Karlsruher Institut für Technologie

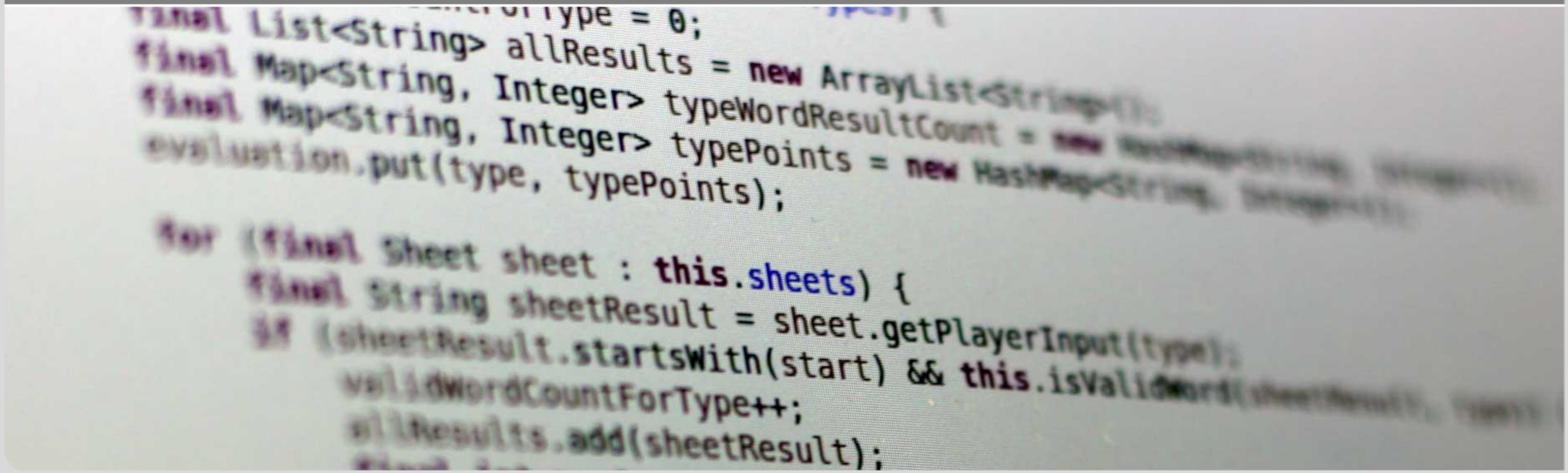
DHBW
Duale Hochschule
Baden-Württemberg

Swing vs. JavaFX

Warum eine alte Bibliothek benutzen…?!

Heusch 2 (2. Bd)
Ratz 13, 14

Institut für Automation und angewandte Informatik

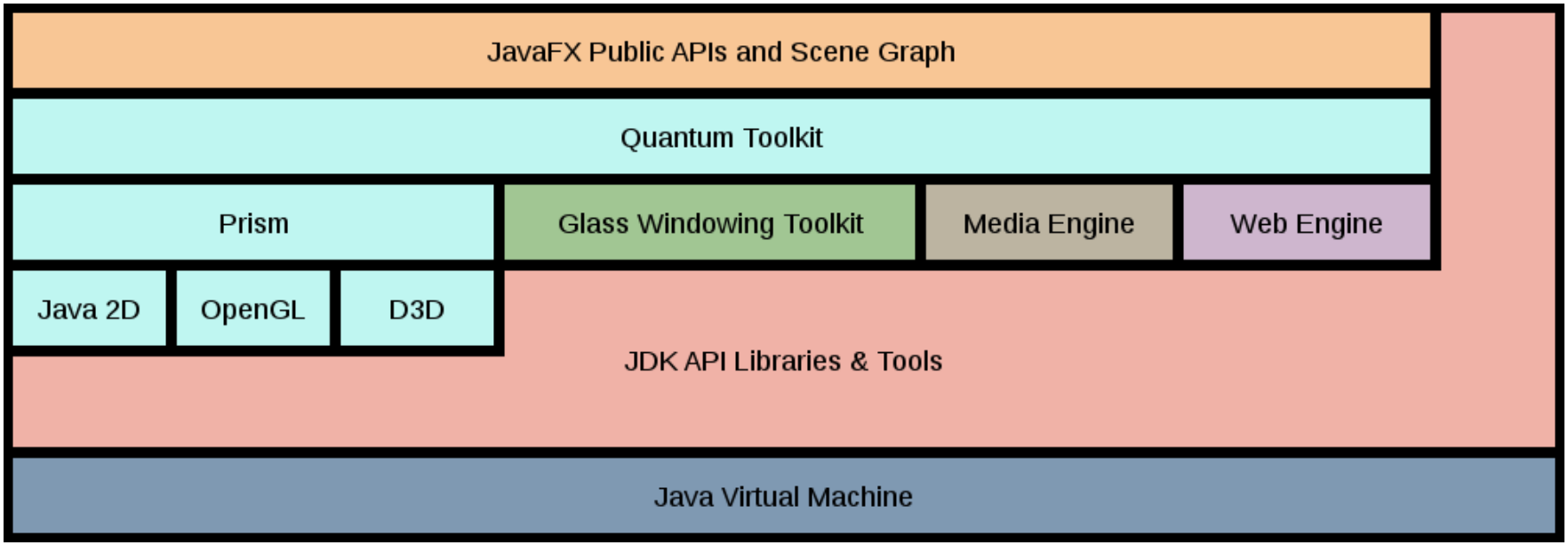


Was ist JavaFX?

