Multimediaprogrammierung

Übung 3

JavaFX

Version 8

What is JavaFX?

- Recommended UI-Toolkit for Java 8 Applications (like e.g.: Swing, AWT)
- Current version: 8 (integrated in Java 8)
- Develop rich multimedia applications that feature audio, video, graphics and animations
- Runs on most platforms thanks to Java

JavaFX 8 - Why is it cool

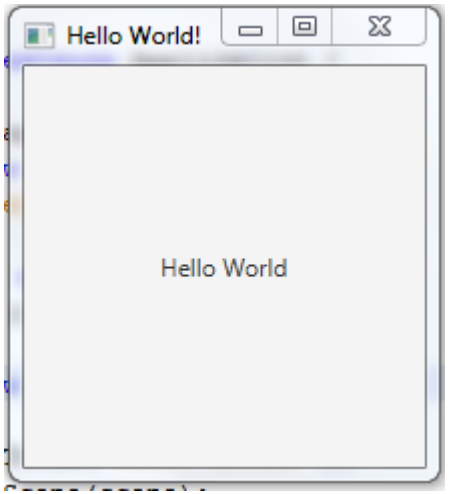
- JavaFX Scene Builder or FXML scripting language
- Cool Features for new platforms (e.g.: Multi-Touch)
- Support for ARM Platforms
- Replacement for Swing UI
- 3D Viewer
- Relies on the new language features of Java 8 (e.g.: Lambdas)
- Nashorn (Write whole application in JS)
- Styling with CSS

JavaFX - What do we need?

- JDK 1.8 (JavaFX 8 is included)
- Recommended IDE: Netbeans 8.0
- Netbeans 8.0 including JDK 1.8: http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html
- Installing in CIP Pool:
  - Download for Linux x64, locate the file with the shell
  - Make the file executable with `chmod +x jdk-8u5-nb-8-linux-x64.sh`
  - Execute it with `./jdk-8u5-nb-8-linux-x64.sh`
  - Install in your home directory
- JavaFX Scene Builder (nice to have)

