Medientechnik

Sommersemester 2016

Übung 01 (Einführung in Java FX)

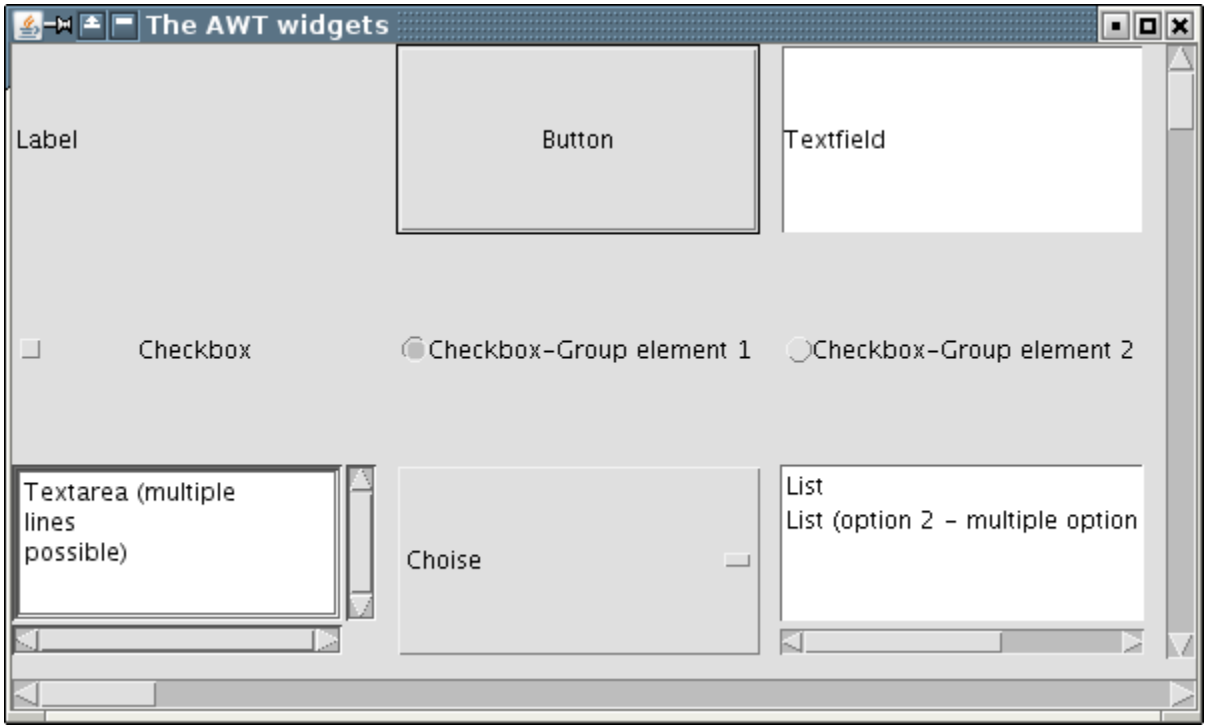
Terminübersicht

Nr	Zeitraum	Thema
1	18.04. - 21.04.	Organisatorisches, Bildbearbeitung
2	**09.05. - 12.05.**	**JavaFX Einführung (GUIs, Szenengraph)**
3	17.05. - 19.05.	Design Patterns: MVC, Observer
4	23.05. - 25.05.	FXML, Bildfilter
5	30.05. - 02.06.	Videobearbeitung
6	06.06. - 09.06.	Video Streams mit JavaFX
7	20.06. - 23.06.	Audiobearbeitung
8	27.06. - 30.06.	Multimedia mit JavaFX