- Framework zur Gestaltung plattformübergreifender, grafischer Benutzeroberflächen

- Testweise seit Java 7 (Update 6) in Java SE enthalten
- Seit **Java 8** fest in Java SE integriert („JavaFX 8“)
- Seit **Java 11** separates Modul (zusätzliche Installation nötig)

JavaFX-Architektur



Quantum: Zugriffsschicht auf Prism, GWT, Media Engine und Web Engine

Prism: Rendering Engine (nutzt vorhandene Grafik-Hardware)

Glass Windowing Toolkit (GWT): Low-Level-Routinen, z.B. zur Fensterverwaltung

Media Engine: Audio und Video abspielen

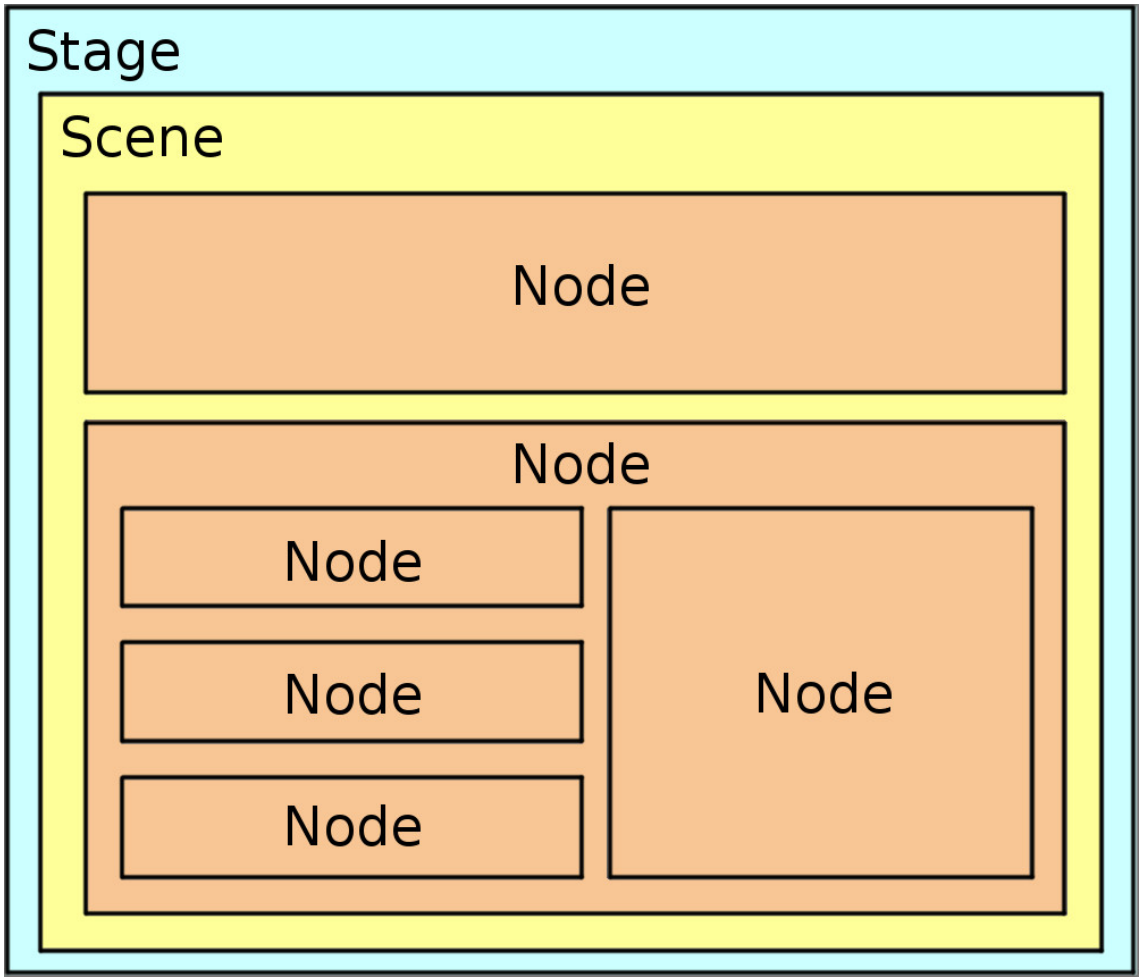
Web Engine: HTML-Rendering, CSS, DOM-Zugriff und DOM-Manipulation

Bildquelle: Wikipedia.org; Autor: “Stkl”

Grundlegendes zu JavaFX-Anwendungen

- JavaFX-Basis-Package: `javafx.*`
- Basisklasse ist (JavaFX-) **`Application`**
  - Instanz von `Application` erzeugen,
    Methode `init()` wird aufgerufen (Implementierung optional)
  - Methode **`start(javafx.stage.Stage)`** aufrufen
    (`Stage` = Hauptcontainer, “Root-Container”)
  - JavaFX-Runtime wartet auf Ende, z.B.
    Aufruf von `Platform.exit()` oder
    auf Schließen des “letzten” Fensters
- Auf der `Stage` können eine oder mehrere `Scene`s platziert werden, die wiederum `Node`s (vergl. “Grundkomponenten” in Swing) enthalten können

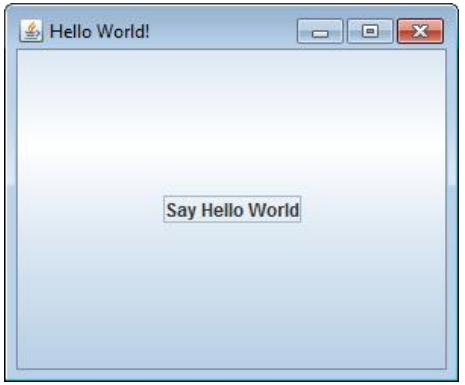
JavaFX: Aufbau UI



Bildquelle: Wikipedia.org; Autor: “Stkl”

