LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

mobile and distributed systems group

Javakurs für Fortgeschrittene

Einheit 04: JavaFX

Kyrill Schmid
Lehrstuhl für Mobile und Verteilte Systeme

Heutige Agenda

Einführung in JavaFX

- Motivation und Eigenschaften
- Hello World in JavaFX
- Komponenten und Szenegraph
- Nutzeraktionen
- GUI gestalten mit CSS

Praxis:

- Log-In Fenster entwerfen
- Log-In Fenster: Button Logik
- Log-In Fenster mit CSS designen
- Hausaufgabe: Bankanwendung in JavaFX schreiben und einbinden

Lernziele

- Grundlagen in JavaFX kennenlernen
- GUIs erstellen und verwenden können
- Konzepte von GUI-Programmierung in Java verstehen

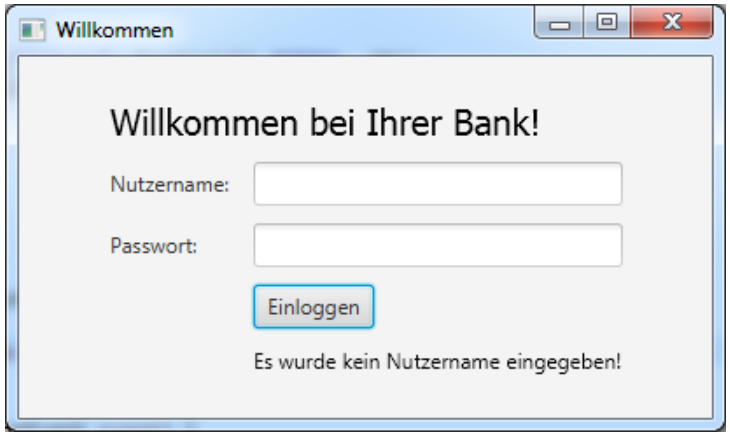
Aufgabe zu JavaFX: Button mit Leben füllen

Füllen Sie nun Ihre zuvor erstellte GUI mit Leben, in dem Sie auf den Einloggen-Button reagieren.

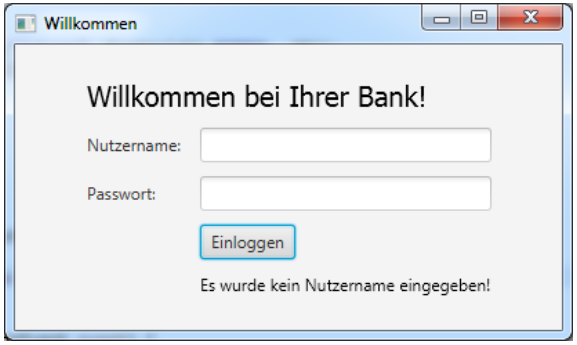
Erstellen Sie zunächst einen Platzhalter für einen Status-Text unterhalb des Einloggen-Buttons.

Reagieren Sie nun wie folgt, wenn der Einloggen-Button gedrückt wurde:

- Falls kein Nutzername eingegeben wurde, erscheint als Status-Text: "Es wurde kein Nutzername eingegeben!„
- Falls kein Passwort Eingegeben wurde, erscheint als Status-Text: "Es wurde kein Passwort eingegeben!„
- Falls Nutzername oder Passwort nicht mit einem von Ihnen akzeptierten Nutzername oder Passwort übereinstimmt, dann geben Sie aus: „Nutzername oder Passwort falsch!“
- Falls beides korrekt, dann geben Sie aus: „Nutzer wird eingeloggt.“



Bisher haben wir mit Standard Design-Elementen gearbeitet!



In Swing mussten die Elemente i.d.R. einzeln designed werden:

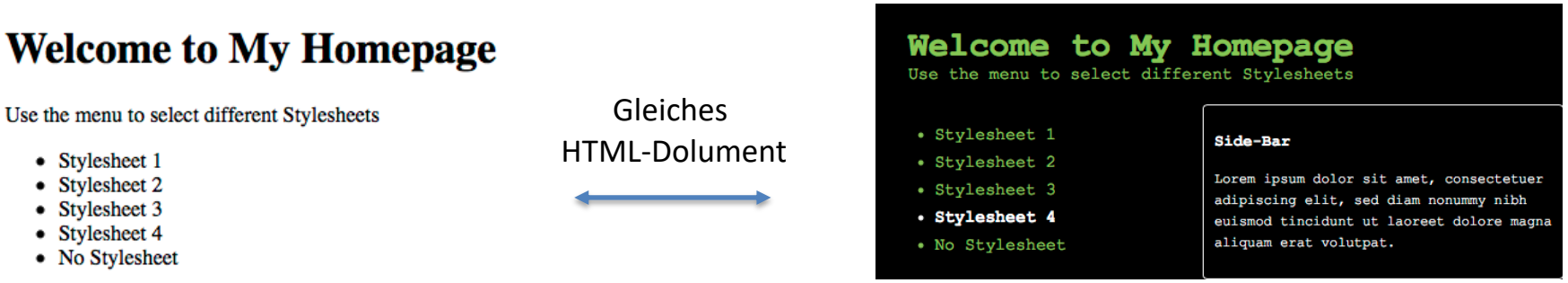
- `myButton.setBackground(new Color…);`
- `myButton.setBorder(…);`

Oder man nutzte open-source Frameworks (Javaxx, Java CSS,…) um cascading style sheets (CSS) mit Swing Komponenten zu verwenden.

In JavaFX ist die Trennung von Inhalt und Layout fest verankert und wird mit *JavaFX CSS* verwirklicht.

- Basiert auf W3C CSS 2.1 und folgt den bekannten Regeln für CSS
- Reference Guide unter: https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

- CSS beschreibt wie HTML-Elemente angezeigt werden sollen
- CCS wird verwendet um Design und Layout für eine Website zu definieren

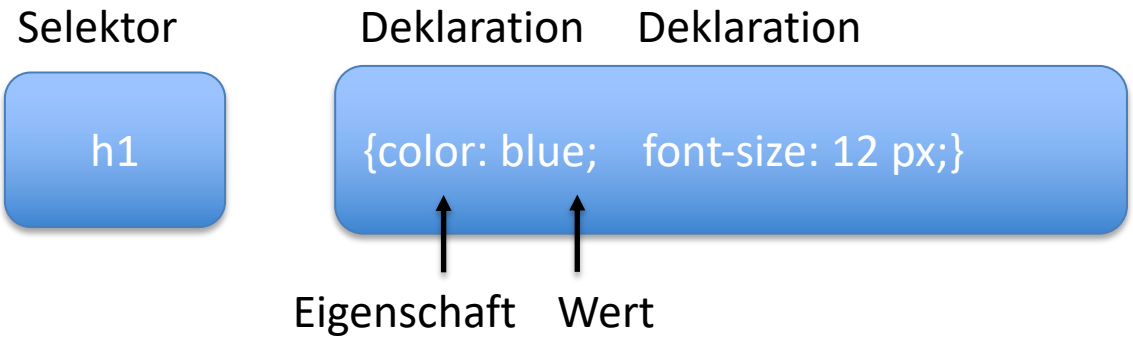


https://www.w3schools.com/css/css_intro.asp

- Warum nicht einfach in das HTML integrieren?
  - HTML war nie dazu gedacht, Tags zur Formatierung einer Website zu beinhalten
  - In HTML 3.2 wurden Tags wie <font> und Farb-Attribute ergänzt
  - Das führte allerdings zu sehr großen HTML-Dokumenten, da alle Informationen in jeder Seite enthalten sein mussten
  - Deshalb entwickelte das World Wide Web Consortium (W3C) CSS
- Die Definition des Styles einer Website erfolgt i.d.R. über ein externes .css-File*Literatur*

CSS Syntax

CSS-Regeln haben folgende allgemeine Syntax:



- Der Selektor gibt das Element an, das gestylt werden soll
- Der Deklarationsblock enthält eine oder mehrere Deklarationen, die durch ein Semikolon getrennt sind
- Jede Deklaration enthält eine CSS-Eigenschaft (Property) und einen WertCSS-Selektoren werden verwendet, um Elemente zu finden basierend auf deren Namen, ID, Klasse, bestimmter Attribute etc.

CSS Selektoren

CSS-Selektoren werden verwendet, um Elemente zu finden basierend auf deren Namen, ID, Klasse, bestimmter Attribute etc.

- **Element-Selektor**: Selektion von Elementen anhand des Namens
  - Beispiel: Selektion aller <p> Elemente:

```
p {text-align: center; color: red;}
```

  - Ergebnis: Alle Absätze (Paragraphs) werden zentriert und rot angezeigt

- **ID-Selektor**: Selektion anhand der ID
  - Verwendung von Hashtag gefolgt von ID:

```
#para1 {text-align: center; color: red;}
```

- **Klassen-Selektor**: Selektion anhand eines bestimmten Klassenattributes
  - Verwendung eines Punktes gefolgt von Klassenname:

```
.center {text-align: center;}
```

Ein paar Unterschiede zu CSS:

- JavaFX Eigenschaften werden mit dem Präfix fx erweitert
- verbietet CSS Layout-Eigenschaften, wie float, position, usw.
- bietet einige Erweiterungen, bspw.: (Hintergrund-)Farben, Ränder, usw.