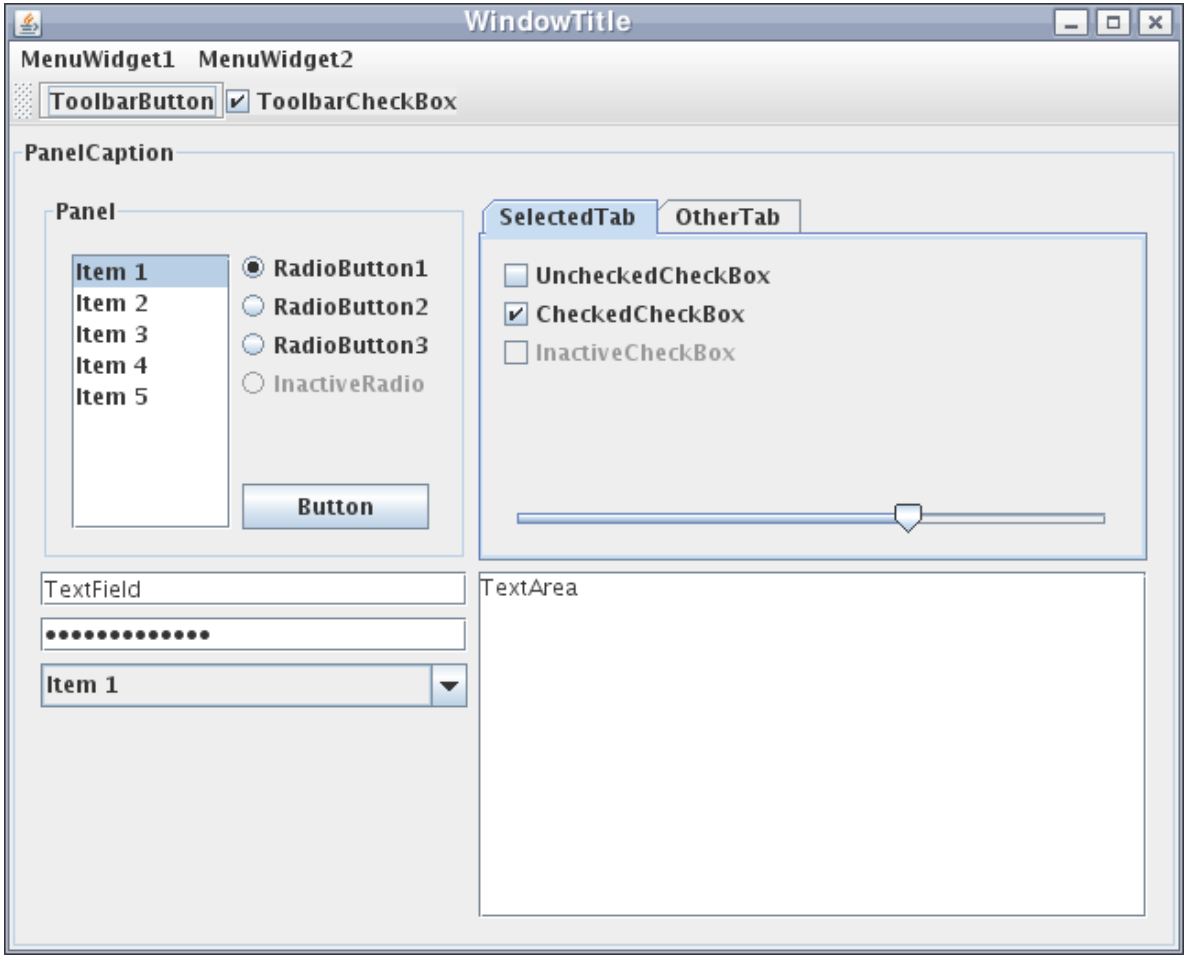
Agenda

- Einführung in JavaFX
  - Application Klasse
  - GUI Elemente
  - Layouting
- Breakout Session
- Wrap-Up Quiz
- Vorbesprechung Übungsblatt 1