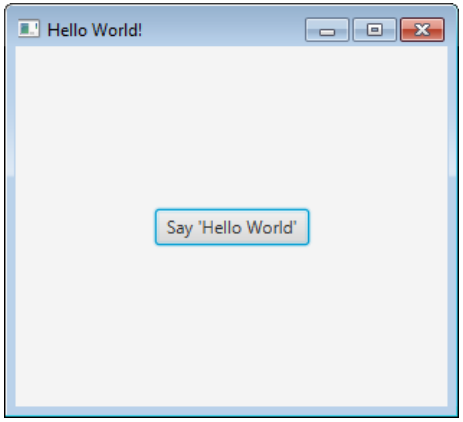
HelloWorld in Swing

```
import … (4 Zeilen);
public class HelloWorldExampleSwing extends JFrame {
   public HelloWorldExampleSwing(String title) {
       super(title);
       this.setLayout(new BorderLayout());
       JButton btn = new JButton("Say Hello World");
       btn.addActionListener(new ActionListener() {
           @Override
           public void actionPerformed(java.awt.event.ActionEvent e) {
               System.out.println("Hello World!");
           }
       });
       this.add(btn);
       this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       this.setSize(300, 250);
       this.setVisible(true);
   }
   public static void main(String[] args) {
       new HelloWorldExampleSwing("Hello World!");
   }
}
```



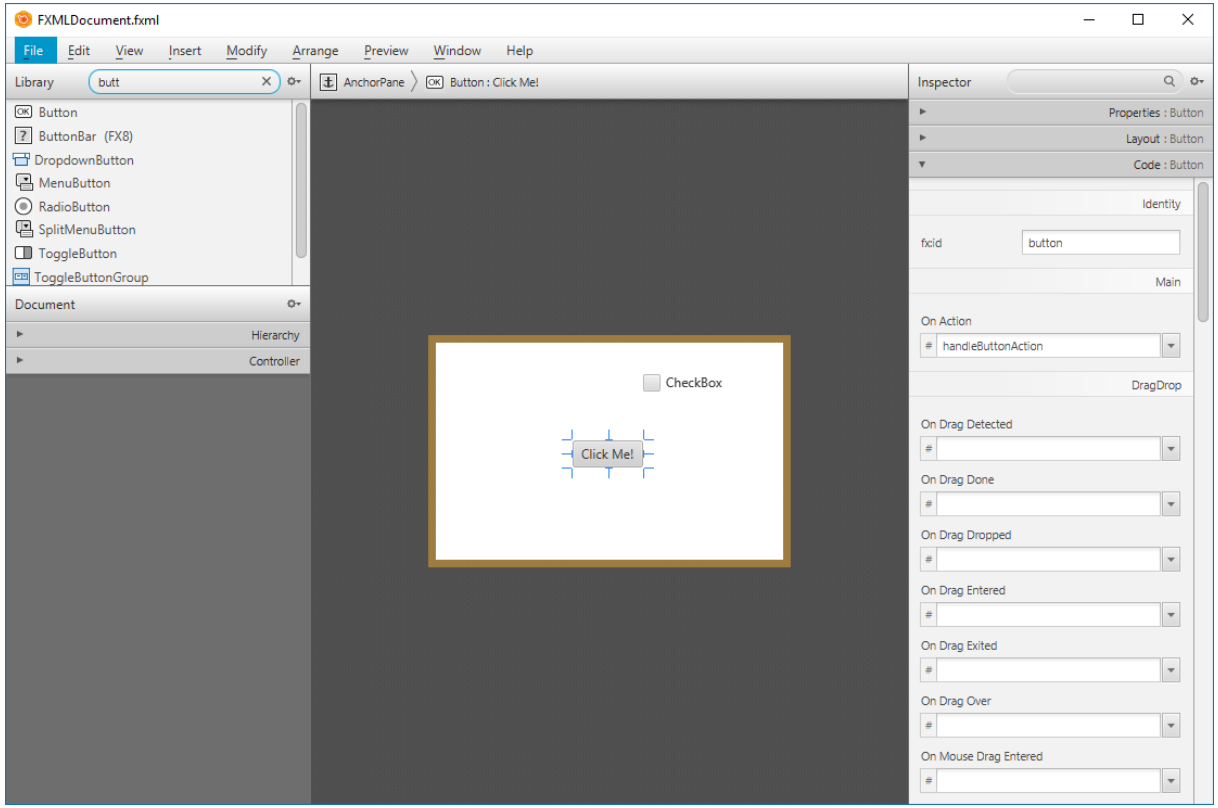
HelloWorld in JavaFX

```
import … (7 Zeilen);
public class HelloWorldExampleFX extends Application {
   @Override
   public void start(Stage primaryStage) {
       Button btn = new Button();
       btn.setText("Say Hello World");
       btn.setOnAction(new EventHandler<ActionEvent>() {
           @Override
           public void handle(ActionEvent event) {
               System.out.println("Hello World!");
           }
       });
       StackPane root = new StackPane();
       root.getChildren().add(btn);
       Scene scene = new Scene(root, 300, 250);
       primaryStage.setTitle("Hello World!");
       primaryStage.setScene(scene);
       primaryStage.show();
   }
   public static void main(String[] args) {
       Application.launch(args); // statische Methode,
                                 // die init() und start(…) aufruft
   }
}
```



Java 8 – Features mit JavaFX

- Richtig geschmeidig wird JavaFX durch Nutzung von grafischen Editoren (z.B. “Scene Builder”) und Codierung in Java 8-Notation (Closures).



Was liefert der Scene Builder?

