden kann
- Dokumentationen: https://docs.oracle.com/javase/8/javafx/api/javafx/stage/Stage.html
  https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Scene.html

JavaFX Scene Graph

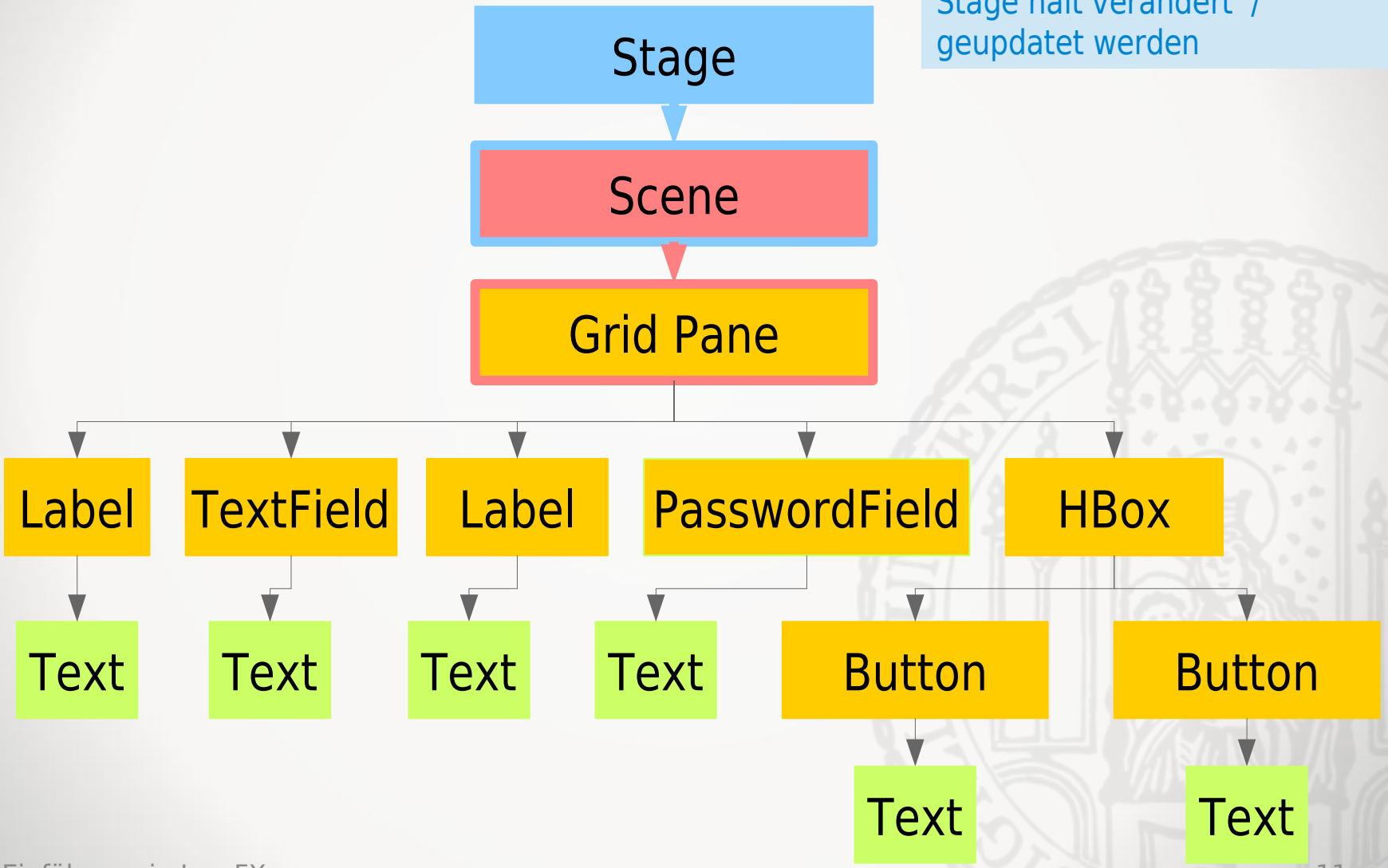


JavaFX Scene Graph

- Ist der Graph, der die Struktur der Elemente einer `Scene` beschreibt
- Ist vorwärtsgerichtet und zyklenfrei
- Enthält einen Wurzelknoten (**Root Node**), von dem aus die anderen Knoten des Graphen erreicht werden
- Der Wurzelknoten ist vom Typ **Parent**
- **Parent Nodes** können ein oder mehrere Kinder haben, die selbst auch Parent Nodes sein können. → Container, etc.
- Alle Zweige im Scene Graph sind Parent Nodes
- **Leaf Nodes** sind Blätter, d.h. sie haben selbst keine Kinder → Formen, Text, Bilder, Mediendateien, etc.