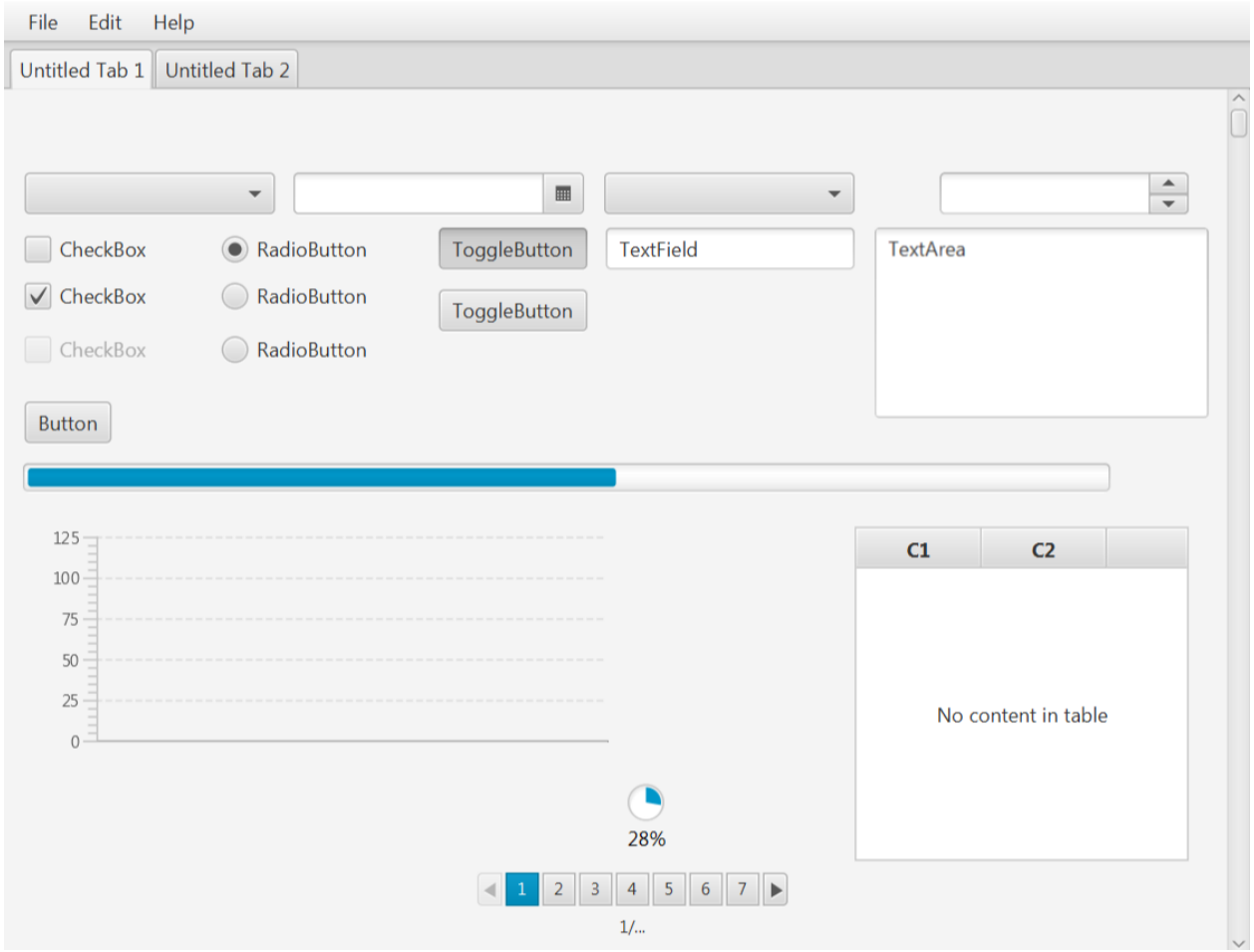
Der Anfang von Java GUIs: AWT



Kurz darauf: Java Swing



Und jetzt: JavaFX!



JavaFX 8 - Features

- FXML
  - Markup Sprache, die auf XML basiert
  - Einfacheres Layouting von GUIs
- CSS für Styling ist sehr einfach
- Moderne Plattformen werden unterstützt, z.B. Multi-Touch
- 3D Viewer
- Mit dem Scene Builder kann man GUIs zusammenklicken
- Aktuelles Java wird unterstützt
- JavaScript Kompatibilität mit “Nashorn”

Anforderungen

- Java Development Kit 1.8 (JDK) enthält JavaFX
  http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Entwicklungsumgebungen:
  - Netbeans (im CIP Pool vorinstalliert)
  - IntelliJ IDEA Community Edition
  - Eclipse
- Dokumentation:
  - Überblick: http://docs.oracle.com/javase/8/javase-clienttechnologies.htm
  - API: http://docs.oracle.com/javase/8/javafx/api/toc.htm

Die Application Klasse

- `public void start(Stage primaryStage) throws Exception`
  - der Ausgangspunkt für alle JavaFX Programme
  - wird automatisch aufgerufen, sobald das Programm bereit ist, die GUI anzuzeigen.

  → diese Methode wird von uns **überschrieben**

- `public static void launch(String... args)`
  - darf nur ein einziges Mal aufgerufen werden
  - wird meistens direkt in der main() Methode **aufgerufen**.
    Falls man eine externe Klasse als GUI starten möchte, geht das so:
    `Application.launch(MyGUI.class,args);`

