Ablaufsteuerung durch Events

Don’t call us, we call you!

Hochschule **Rosenheim**
University of Applied Sciences

Agenda

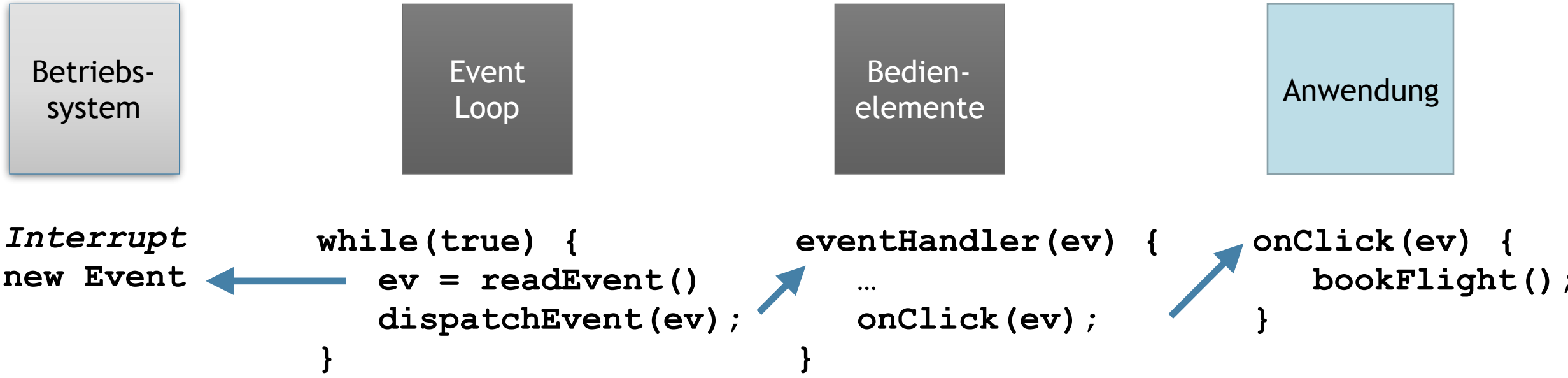
1. Überblick und Theorie (20 Minuten)
2. Demo Login Dialog (20 Minuten)
3. Eventverarbeitung mit JavaFX (20 Minuten)

Hochschule **Rosenheim**
University of Applied Sciences

Events steuern den Kontrollfluss

- Kontrollstrukturen in Programmiersprachen
  - **Funktionen / Methoden**
  - **Schleifen**
  - **Verzweigungen**
  - **Ausnahmen**
  - **Events**
    - Die Hauptschleife in UI-Programmen bezeichnet man als Event-Loop
    - Die Schleife wird bei allen Mainstream-UI -Technologien in einem einzigen Thread durchlaufen

Hochschule **Rosenheim**
University of Applied Sciences

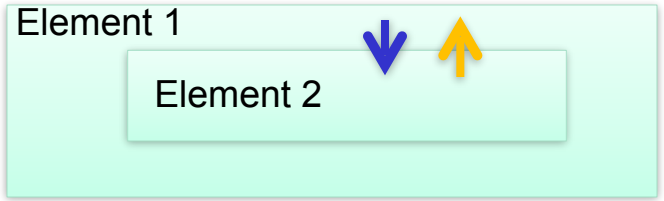

Betriebs-
system
Event
Loop
Bedien-
elemente
Anwendung
Interrupt
new Event
while(true) {
ev = readEvent()
dispatchEvent(ev);
}
eventHandler(ev) {
...
onClick(ev);
}
onClick(ev) {
bookFlight();
}

Wichtige Begriffe

- **Event** == Information über ein Ereignis (Was, wann, wo)
- **Eventhandler** == Funktion / Methode die aufgerufen wird wenn ein Event eintritt (z.B. KeyPressed)
- **Eventfilter** == Funktion / Methode die aufgerufen wird wenn ein Event eintritt. Der Filter kann das Event blockieren/konsumieren. Der Handler wird dann nicht mehr ausgelöst.
- **Callback** == Der Eventhandler/-filter wird nie manuell aufgerufen – Das UI-Framework steuert den Aufruf
- **Event-Loop** == Die Hauptschleife in allen Oberflächen ist ein Thread der die Events an die Handler weiterleitet.