Beispiel: LoginView

Elemente im SceneGraph können nur durch den Application Thread, der ihre Stage hält verändert / geupdatet werden



Nodes

- Primitive Elemente
  - Shape: Rectangle, Polygon, Circle, etc.
  - Text
  - ImageView (zeigt eine Bilddatei an)
  - Canvas
  - ...
- Kontrollelemente
- Container

Nodes können nicht nur gerendert werden, alle Nodes lassen sich auch bewegen, skalieren, drehen oder scheren. Außerdem lassen sich auf Nodes visuelle Effekte anwenden.

Kontrollelemente in JavaFX

Button

- Kann auf Mausklicks mittels eines EventHandlers reagieren
- Kann ImageView (Bild), Text oder eine Kombination daraus anzeigen

```
@Override
public void start(Stage primaryStage){

                VBox box = new VBox();
                box.setAlignment(Pos.CENTER);

                ImageView imageViewBig = new ImageView("download.png");
                Button pictureButton = new Button();
                pictureButton.setGraphic(imageViewBig);

                Button textButton = new Button("Download");

                ImageView imageViewSmall = new ImageView("download.png");
                imageViewSmall.setFitWidth(16);
                imageViewSmall.setFitHeight(16);

                Button bothButton = new Button("Download", imageViewSmall);


                box.getChildren().addAll(pictureButton, textButton, bothButton);

                primaryStage.setScene(new Scene(box));
                primaryStage.setTitle("ButtonBeispiel");
                primaryStage.show();

}
```

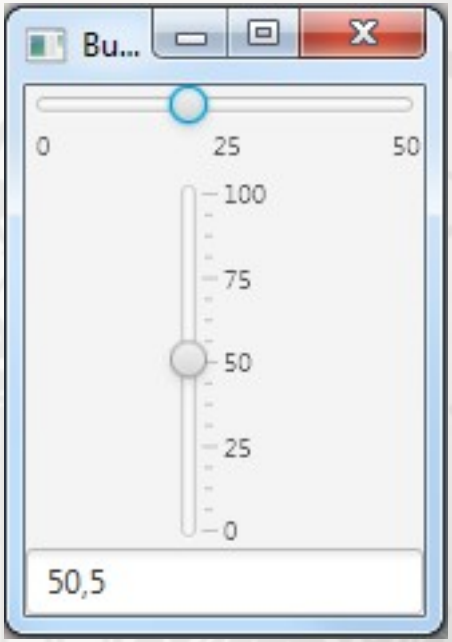
Aus Platzmangel auf den Folien ist der Aufbau der Scene in die start() Methode gequetscht. Was hier der Übersichtlichkeit dient, wäre in einem echten Programm nicht modular, schwer zu warten und kein schöner Programmierstil.

Kontrollelemente in JavaFX

Slider

- Einheit aus Schieberegler und einer Skala
- Kann entweder stufenlos oder in Schritten verschoben werden

```
@Override
public void start(Stage primaryStage){
                VBox box = new VBox();
                box.setAlignment(Pos.CENTER);

                Slider horizontalSlider = new Slider();
                horizontalSlider.setMin(0);
                horizontalSlider.setMax(50);
                horizontalSlider.setValue(20);
                horizontalSlider.setShowTickLabels(true);

                Slider verticalSlider = new Slider();
                verticalSlider.setOrientation(Orientation.VERTICAL);
                verticalSlider.setMin(0);
                verticalSlider.setMax(100);
                verticalSlider.setValue(50.5);
                verticalSlider.setShowTickLabels(true);
                verticalSlider.setShowTickMarks(true);

                TextField field = new TextField(""+verticalSlider.getValue());

                 field.textProperty().bindBidirectional(verticalSlider.valueProperty(),
                                                        NumberFormat.getNumberInstance());

                box.getChildren().addAll(horizontalSlider, verticalSlider, field); ...
}
```