CSS Styles werden nun für Knoten im JavaFX Szenegraphen verwendet.

- Für das Mapping gelten die bekannten CSS Regeln für Selektoren:
  - **Element-Selektoren:**
    - Analog zu einem Element in HTML
    - I.d.R. einfach der Name der Klasse, bspw.: `Button`, `Text`, usw.
    - Kann abgefragt werden mittels `public String getTypeSelector()`

```
// Beispiel für Typ-Selektor Definition:

Button {
-fx-font-size: 25px;
-fx-font-weight: bold;
-fx-font-style: italic;
-fx-font-family: "Arial Blank"
}
```

- **Klassen-Selektoren:**
  - Jeder Knoten im Szenegraph kann zu einer oder mehreren Klassen gehören (analog zum class-Attribut in HTML)
  - Elemente können mittels `getStyleClass().add(String class)` zu einer Klasse hinzugefügt werden:
    - Bsp.: `myButton.getStyleClass().add("MeineKlasse");`
  - Ansprechbar dann mit bekannter Punkt-Notation:

```
.MeineKlasse{
   -fx-font-weight: bold;
}
```

- **ID- Selektoren:**
  - Jeder Knoten besitzt ein ID-Attribut (analog zu id in HTML)
    - Kann mit `setID(String id)` gesetzt werden.
    - Bsp.: myButton.setId("button1");
  - Ansprechbar dann mit bekannter #-Notation

```
#button1{
   -fx-font-weight: bold;
}
```

Daneben stehen uns auch ein Teil der sog. Pseudoklassen zur Verfügung:

Pseudoklasse	Auswirkung
Focused	Wenn das Element den Fokus erhält
Hover	Wenn der Mouse-Zeiger über dem Element steht
Pressed	Wenn das Element angeklickt wird

```
// Beispiel für Pseudoklassen:

Button:hover {

   -fx-border-width: 0.0 ;
   -fx-font-size: 10px;
   -fx-font-weight: bold;
   -fx-font-style: italic;
   -fx-font-family: "Arial Blank";

}
```

CSS in Code einbinden

2 Arten, um CSS in unsere JavaFX-Applikation einzubinden (analog zu HTML):

- Inline Styles
  - Direkt im Code (Class File)
  - Jeder `Node` verfügt über die Methode `public final void setStyle(String value)`
    - Bsp.: `myButton.setStyle("-fx-font-size: 20px");`
  - Hat höchste Priorität, wäre aber sehr umständlich
- Style sheets
  - Eigene separate CSS Datei.css
  - Wird i.d.R. der Szene mitgeteilt:
    - `scene.getStylesheets().add(getClass().getResource("application.css").toString());`
  - Damit können wir globale Design-Einstellungen für eine Szene definieren.
    - Sehr komfortabel

Aufgabe zu JavaFX: GUI individuell designen

Verwenden Sie nun JavaFX CSS Elemente, um Ihre zuvor erstellte GUI individuell nach Ihren Bedürfnissen zu gestalten.

- Seien Sie dabei etwas kreativ und spielen ein wenig mit den Möglichkeiten, die Ihnen JavaFX CSS bietet.
- Verwende Sie auch einmal inline Styles bzw. verschiedene Style Sheets.
- Ihre GUI könnte sich wie folgt verändern:

Neuer Inhalt im gleichen Fenster

Will man nun den Inhalt in seinem Fenster neu gestalten, bspw. wenn der Nutzer eingeloggt ist, wird einfach eine neue Szene eingesetzt:

- `primaryStage.setScene(new Scene(new NeueSzene()));`
- Kann als separate Klasse Definiert werden

Hausaufgabe:

Nachdem Sie nun bereits ein funktionsfähiges Log-In Fenster gestaltet haben, sollten Sie nun eine Szene in JavaFX für die Bank-Anwendung von letzter Stunde schreiben und bei einem erfolgreichen Log-In diese aufrufen.

Die Funktionen der Buttons sollten dann genauso funktionieren, wie bei der letzten Hausaufgabe gefordert.

Zum Vergleich, dies war die
GUI für die Bankanweisung in Swing =>