- XML-Code zur Beschreibung des Layouts (verkürzt)

```
<GridPane fx:controller="de.dhbwka.examples.fx.FXMLExampleController"
          xmlns:fx="http://javafx.com/fxml" alignment="center">
    <HBox spacing="10" alignment="bottom_right"
          GridPane.columnIndex="1" GridPane.rowIndex="4">
        <Text fx:id="actiontarget"
          GridPane.columnIndex="1" GridPane.rowIndex="6"/>
        <Button text="Sign In“ onAction="#handleSubmitButtonAction"/>
    </HBox> <!-- plus weitere Elemente... -->
</GridPane>
```

- Generierter Java-Code

```
packacke de.dhbwka.examples.fx;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.text.Text;
public class FXMLExampleController {
    @FXML private Text actiontarget;
    @FXML protected void handleSubmitButtonAction(ActionEvent event) {
        actiontarget.setText("Sign in button pressed"); // "Nutzcode"
    }
}
```

Eventhandling mit JavaFX

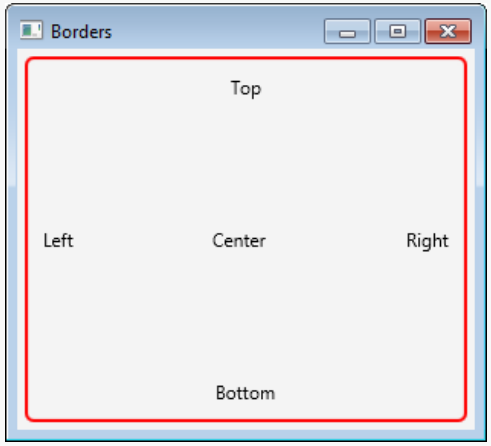
- Generische Klasse `EventHandler` spezifiziert durch Type-Parameter (s. “Collections”)

```
EventHandler<ActionEvent> handler = new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.println("Hello World!");
    }
};
btn.setOnAction(handler);
```

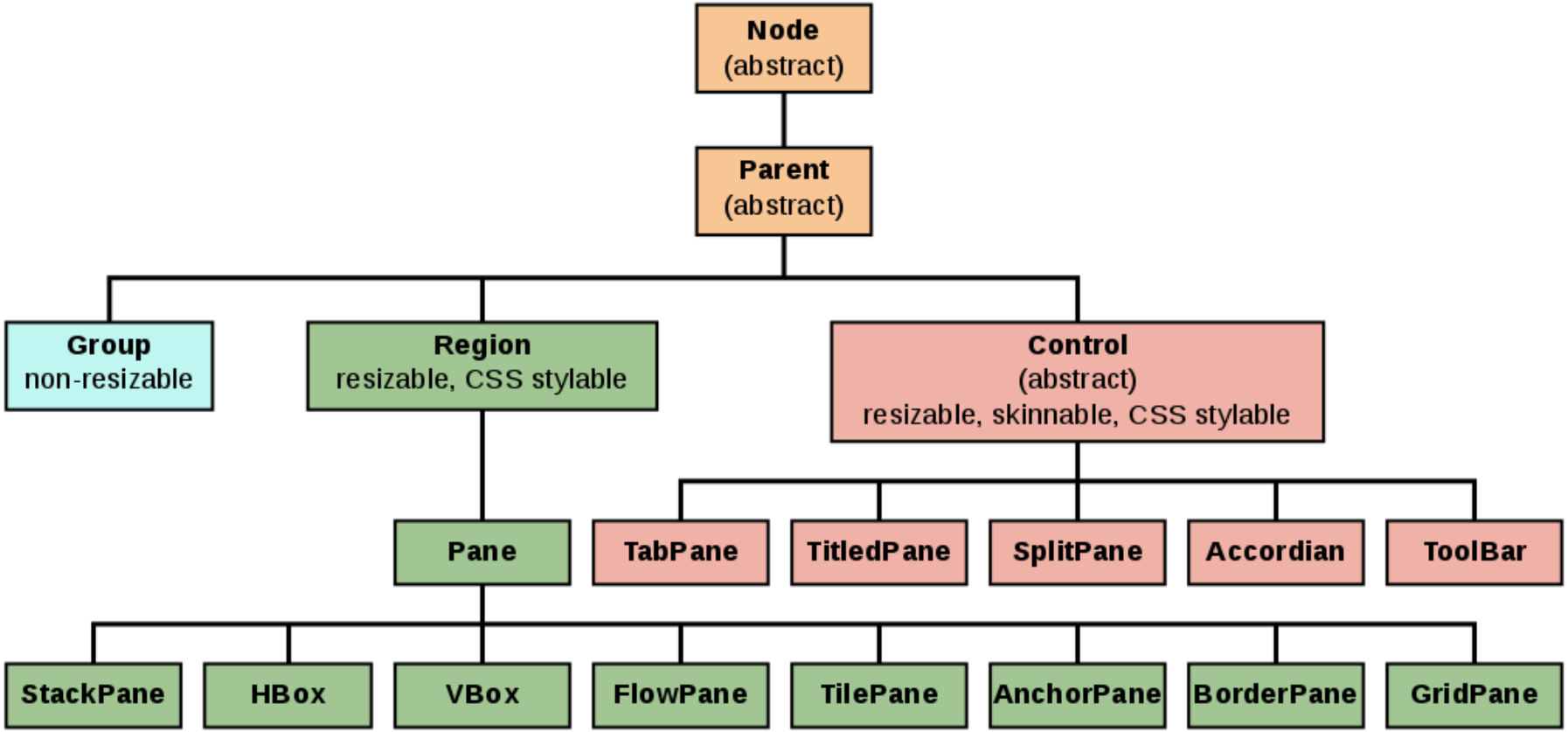
Layout mit JavaFX

- Es gibt im Prinzip alle Swing Layouts auch in JavaFX, z.B. `javafx.scene.layout.BorderPane`

```
Text centerText = new Text("Center");
Text topText = new Text("Top");
Text rightText = new Text("Right");
Text bottomText = new Text("Bottom");
Text leftText = new Text("Left");
// Set the alignment of the Text fields
BorderPane.setAlignment(topText, Pos.TOP_CENTER);
BorderPane.setAlignment(bottomText, Pos.BOTTOM_CENTER);
BorderPane.setAlignment(leftText, Pos.CENTER_LEFT);
BorderPane.setAlignment(rightText, Pos.CENTER_RIGHT);
// Create a BorderPane with a Text node in each of the five regions
BorderPane root = new BorderPane(centerText, topText, rightText, bottomText, leftText);
// Set the Style-properties of the BorderPane
root.setStyle( "-fx-padding: 10; -fx-border-style: solid inside;" +
               "-fx-border-width: 2; -fx-border-insets: 5;" +
               "-fx-border-radius: 5; -fx-border-color: red;" );
Scene scene = new Scene(root, 300, 250);
```