Hello World in JavaFX

```
public class HelloWorld extends Application {
    @Override
    public void start(Stage primaryStage) {
        Label label = new Label();
        label.setText("Hello World");

        StackPane root = new StackPane();
        root.getChildren().add(label);

        Scene scene = new Scene(root, 200, 200);

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

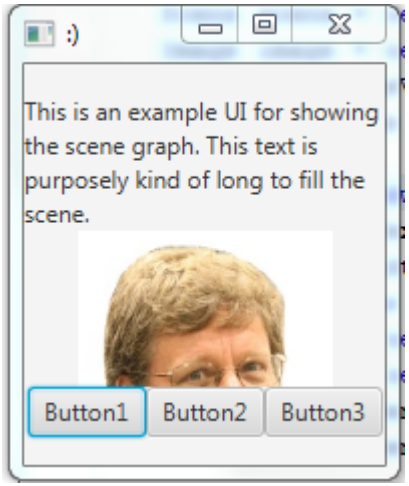


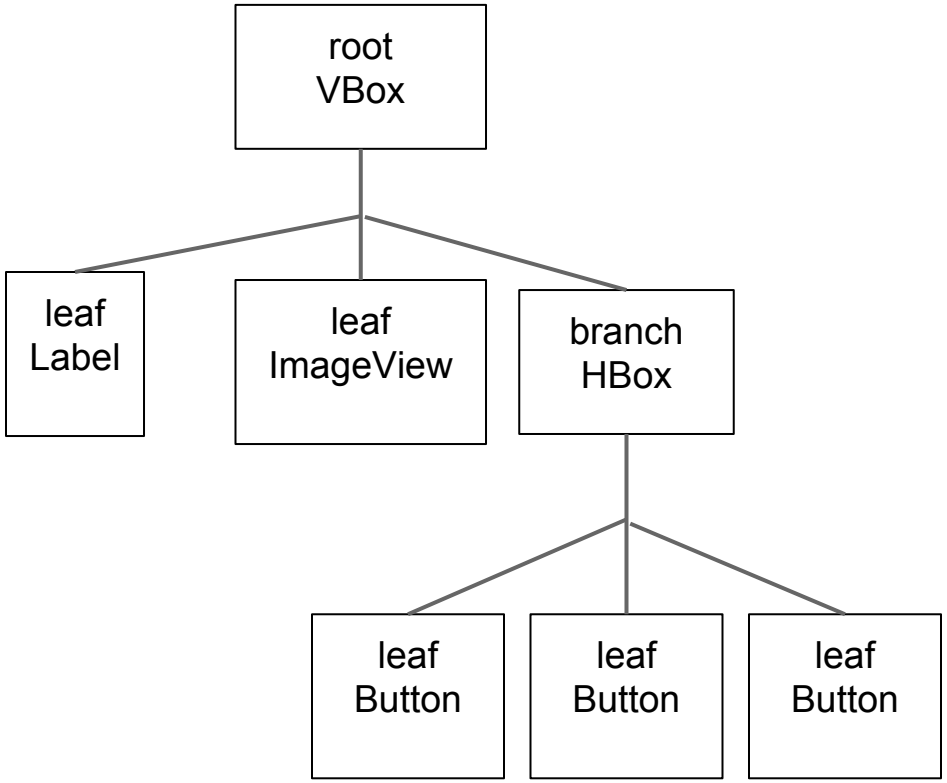
Theater Metaphor

https://commons.wikimedia.org/wiki/File:Royal_Alexandra_Stage.jpg

Scene Graph

Everything is a node in the Scene Graph





Which elements can be nodes?

- UI Elements(Buttons, Sliders)
- Layouts
- ImageView
- Shapes(Circle,Rectangle)
- Canvas
- Swing

Scene Graph - Nodes

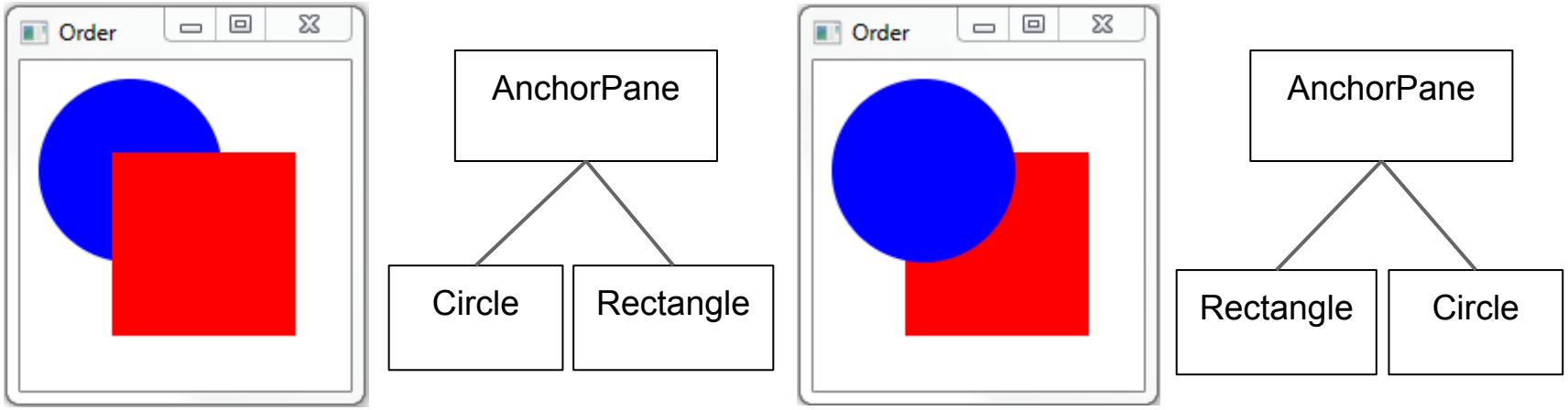
To every node you can add

- transformations(rotation, scaling, translation, etc)
- visual effects(shadows, bloom, reflections)
- animations(transitions, timeline animations)

Use groups to add these to multiple elements at a time

Scene Graph

The order of the nodes matters!



All JavaFX applications extend javafx.application.Application

