Java-Grundlagenschulung
Tag 9

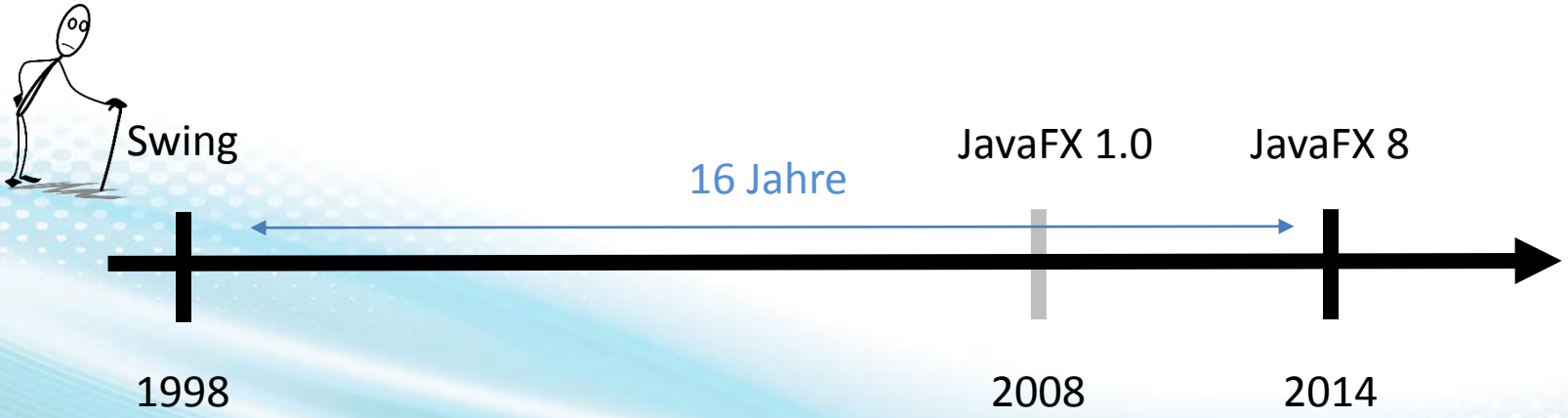
SHD Einzelhandelssoftware GmbH

Agenda

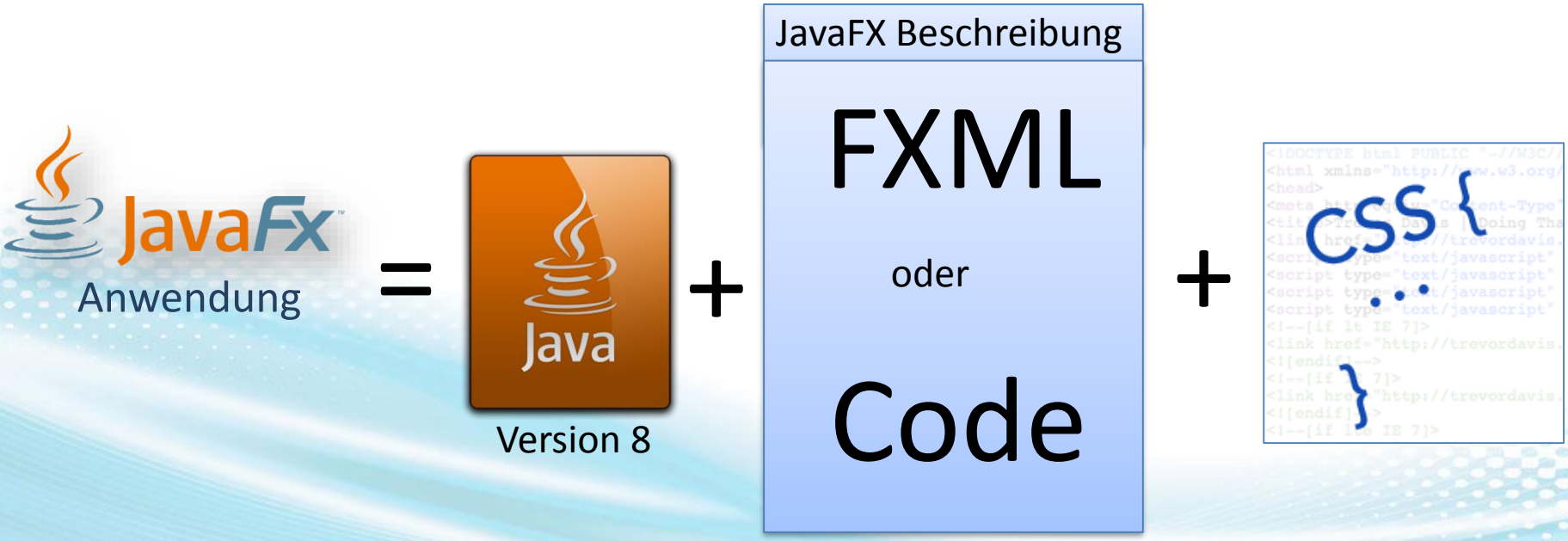
- JavaFX, das neue UI-Framework
- Was brauche ich für JavaFX
- Der Aufbau einer View
- Aufgabe 1 - View erstellen
- Der Aufbau von JavaFX
- Aufgabe 2 - Die Startklasse vorbereiten
- Aufgabe 3 - Das erste Fenster öffnen
- Der ViewController
- Aufgabe 4 - ViewController erstellen
- Aufgabe 5 - Der ActionListener
- Aufgabe 6 - Wechsel zu einer anderen View
- Aufgabe 7 - Styling mit CSS