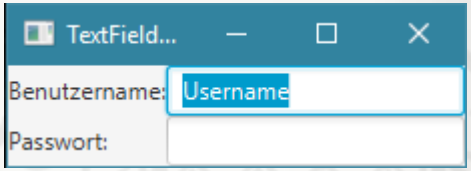


Kontrollelemente in JavaFX

TextField

- Feld, das eine Eingabe als Wert speichert
- Kann bereits mit einem Text initiiert werden
- Ein PasswordField erfüllt die gleiche Funktion, aber maskiert die Eingabe

```
@Override
public void start(Stage primaryStage){

                GridPane grid = new GridPane();

                TextField userField = new TextField("Username");
                PasswordField passwordField = new PasswordField();

                Label userLabel = new Label("Benutzername:");
                Label passwordLabel = new Label("Passwort:");

                grid.add(userLabel, 0, 0);
                grid.add(passwordLabel, 0, 1);
                grid.add(userField, 1, 0);
                grid.add(passwordField, 1, 1);

                primaryStage.setScene(new Scene(grid));
                primaryStage.setTitle("TextFieldBeispiel");
                primaryStage.show();

}
```

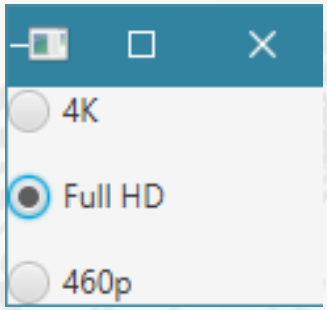


Kontrollelemente in JavaFX

Radio Button

- Runder Button, der ausgewählt sein und wieder abgewählt werden kann
- Lässt sich in ToogleGroups zusammenfassen

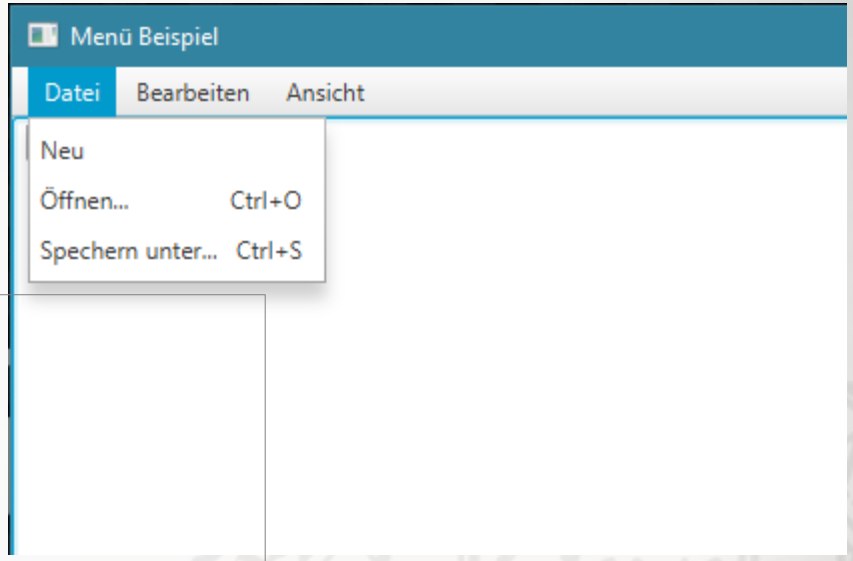
```
@Override
public void start(Stage primaryStage){

                VBox box = new VBox();
                box.setSpacing(15);

                ToggleGroup resolutionSelection = new ToggleGroup();

                RadioButton lowResButton = new RadioButton("460p");
                RadioButton highResButton = new RadioButton("Full HD");
                RadioButton fourkButton = new RadioButton("4K");

                lowResButton.setToggleGroup(resolutionSelection);
                highResButton.setToggleGroup(resolutionSelection);
                fourkButton.setToggleGroup(resolutionSelection);

                highResButton.setSelected(true);

                box.getChildren().add(fourkButton);
                box.getChildren().addAll(highResButton, lowResButton);

                primaryStage.setScene(new Scene(box));
                primaryStage.setTitle("RadioButton Beispiel");
                primaryStage.show();

}
```



Kontrollelemente in JavaFX

Menu

- Ein Menü, das aus einer Menüleiste (MenuBar) besteht, die Untermenüs enthalten kann
- Sobald ein MenuItem gewählt wurde, klappt das Untermenü wieder ein