```
public class HelloWorld extends Application {
    @Override
    public void start(Stage primaryStage) {
        Label label = new Label();
        label.setText("Hello World");

        StackPane root = new StackPane();
        root.getChildren().add(label);

        Scene scene = new Scene(root, 200, 200);

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

Create (UI) nodes you want

Create a pane as the root of the scene graph, add nodes

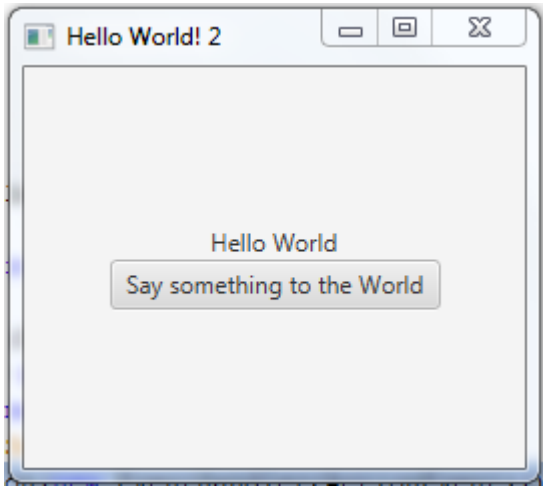
Create Scene with root node and window size

Set title and scene on stage and make it visible

Launch your JavaFX application from the public static void main

UI Example: Button

```
public class HelloWorld2 extends Application {

    boolean hello = true;
    @Override
    public void start(Stage primaryStage) {
        Label label = new Label();
        Button btn = new Button();
        btn.setText("Say something to the World");
        btn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                label.setText(hello?"Hello World":"Bye
World");
                hello=!hello;
            }
        });
        VBox vbox = new VBox();
        vbox.setAlignment(Pos.CENTER);
        vbox.getChildren().addAll(label,btn);
        Scene scene = new Scene(vbox, 250, 200);

        (...)
```



Sidenote-Lambda Expressions

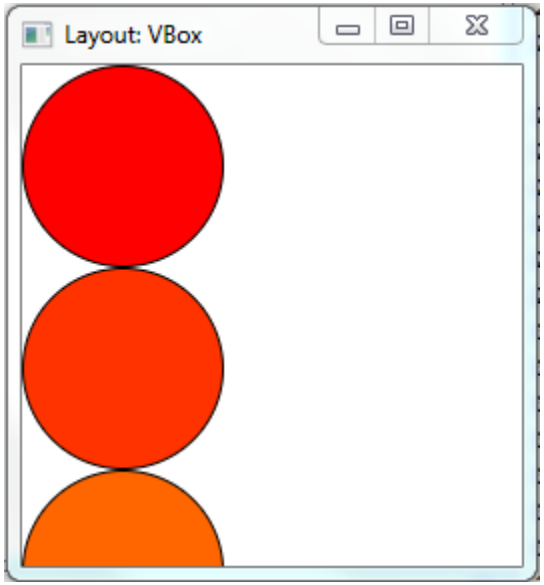
This..