JavaFX, das neue UI-Framework

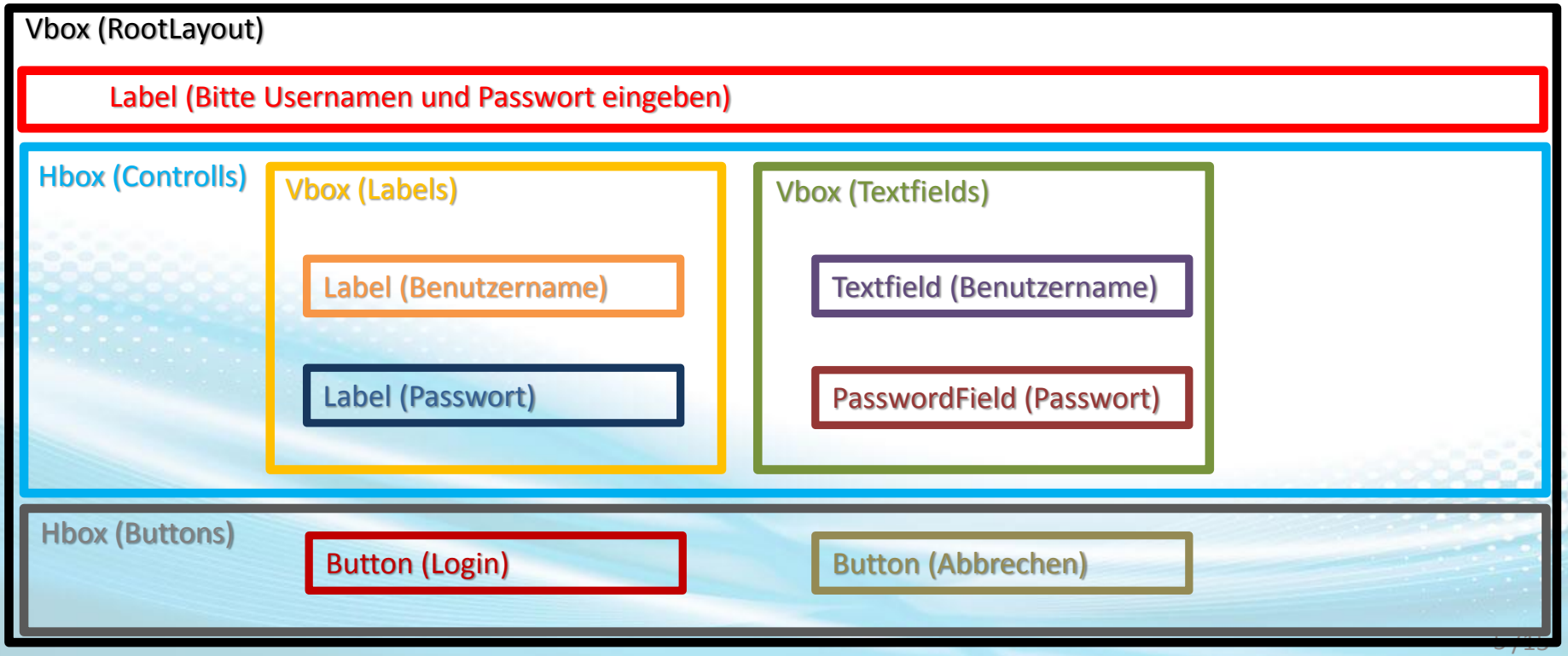
Offizieller Nachfolger von Swing



Was brauche ich für JavaFX



Der Aufbau einer View



Aufgabe 1 - View erstellen

1. Erstelle ein neues Java Projekt in IntelliJ.
2. Erstelle den Ordner resources & markiere ihn als Resources
3. Erstelle im Ordner resources den Unterordner views
4. Erstelle mit dem SceneBuilder das View LoginView.fxml & speichere es im views Ordner
5. Lade die LoginView.fxml im SceneBuilder und erstelle ein Login Formular