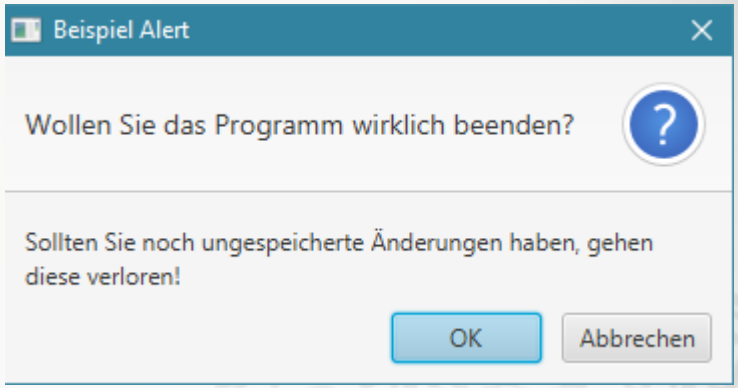



```
@Override
public void start(Stage primaryStage){

                VBox box = new VBox();

                TextField field = new TextField();
                field.setPrefSize(640, 400);
                field.setAlignment(Pos.TOP_LEFT);

                MenuBar menuBar = new MenuBar();
                Menu menuFile = new Menu("Datei");
                Menu menuEdit = new Menu("Bearbeiten");
                Menu menuView = new Menu("Ansicht");

                MenuItem itemNew = new MenuItem("Neu");
                MenuItem itemOpen = new MenuItem("Öffnen...");
                itemOpen.setAccelerator(KeyCombination.keyCombination("Ctrl+O"));
                MenuItem itemSaveAs = new MenuItem("Spechern unter...");
                itemSaveAs.setAccelerator(KeyCombination.keyCombination("Ctrl+S"));

                menuFile.getItems().addAll(itemNew, itemOpen, itemSaveAs);
                menuBar.getMenus().addAll(menuFile, menuEdit, menuView);
                box.getChildren().addAll(menuBar, field);
                ...

}
```

Kontrollelemente in JavaFX

Alert

- Ein eigenes Dialogfenster, das verschiedene Typen besitzen kann: Error, Warning, etc.
- Je nach Typ ändert sich der Inhalt (Buttons, etc.)




```
@Override
public void start(Stage primaryStage){

            Alert alert = new Alert(AlertType.CONFIRMATION);
            alert.setTitle("Beispiel Alert");
            alert.setHeaderText("Wollen Sie das Programm wirklich beenden?");
            alert.setContentText("Sollten Sie noch ungespeicherte Änderungen“
                        +" haben, gehen diese verloren!");

            alert.showAndWait();
}
```

Kontrollelemente in JavaFX

Weitere

Oracle

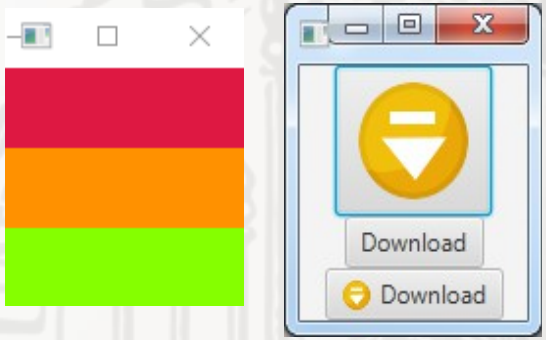
Layout in JavaFX

HBox & VBox

- HBox und VBox sind Panes, d.h. Container, die selbst nicht sichtbar sind (Glasscheiben), aber Nodes enthalten
- Mit .getChildren() erhält man die Liste der enthaltenen Elemente einer VBox/HBox
- Mit .getChildren.add(Node) oder .getChildren.addAll(Node,Node,...) fügt man folglich neue Elemente hinzu
- Die VBox ordnet Elemente der Reihe nach vertikal an, die HBox horizontal
- Mit .setSpacing(double) lässt sich der Abstand zwischen den Elementen definieren
- Implementierungsbeispiele finden sich in vorangegangenen Beispielen zu Kontrollelementen