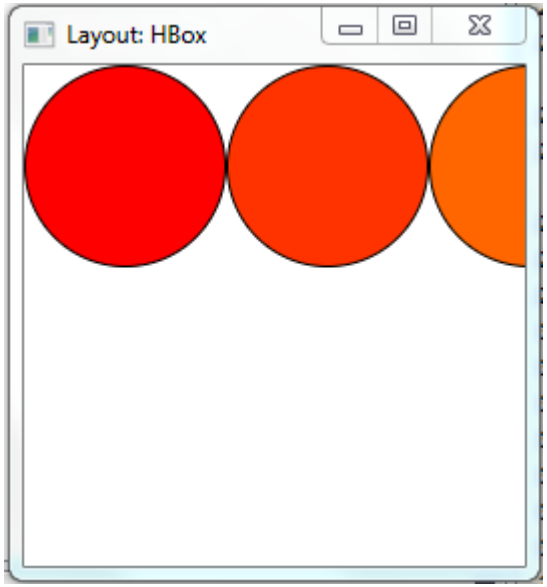
```
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        label.setText(hello?"Hello World":"Bye World");
        hello=!hello;
    }
});
```

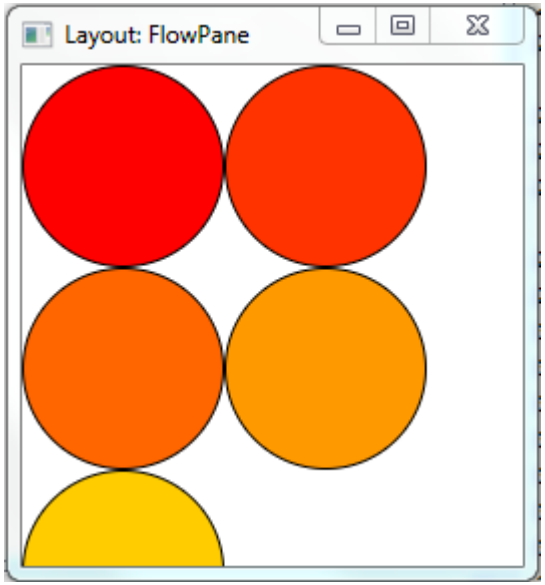
...becomes this.

```
btn.setOnAction(e->{
        label.setText(hello?"Hello World":"Bye World");
        hello=!hello;
});
```