Beispiel

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

public class HelloWorld extends Application{

    public void start(Stage primaryStage)
            throws Exception {

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

}
```

HelloWorld1.java

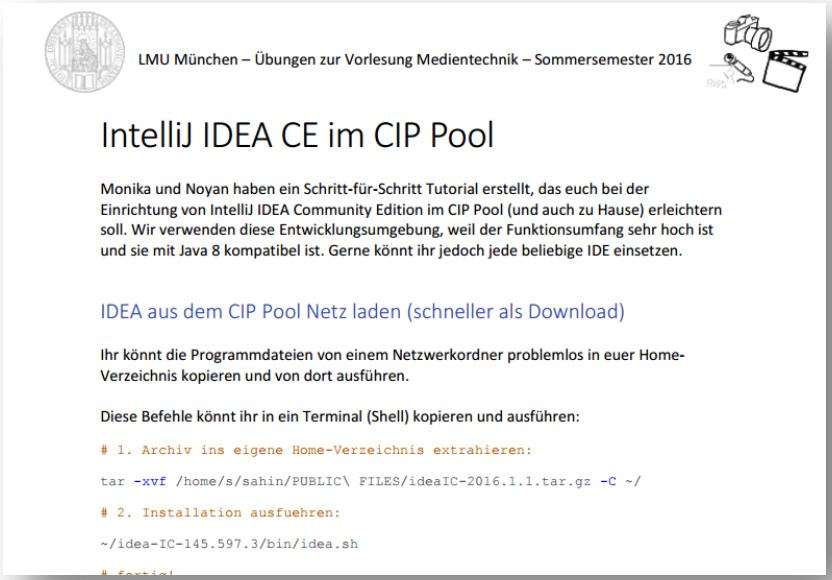
Programm ausführen

- Option 1: Kommandozeile
  - zuerst kompilieren: `javac packageName/Klasse.java`
  - ausführen: `java packageName.Klasse`
  - Unter Windows ist zu beachten: Der Pfad zum JDK muss sich in der Umgebungsvariable “PATH” befinden

- **Option 2: “Run” Dialog von der IDE**
  - Konfiguration notwendig (Klasse mit main-Methode definieren)
  - auf Dauer komfortabler

IntelliJ IDEA Tutorial

- Eure Tutoren Noyan und Monika haben für euch alles vorbereitet: http://www.medien.ifi.lmu.de/lehre/ss16/mt/uebung/ressourcen/mt_help_01.pdf



LMU München – Übungen zur Vorlesung Medientechnik – Sommersemester 2016

IntelliJ IDEA CE im CIP Pool

Monika und Noyan haben ein Schritt-für-Schritt Tutorial erstellt, das euch bei der Einrichtung von IntelliJ IDEA Community Edition im CIP Pool (und auch zu Hause) erleichtern soll. Wir verwenden diese Entwicklungsumgebung, weil der Funktionsumfang sehr hoch ist und sie mit Java 8 kompatibel ist. Gerne könnt ihr jedoch jede beliebige IDE einsetzen.

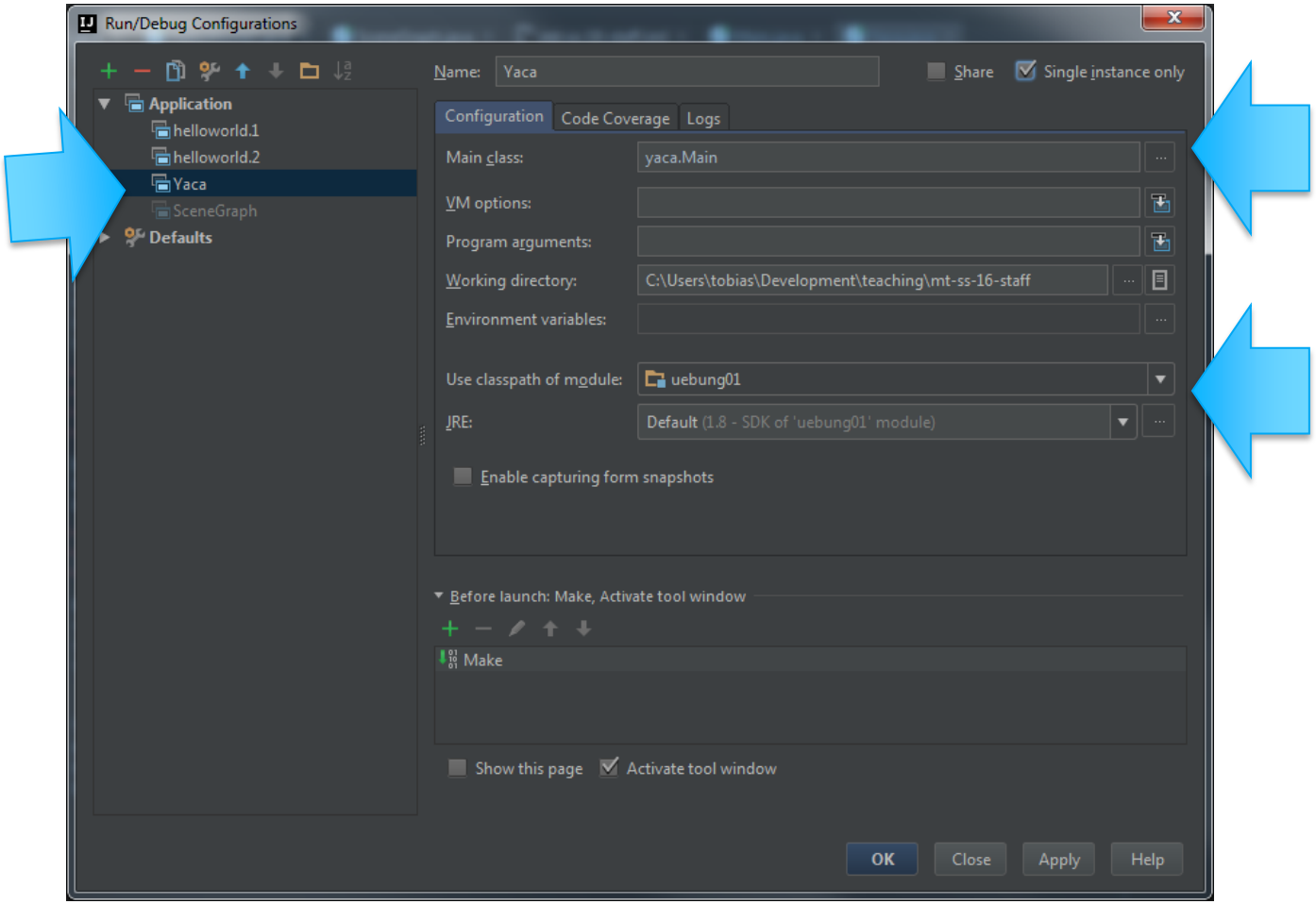
IDEA aus dem CIP Pool Netz laden (schneller als Download)

Ihr könnt die Programmdateien von einem Netzwerkordner problemlos in euer Home-Verzeichnis kopieren und von dort ausführen.

Diese Befehle könnt ihr in ein Terminal (Shell) kopieren und ausführen:

```
# 1. Archiv ins eigene Home-Verzeichnis extrahieren:
tar -xvf /home/s/sahin/PUBLIC\ FILES/ideaIC-2016.1.1.tar.gz -C ~/
# 2. Installation ausfuehren:
~/idea-IC-145.597.3/bin/idea.sh
# fertig!
```

Beispiel Konfiguration in IDEA



HelloWorld - Ergebnis?

- Es ist nichts zu sehen!
- Wir brauchen eine “Szene” für die GUI

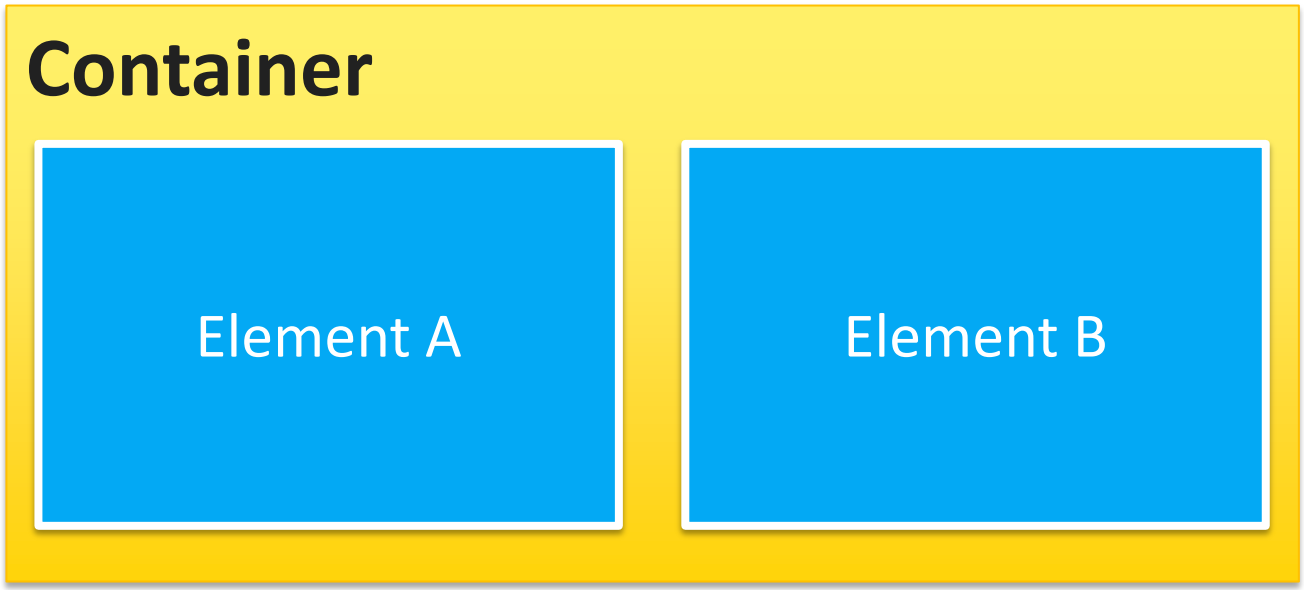
```
C:\Users\tobias\Development\teaching\mt-ss-16-staff\uebung01\src>javac hello\HelloWorld1.java