Der Aufbau von JavaFX

Stage

Scene 2

Aufgabe 2 - Die Startklasse vorbereiten

- Erstelle eine Main Klasse / Methode
- Die Main Klasse muss von Application erben
- Die main() Methode soll nur die Methode launch(args) von Application aufrufen.
- Überschreibe die start-Methode, welche in Application abstract ist.

Aufgabe 3 - Das erste Fenster öffnen

- Lade die FXML-Resource über den ContextClassLoader:
  Thread.currentThread().getContextClassLoader().getResource("views/MainView.fxml")

- Benutze den FXMLLoader um die FXML-Resource zu laden:
  FXMLLoader loader = new FXMLLoader(resource)

- Lade das RootLayout:
  Parent rootParent = loader.load()

- Erstelle mit dem Basislayout eine neue Scene:
  Scene scene = new Scene(rootParent)

Aufgabe 3 - Das erste Fenster öffnen

- Setze dem primaryStage Parameter der start-Methode die eben erzeugte Scene mit primaryStage.setScene();
- Setze der Stage einen Titel mit primaryStage.setTitle(„Login“);
- Öffne die Stage mit stage.show();
- Starte dein Programm.. =)

Der ViewController

LoginView.fxml

- Infos zum Aufbau der View
- Eigenschaften der Controls (Position,Breite , Name, Id, Aktiv, Style…)

LoginView.java

- Bereitstellung der Controls im JavaCode
- Feldname = Control fx:id
- Enthält relevante UI-Logik

Aufgabe 4 – ViewController erstellen

- Erstelle im src Verzeichnis ein package viewcontroller
- Erzeuge im viewcontroller package die Klasse LoginView
- Vergebe allen Controls auf die zugegriffen werden soll eine fx:id im Scene Builder (Rechts Tab „Code“)
- Hinterlege im SceneBuilder die Controller Klasse viewcontroller.LoginView (Links Tab „Controller“)
- Gehe in IntelliJ und öffne die LoginView.fxml. Erzeuge über alt+Enter die passenden Felder im ViewController anhand der fx:id. Ändere den AccessModifier der Felder von public auf private und annotiere sie mit @FXML

Aufgabe 4 - Der ActionListener

1. Erzeuge in deiner Controller Klasse eine init()-Methode

2. Erstelle in dieser für deine Buttons einen ActionListener.
   verwende die Methode .setOnAction(…); auf dem Button.

3. Implementiere eine beliebige Logik in den Listener-Handler.

4. Hole dir deinen Controller in der MainKlasse mit Hilfe des FXMLLoader
   LoginView loginViewController = loader.getController()

5. Rufe die init-Methode in der Main Klasse auf deinem Controller Objekt auf

Aufgabe 5 – Wechsel zu einer anderen View

1. Erzeuge ein weiteres (Main)View, was deine Hauptapplikation darstellen soll. (z.B. einen Taschenrechner.., halte dich in der Übung aber nicht zulange damit auf damit du die folge Aufgaben schaffst..)

2. Schreibe deinen ActionListener auf dem Login Button dahingehend um, dass dein neues MainView auf der Stage angezeigt werden soll.

3. Prüfe in der Login Methode darauf, ob ein erlaubter User einloggt. Wenn ja geht’s in die MainView, wenn nicht, soll das LoginView eine entsprechende Hinweismeldung bringen. Für Pro´s : Lasse das LoginView sich schütteln

Aufgabe 6 – Styling mit CSS

1. Erzeuge ein Main.css File im package views deiner resources

2. Lade den Style für deine Login Scene mit:
   URL cssResource = Thread.currentThread().getContextClassLoader().getResource("views/Main.css");
   scene.getStylesheets().addAll(cssResource.toExternalForm());

3. Baue in deine .css Datei einen Style für deinen Button ein, welcher bei einem MouseOver den Button Hellblau färbt.

```
.button:hover{
    -fx-color: #3488f1;
}
```

4. Baue in deine .css Datei einen Style ein der die Schriftart auf ComicSans und die Schriftgröße auf 14px ändert.

```
.root {
    -fx-font-family: "Comic Sans MS";
    -fx-font-size: 14px;
}
```