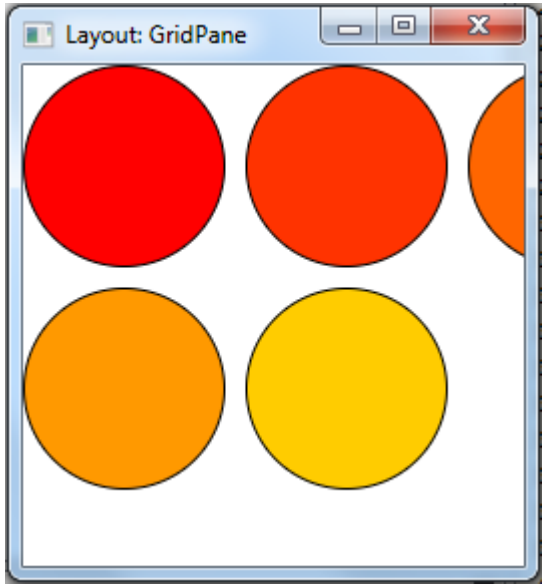
Layouts / Panes

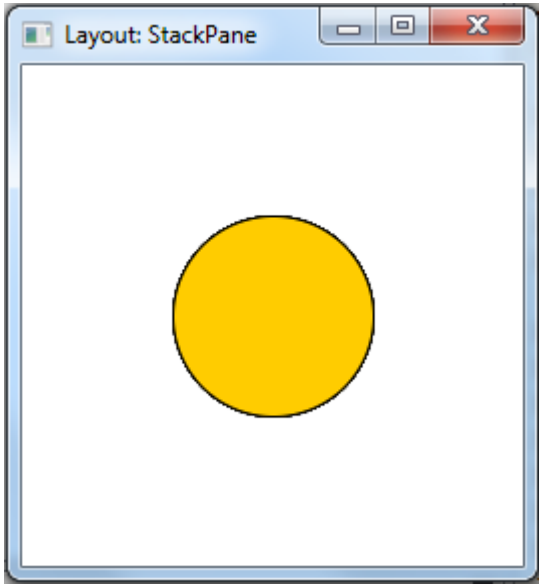






Layouts / Panes





JavaFX - Building a UI

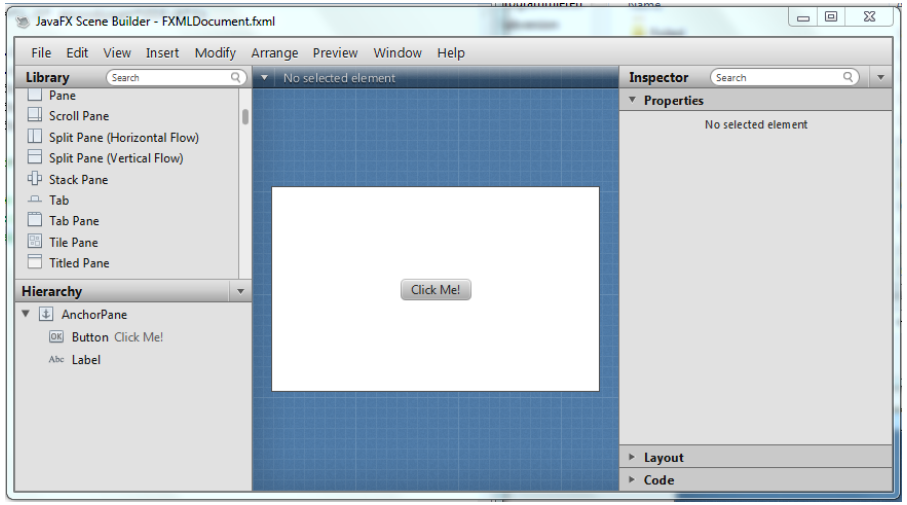
- By writing source code : `new Button("Hello!");`
- With **FXML** :

  A XML-based language for User Interfaces
- With the **JavaFX Scene Builder** :

  Drag and Drop UI Components

Java FXML Application

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane id="AnchorPane" prefHeight="200.0"
            prefWidth="320.0" xmlns:fx="http://javaf
            xmlns="http://javafx.com/javafx/2.2" fx:
  <children>
    <Button fx:id="button" layoutX="126.0" layoutY="
    <Label fx:id="label" layoutX="126.0" layoutY="12
    <Button layoutX="132.0" layoutY="51.0" mnemonicP
  </children>
</AnchorPane>
```



Write FXML Code...

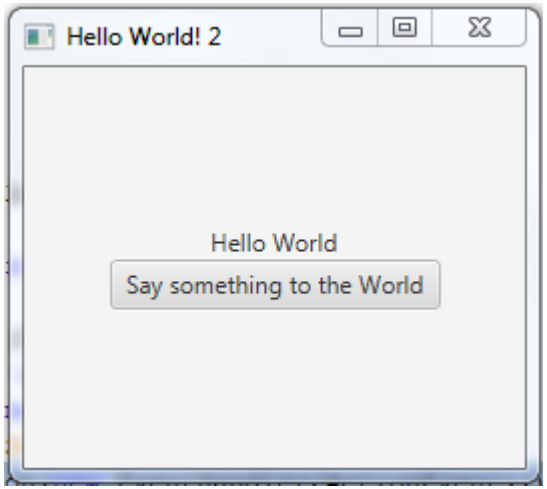
...or Use the Scene Builder (generates FXML file)

A Button with FXML (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.paint.*?>
<AnchorPane id="AnchorPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="200.0" prefWidth="250.0" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.
com/javafx/2.2">
  <children>
    <VBox alignment="CENTER" layoutX="0.0" layoutY="0.0" prefHeight="200.0" prefWidth="250.0">
      <children>
        <Label id="mylabel" fx:id="my_label" text="" />
        <Button id="mybutton" fx:id="my_button"  text="Say something to the World!" />
      </children>
    </VBox>
  </children>
</AnchorPane>
```