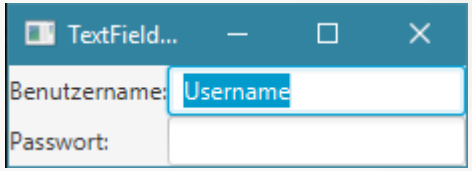
HBox

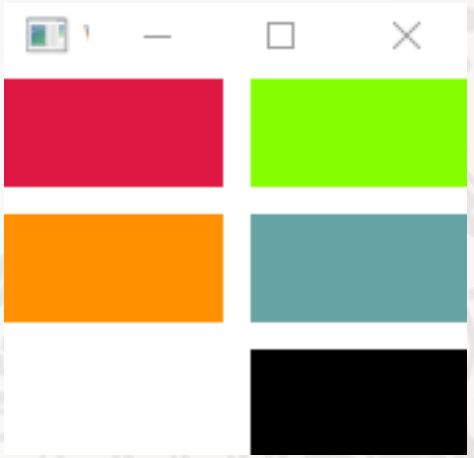
VBox



Layout in JavaFX
GridPane

- Pane, die eine Anordnung in tabellarischer Form ermöglicht
- Nodes werden wie folgt eingefügt: .add(Node, spaltenIndex, zeilenIndex)
- Wobei die Indizes int Werte sind, die bei 0 beginnen
- Die Abstände zwischen den Elementen lassen sich mit .setHgap(double) bzw. .setVgap(double) definieren
- Ein Implementierungsbeispiel findet sich im vorangegangenen Beispiel zu TextField Kontrollelementen

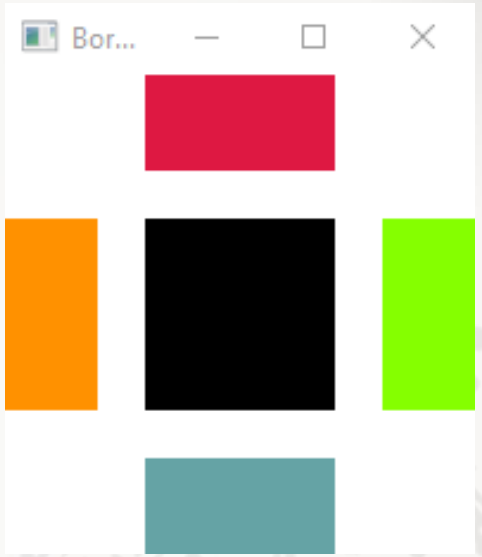




Layout in JavaFX

BorderPane

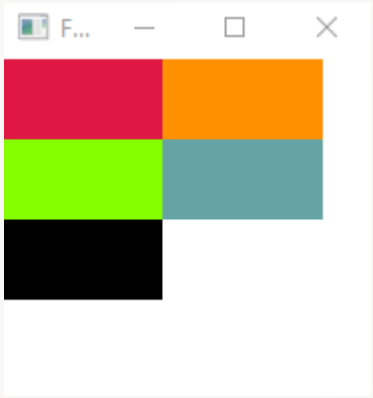
- Pane, die 5 Sektionen hat: Top, Bottom, Left, Right, Center
- Nodes werden z.B. so eingefügt: .setCenter(Node)



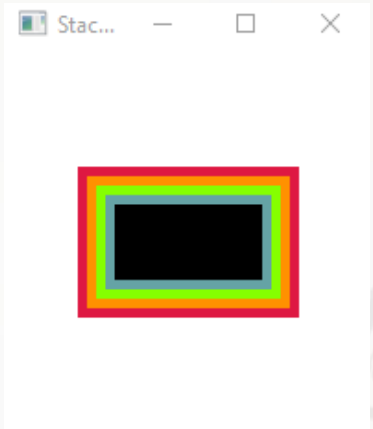

```
...
border.setTop(content1);
border.setLeft(content2);
border.setRight(content3);
border.setBottom(content4);
border.setCenter(content5);

BorderPane.setAlignment(content1, Pos.CENTER);
BorderPane.setAlignment(content2, Pos.CENTER);
BorderPane.setAlignment(content3, Pos.CENTER);
BorderPane.setAlignment(content4, Pos.CENTER);
BorderPane.setAlignment(content5, Pos.CENTER);
...
```

Layout in JavaFX

Weitere Layouts



FlowPane



StackPane

...