Kommt bekannt vor: JavaFX-Layouts



Bildquelle: Wikipedia.org; Autor: “Stkl”

Schreibweise mit Closures (1)

- Die Implementierung per anonymer innerer Klasse lässt sich seit Java 8 auch in Form von Closures (Funktion) schreiben.
  Hier eine leicht abgewandelte Form
  (Ausgabe des Events (per `.toString()`):

```
EventHandler<ActionEvent> handler = new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.println(event);
    }
};
btn.setOnAction(handler);
```

Schreibweise mit Closures (2)

- Funktionen (Closures) können direkt als Parameter übergeben werden. Es gibt mehrere Varianten:

```
Button btn = new Button("Say 'Hello World'");

// Variante 1 - vollständig
btn.setOnAction( (ActionEvent event) -> { System.out.println(event); } );

// Variante 2 – Typ weglassen, Eine Anweisung statt Block
btn.setOnAction( (event) -> System.out.println(event) );

// Variante 3 – Auch noch die Klammer weglassen
btn.setOnAction( event -> System.out.println(event) );

// Variante 4 – Ein Resultat, ein Parameter? Einfach Durchreichen!
btn.setOnAction( System.out::println );
```

Das geht übrigens auch mit Swing…

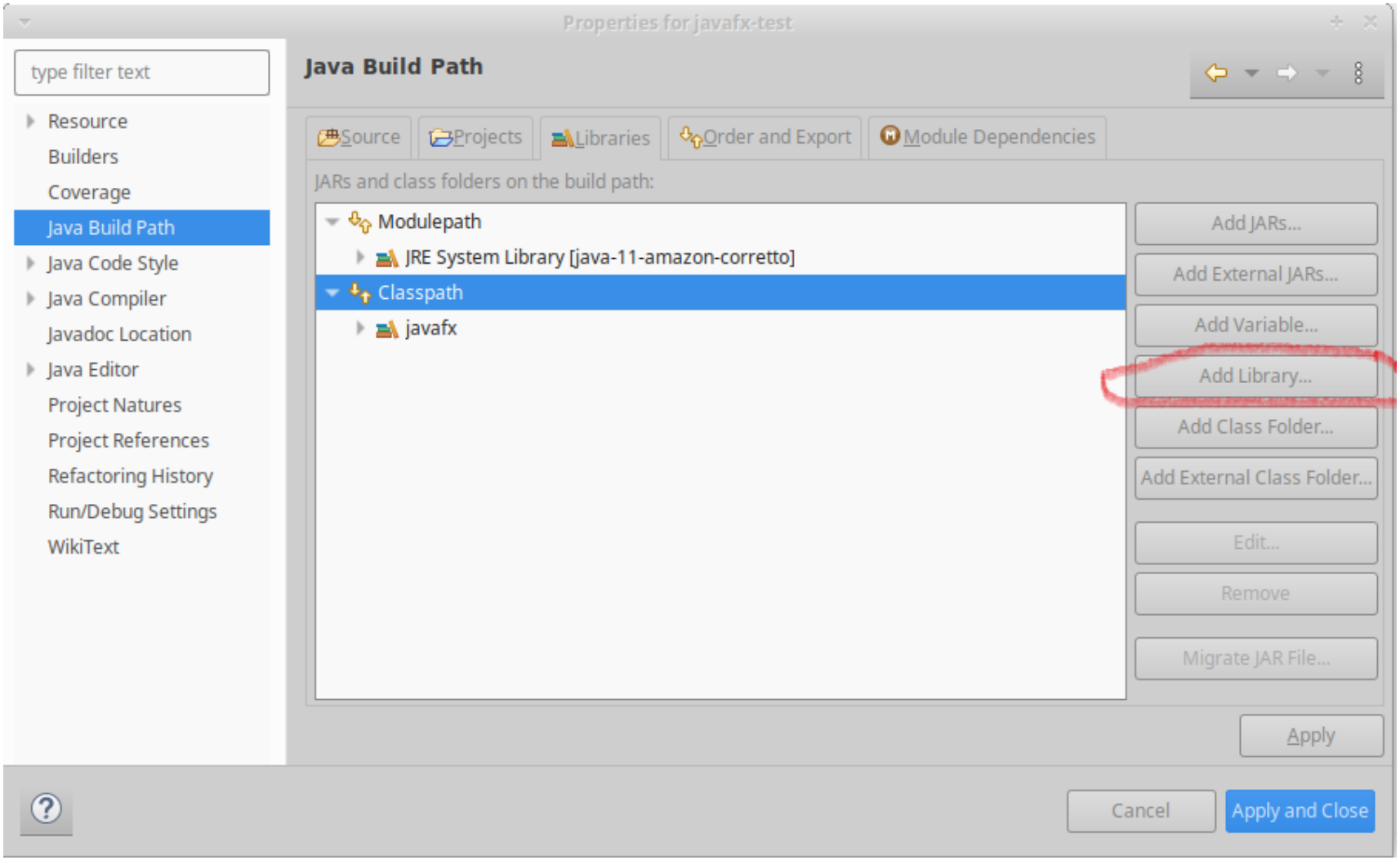
- …vorausgesetzt man verwendet Java ab Version 8

```
JButton btn = new JButton("Say 'Hello World'");

// Ersetze
btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(java.awt.event.ActionEvent e) {
        System.out.println("Hello World!");
    }
});

// durch
btn.addActionListener( (ActionEvent event) -> { System.out.println("Hello World!"); } );

// oder einfach durch
btn.addActionListener( event -> System.out.println("Hello World!") );
```