A Button with FXML (2)

```
public class HelloWord2FXML extends Application {

    boolean hello = true;
    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource
("FXMLDocument.fxml"));

        Scene scene = new Scene(root);
        Button btn = (Button) scene.lookup("#mybutton");
        Label label = (Label) scene.lookup("#mylabel");

        btn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                label.setText(hello?"Hello World":"Bye World");
                hello=!hello;
            }
        });

        stage.setScene(scene);
        stage.show();
    }

(...)
```



FXMLDocumentController (1)

In the FXML Document:

```
<VBox (...) fx:controller="package.FXMLDocumentController">
    <children>
        <Button onAction="#handleButtonAction" fx:id="buttonid" />
```

In the FXMLDocumentController code:

```
@FXML
private Button buttonid;
```

FXMLDocumentController (2)

In the FXML Document:

```
<VBox (...) fx:controller="package.FXMLDocumentController">
    <children>
        <Button onAction="#handleButtonAction" fx:id="buttonid" />
```

In the FXMLDocumentController code:

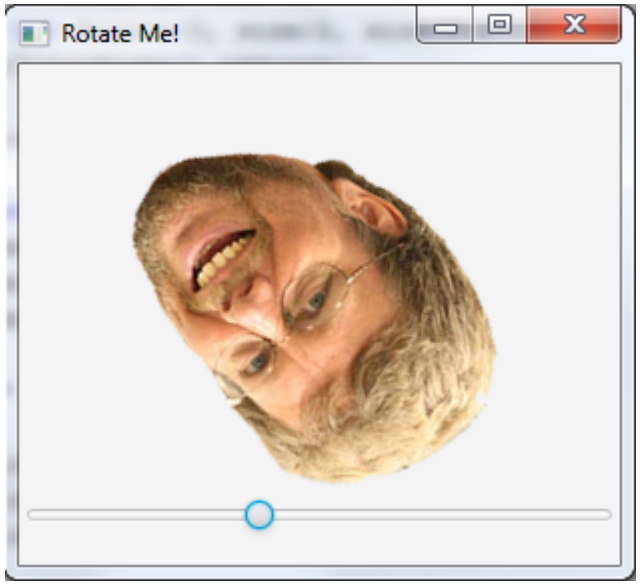
```
@FXML
    private void handleButtonAction(ActionEvent event) {
        (...)
    }
```

JavaFX - Data Binding

- Concept of binding one property to another
- Replaces the observer pattern
- the second property changes when the first one does.
- when the model changes, the bound representation will change without having to refresh
- **bind(...)**
- bidirectional binding is possible too

JavaFX - Data Binding

```
public class BindApplication extends Application {

    @Override
    public void start(Stage primaryStage) {
        Slider slider = new Slider();
        slider.setMin(0);
        slider.setMax(360);

        Image image = new Image("head.png");
        ImageView imageView = new ImageView();
        imageView.setImage(image);
        int size = (int)image.getWidth();
        Rotate rot = new Rotate(0, size/2, size/2);
        imageView.getTransforms().add(rot);

        rot.angleProperty() .bind(slider.valueProperty());

        (...)
```



binds the *angleProperty* of the Rotation to the *valueProperty* of the Slider

Data Binding - Using Properties

Use your own Properties:

```
SimpleIntegerProperty myIntegerProperty = new
SimpleIntegerProperty();

(...)

myLabel.textProperty().bind(myIntegerProperty.asString());
```

There are Properties for every Datatype.

You can even specify your own.

Useful Links

JavaFX documentation and tutorials: http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

JavaFX API:

http://docs.oracle.com/javase/8/javafx/api/

JavaFX Ensemble (Examples with Code):

http://download.oracle.com/otndocs/products/javafx/2/samples/Ensemble/index.html

JavaFX Scene Builder 2.0:
http://www.oracle.com/technetwork/java/javafx/downloads/devpreview-1429449.html