Layout in JavaFX

Alignment

- Die Inhalte von Panes und anderen Parents lassen sich mittels der Methode .setAlignment(Pos) positionieren
- Pos Werte sind z.B.:

  Pos.CENTER, Pos.BASELINE_RIGHT, Pos.BOTTOM_LEFT, etc
- Hilfreiche Informationen zu Alignment und Size in JavaFX:

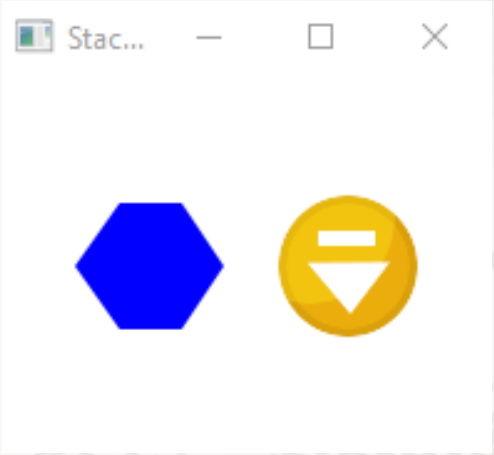
  http://docs.oracle.com/javase/8/javafx/layout-tutorial/size_align.htm#CJHBCBJB

Weiterführende Themen

Canvas API

- Die Canvas ist im Kern ein manipulierbares Bild. Sie ist also nicht objekt- sonder pixelbasiert
- Sie besitzt Alpha-Werte, d.h. Sie kann transparente Pixel enthalten

Zeichnen und löschen von Formen, aber auch von Bildern auf der Canvas ist mittels des GraphicsContext2D() möglich



- Weiterführendes:
  http://docs.oracle.com/javase/8/javafx/graphics-tutorial/canvas.htm#JFXGR214

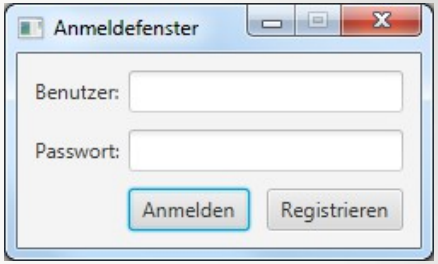
```
Canvas canvas = new Canvas(200,150);

                        canvas.getGraphicsContext2D().setFill(Color.BLUE);

double[] xPoints = {50,75,92,75,50,32};
double[] yPoints = {50,50,75,100,100,75};
canvas.getGraphicsContext2D().fillPolygon(xPoints, yPoints, 6);

canvas.getGraphicsContext2D().drawImage(new Image("download.png"), 110, 43);
```

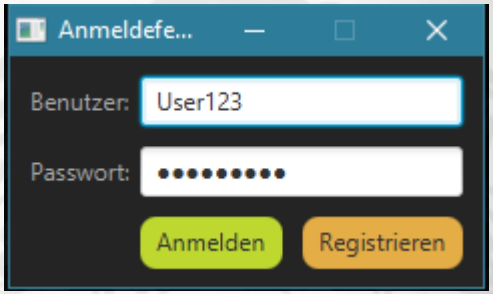
Weiterführende Themen

Styling mit CSS

```
public class LoginView{
...

  private GridPane filledContainer() {

      //Add the fitting CSS styles to the buttons
      confirmButton.getStyleClass().add("button-confirm");
      registerButton.getStyleClass().add("button-register");

      //Configure layout of HBox
      buttonBox.setSpacing(10);

      //Put the Buttons in the HBox
      buttonBox.getChildren().addAll(
              confirmButton,registerButton);

      //Configure layout of GridPane
      containerPane.setPadding(new Insets(10,10,10,10));
      containerPane.setHgap(5);
      containerPane.setVgap(10);

      //Put everything on the GridPane
      containerPane.add(userLabel,0,0);
      containerPane.add(userField,1,0);
      containerPane.add(passwordLabel,0,1);
      containerPane.add(passwordField,1,1);
      containerPane.add(buttonBox,1,2);

      //return the GridPane that contains all elements
      return containerPane;
   }

---
}
```

```
.root{
   -fx-background-color: #242424;
}
.label{
   -fx-text-fill: #AAAAAA;
}
.button{
   -fx-background-radius: 8;
}
.button-confirm{
   -fx-background-radius: 8;
   -fx-background-color: #BED830;
}
.button-register{
   -fx-background-radius: 8;
   -fx-background-color: #e4ad45;
}
```






```
   @Override
   public void start(Stage primaryStage) throws Exception {

      LoginView loginContent = new LoginView();
      primaryStage.setScene(loginContent.getScene());

      PrimaryStage.getScene().getStylesheets().add("LoginStylesheet.css");

      primaryStage.setTitle("Anmeldefenster");
      primaryStage.sizeToScene();
      primaryStage.setResizable(false);
      primaryStage.show();

   }
```

Tutorials und detaillierte Infos:

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