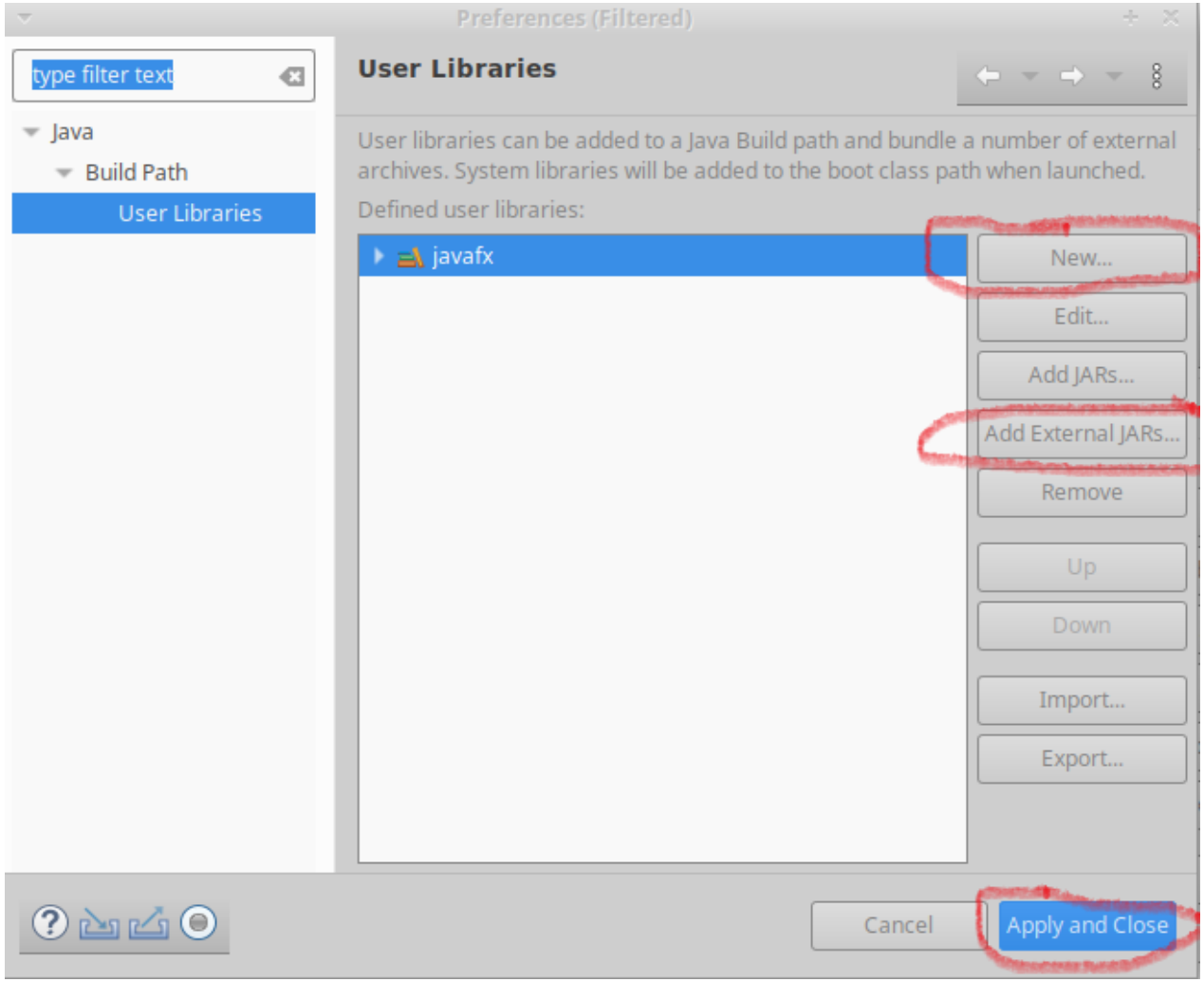
Einbinden von JavaFX in eclipse (1)

- Download von passender Version JavaFX
  - https://gluonhq.com/products/javafx/
- Projekt erstellen, anschließend über Properties → Java Build Path → Reiter "Libraries", dann "classpath" wählen
- "Add Library…" wählen, dann "User library", dann "User libraries…", anschließend "New…" wählen und neue library erstellen
- Anschließend library auswählen und erstellte .jars über "Add External JARs…" aus dem Dateisystem hinzufügen
- Schließlich "Apply and Close"
- Danach noch folgende Zeile als VM-arguments der runconfig hinzufügen (nur ein Beispiel, es gibt noch mehr)
  - --module-path /path/to/javafx/lib --add-modules javafx.base,javafx.controls,javafx.fxml,javafx.swing