C:\Users\tobias\Development\teaching\mt-ss-16-staff\uebung01\src>java hello.HelloWorld1
```

Bühne und Szene

- GUI Elemente brauchen eine “Szene”
- Ein Programm hat “Bühnen”, die aus Szenen bestehen, z.B. Hauptbildschirm + Einstellungen
- Wichtigste Klasse: `javafx.scene.Scene;` (Doku)
  - Beispiel Konstruktor:
    ```
    Scene scene = new Scene(root,width,height);
    ```
  - root:
    Container Element, z.B. VBox, FlowPane, GridPane, BorderPane,…

Layout

- Die “Anordnung” der Elemente innerhalb einer Szene wird objektorientiert modelliert
- Es gibt verschiedene Klassen, die als Container für andere Elemente eingesetzt werden können.



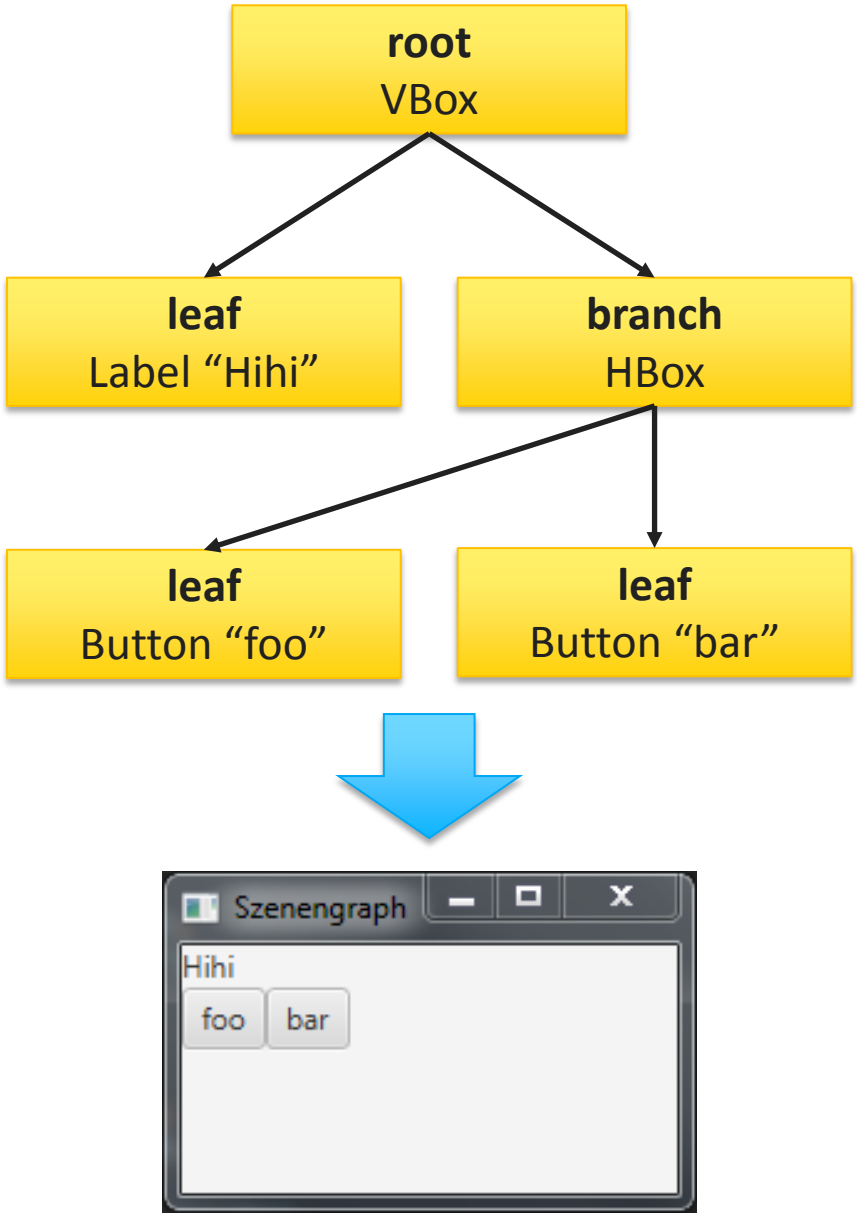
Hello World 2: FlowPane + Scene

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class HelloWorld2 extends Application {
    public void start(Stage primaryStage)
            throws Exception {

        int width = 200, height = 200;

        FlowPane flowPane = new FlowPane();
        Scene scene = new Scene(flowPane, width, height);

        primaryStage.setTitle("Hello World");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args){
        launch(args);
    }
}
```

HelloWorld2.java

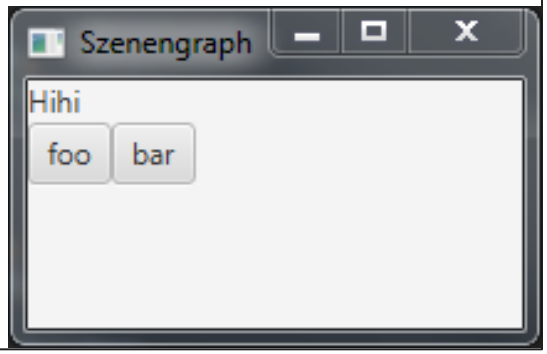
Szenengraph

- Ein Graph besteht aus “Knoten” und “Kanten”.
- Besondere Knoten:
  - **root**: Wurzelknoten
  - **branch**: Zweig
  - **leaf**: Blatt = Ende eines Zweigs
- Die Knoten im Szenengraph sind entweder **GUI Elemente** oder **Container**, die wieder verschachtelt sind



Szenengraph Beispiel

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class SceneGraph extends Application {
    public void start(Stage primaryStage) throws Exception {
        int width = 200, height = 100;

        VBox root = new VBox();
        Label leaf1 = new Label("Hihi");
        HBox branch = new HBox();
        Button leftButton = new Button("foo");
        Button rightButton = new Button("bar");

        root.getChildren().add(leaf1);
        root.getChildren().add(branch);
        branch.getChildren().add(leftButton);
        branch.getChildren().add(rightButton);