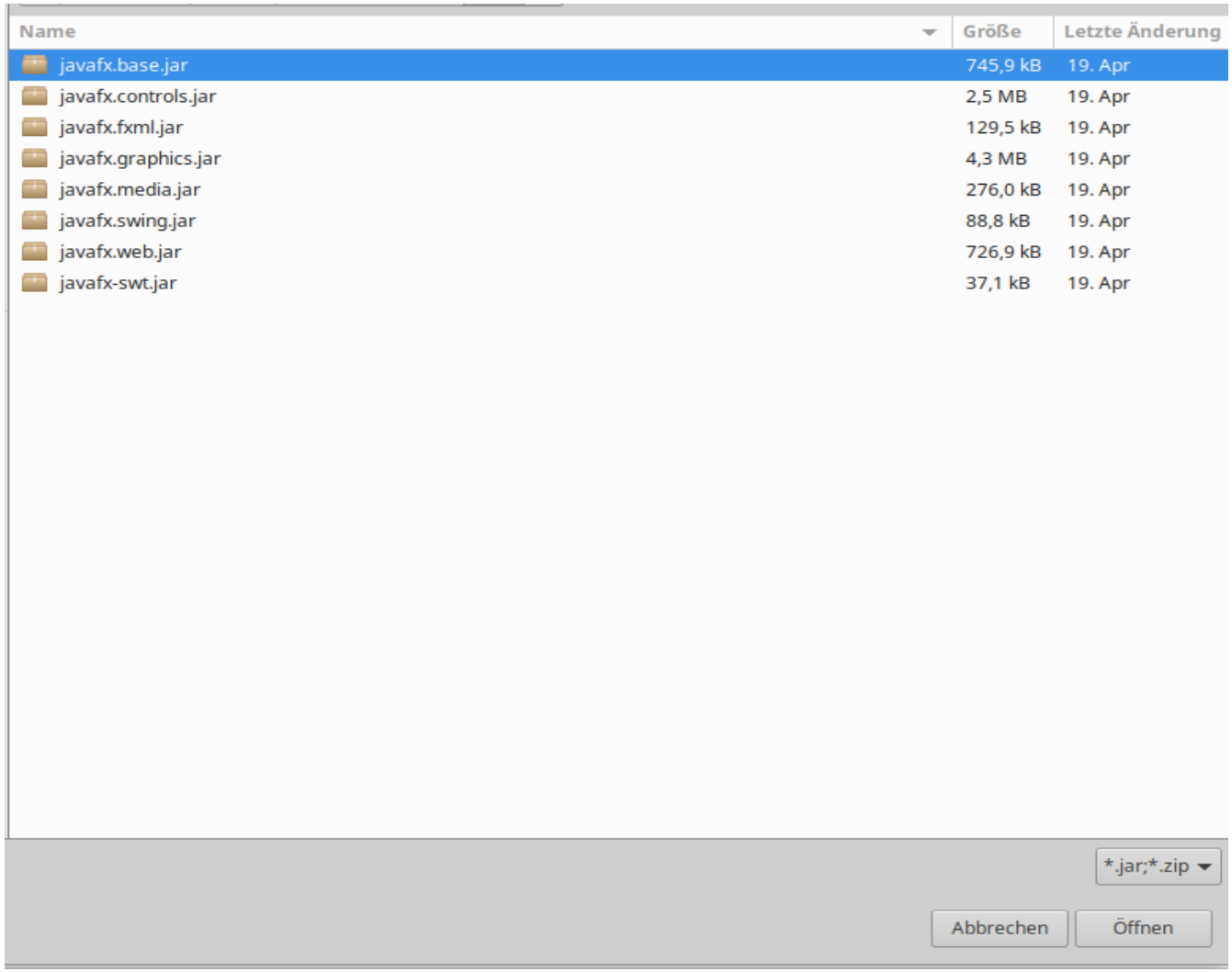
Einbinden von JavaFX in eclipse (2)



Einbinden von JavaFX in eclipse (3)

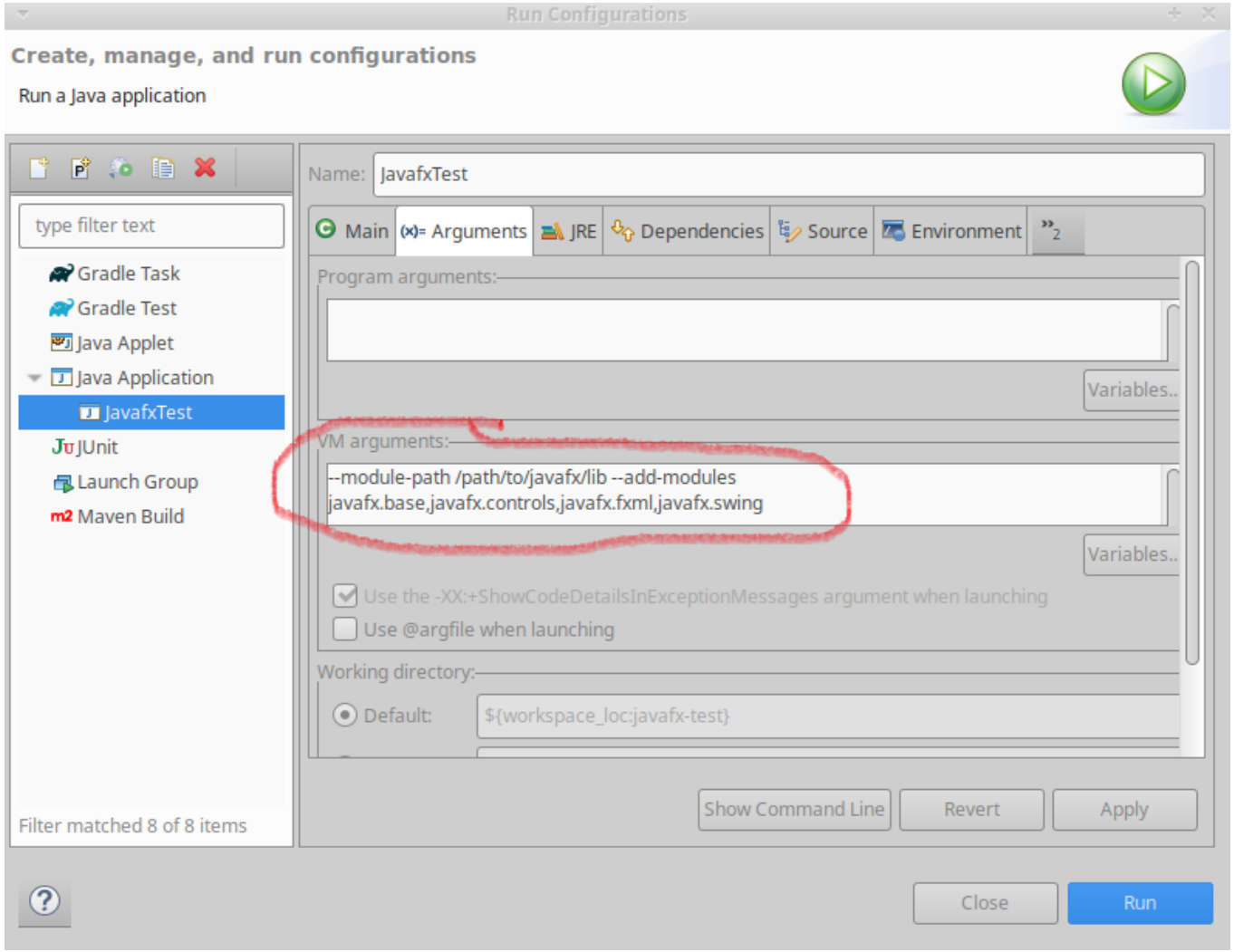


Einbinden von JavaFX in eclipse (4)



alle selektieren!

Einbinden von JavaFX in eclipse (5)

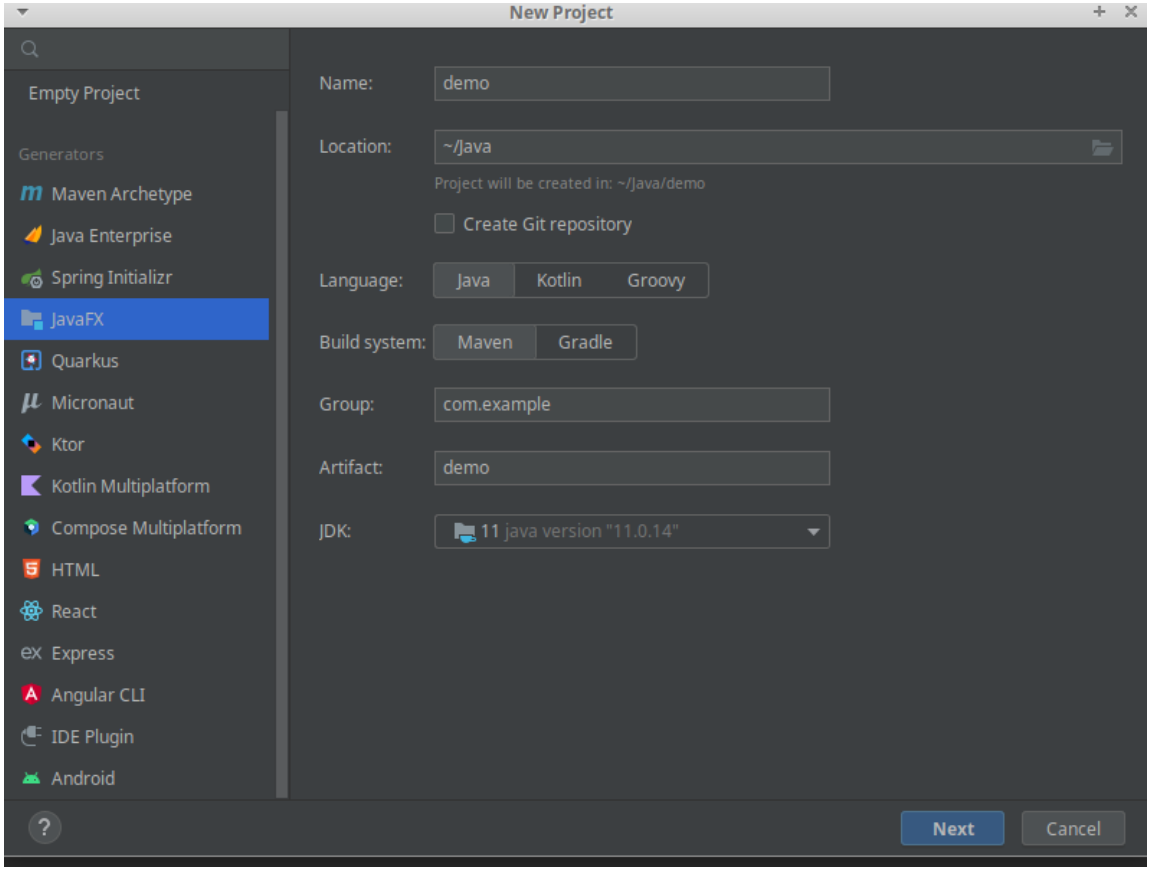


"/path/to/javafx" entsprechend anpassen ☺

Einbinden von JavaFX in IntelliJ

- Sehr leicht: New… → Project und links JavaFX auswählen
- Projekt erstellen und fertig ☺

auswählen →



JavaFX sieht doch gut aus!

- Warum also nicht in der Programmieren-Vorlesung?
  - **Komplexität beim Styling** (spezielles CSS bzw. spezielle Styling-Philosophie) – *“Man muss jede Menge wissen!”*
  - Eigene Philosophie mit **Stages und Scenes**.
    Zwar nicht *sehr* komplex, aber doch zusätzliche Komplexität, die wir in unseren (einfachen) Beispielen kaum nutzen.
  - Einstieg in **Java 8** bedeutet deutliche Hürde (insbes. Lambdas).
  - Prinzipien der GUI-Entwicklung werden auch an den **(einfacheren!) Swing-Beispielen** klar.