        Scene scene = new Scene(root, width, height);
        primaryStage.setTitle("Szenengraph");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args){
        launch(args);
    }
}
```



SceneGraph.java

Break Out: Eingabefelder

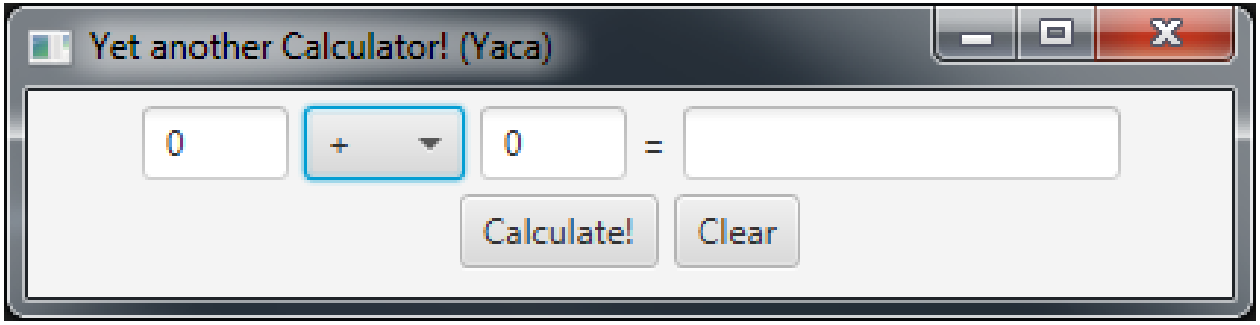
- Am Beispiel “SceneGraph.java” weiterarbeiten
- Material hier downloaden: www.medien.ifi.lmu.de/lehre/ss16/mt/uebung/ressourcen/mt_material01.zip

- Jetzt hinzufügen:
  - Text Eingabefeld
  - Passwort Eingabefeld

- Zeit: 8 Minuten.

Yaca (Yet another Calculator)

- Ziel für heutige Übung: “Taschenrechner GUI”
  - Elemente kennenlernen & einsetzen
  - Anordnung mit Layout Panes
- Nächste Übung:
  - Events
  - MVC
  - Observer



Projekt erstellen

- IDE starten (ggf. workspace festlegen)
- Neues Java(FX) Projekt “MT-Uebung” anlegen
- *[optional]* Modul “uebung01” erstellen.
- 2 neue Klassen erstellen (am besten ohne package)
  - `Yaca.java`: erbt von Application
  - `Main.java`: zum Starten des Programms

Startpunkt des Programms: Main.java

Main.java

```
import javafx.application.Application;

public class Main {
    public static void main(String[] args){
        Application.launch(Yaca.class,args);
    }
}
```

Basis GUI: Yaca.java

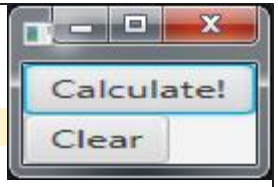
Yaca.java

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;

public class Yaca extends Application {
    public void start(Stage primaryStage) throws Exception {

        BorderPane root = new BorderPane();

        Scene scene = new Scene(root);
        primaryStage.setScene(scene);
        primaryStage.sizeToScene();
        primaryStage.setTitle("Yet another Calculator! (Yaca)");
        primaryStage.show();
    }
}
```

Ergebnis …

Buttons

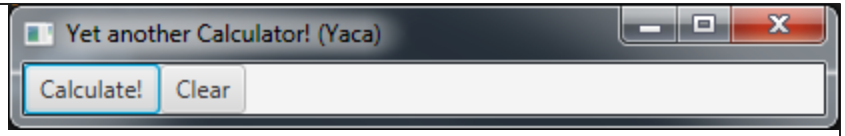
Yaca.java




```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;

public class Yaca extends Application {
    public void start(Stage primaryStage) throws Exception {

        BorderPane root = new BorderPane();


        Button calculateButton = new Button("Calculate!");
        Button resetButton = new Button("Clear");

        root.setLeft(calculateButton);
        root.setBottom(resetButton);

        Scene scene = new Scene(root);
        primaryStage.setScene(scene);
        primaryStage.sizeToScene();
        primaryStage.setTitle("Yet another Calculator! (Yaca)");
        primaryStage.show();
    }
}
```

Better Layout with FlowPane

Yaca.java

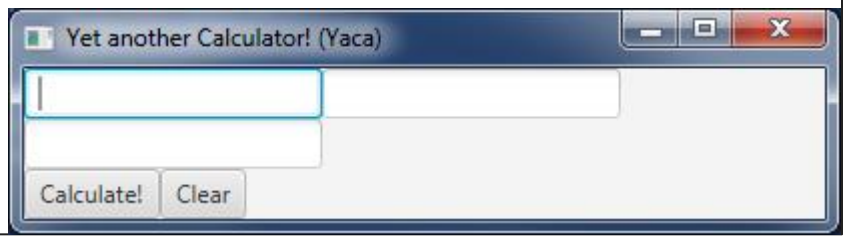



```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class Yaca extends Application {
    public void start(Stage primaryStage) throws Exception {

        BorderPane root = new BorderPane();

        Button calculateButton = new Button("Calculate!");
        Button resetButton = new Button("Clear");

        FlowPane bottomRow = new FlowPane();

        bottomRow.getChildren().addAll(
                calculateButton,
                resetButton
        );
        root.setBottom(bottomRow);

        Scene scene = new Scene(root);
        primaryStage.setScene(scene);
        primaryStage.sizeToScene();
        primaryStage.setTitle("Yet another Calculator! (Yaca)");
        primaryStage.show();
    }
}
```

Top Row & Text Fields

Yaca.java

```
import …
public class Yaca extends Application {
    public void start(Stage primaryStage) throws Exception {

        […]

        FlowPane topRow = new FlowPane();
        TextField firstNumberField = new TextField();
        TextField secondNumberField = new TextField();
        TextField resultField = new TextField();

        topRow.getChildren().addAll(
                firstNumberField,
                secondNumberField,
                resultField
        );

        bottomRow.getChildren().addAll(
                calculateButton,
                resetButton
        );

        root.setTop(topRow);
        root.setBottom(bottomRow);

        […]
    }
}
```



ComboBox & Label

Yaca.java

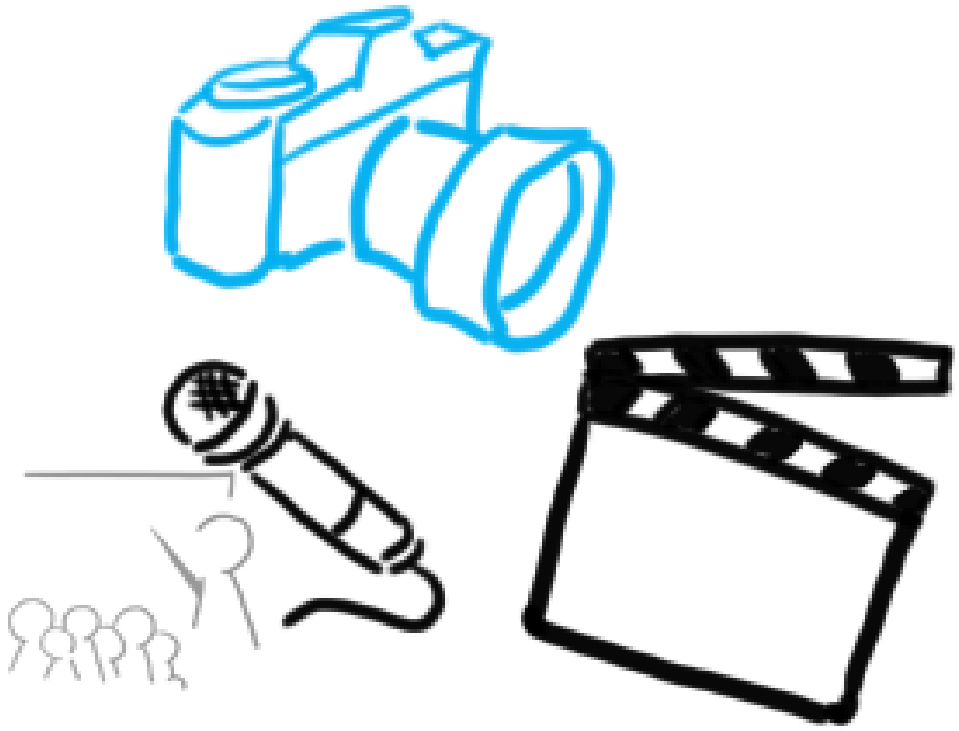
```
import …
public class Yaca extends Application {
    public void start(Stage primaryStage) throws Exception {

        […]

        Label equalsLabel = new Label("=");

        ObservableList<String> operators =
                FXCollections.observableArrayList("+", "-", "*", "/");

        ComboBox<String> operatorBox = new ComboBox<String>(operators);

        topRow.getChildren().addAll(
                firstNumberField,
                operatorBox,
                secondNumberField,
                equalsLabel,
                resultField
        );

        […]

    }
}
```

Breakout:

- Die GUI is jetzt fast komplett.
- Aufgaben:
  - Breite der Textfelder sinnvoll begrenzen, sodass sie in eine Zeile passen
  - Standard-Wert für die Textfelder
  - Standard-Operator in der ComboBox programmatisch festlegen
  - Ergebnisfeld als “read-only” setzen
  - Textfelder dürfen nur Zahlen enthalten
  - Abstand (Padding oder Margin) einfügen, damit die Elemente nicht “aneinander kleben”
  - Inhalt der einzelnen FlowPane Elemente zentrieren
- Zeit: 20 Minuten, danach gemeinsame Besprechung

Wrap-Up Quiz

1. Welche Methode der Application Klasse muss überschrieben werden? Welche Parameter nimmt sie?
2. Wofür braucht man „Szenen“?
3. Wie heißt die Markup Sprache, mit der JavaFX GUIs definiert werden können?
4. Welche GUI Technologie war die „Mutter aller Java GUIs“?
5. Wie heißt die Klasse mit der man ein Dropdown erzeugen kann?
6. Wodurch unterscheiden sich FlowPane und BorderPane?



Vielen Dank!

WELCHE FRAGEN HABT IHR?

Material & Links

- MMP Folien: Emanuel von Zezschwitz, http://www.medien.ifi.lmu.de/lehre/ss15/mmp/
- Tutorials:
  - https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
  - http://code.makery.ch/library/javafx-8-tutorial/
  - http://www.javafxtutorials.com/

- IDE specific
  - Scene Builder in IntelliJ IDEA: https://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-intellij.htm