Hochschule **Rosenheim**
University of Applied Sciences

http://www.youtube.com/watch?feature=player_detailpage&v=PNLNEzXcZE4#t=293s



Bubbling vs. Capturing

- Legt die Reihenfolge im Handleraufruf fest
  - Event Capturing stellt den Event erst Element1 zu
    - Historisch: Standard bei Netscape
  - Event Bubbling stellt den Event erst Element2 zu.
    - Historisch: Standard bei Microsoft
  - **W3C Model & JavaFX:** Zuerst Capturing, dann Bubbling

- Bubbling ist für Default-Handler sinnvoll
  - Ein einziger Action - Handler für alle Buttons eines Taschenrechners
  - Drag & Drop auf der ganzen UI
- Capturing ermöglich übergeordnete Filter

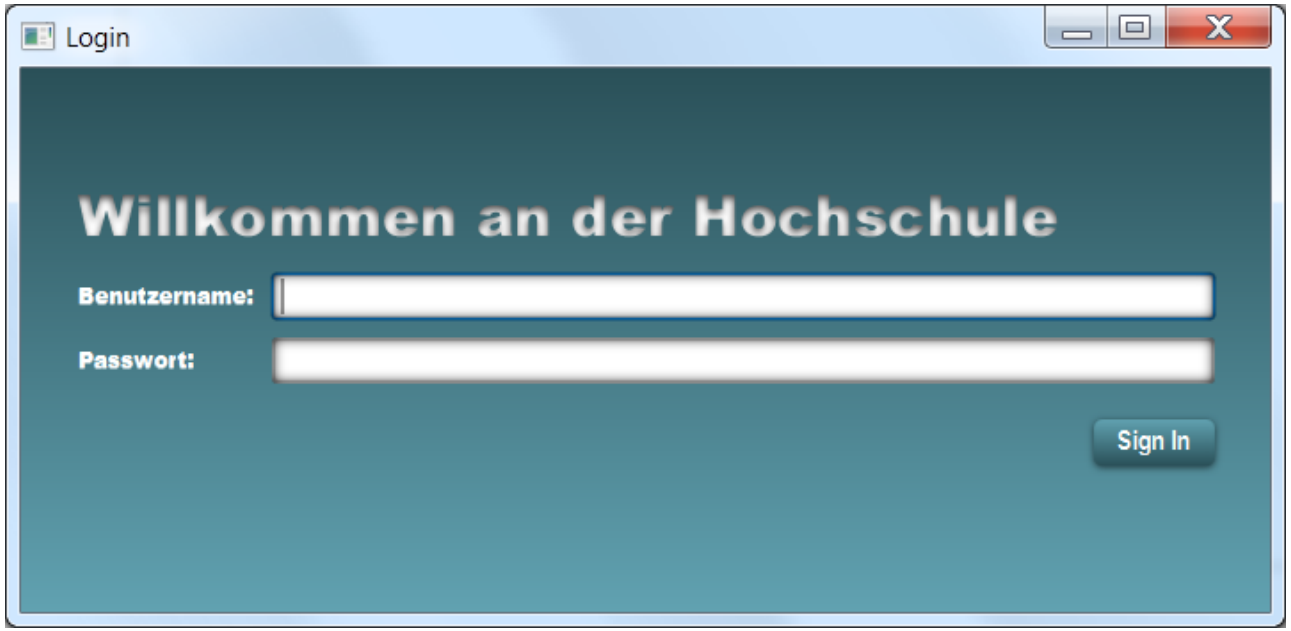
Hochschule Rosenheim
University of Applied Sciences

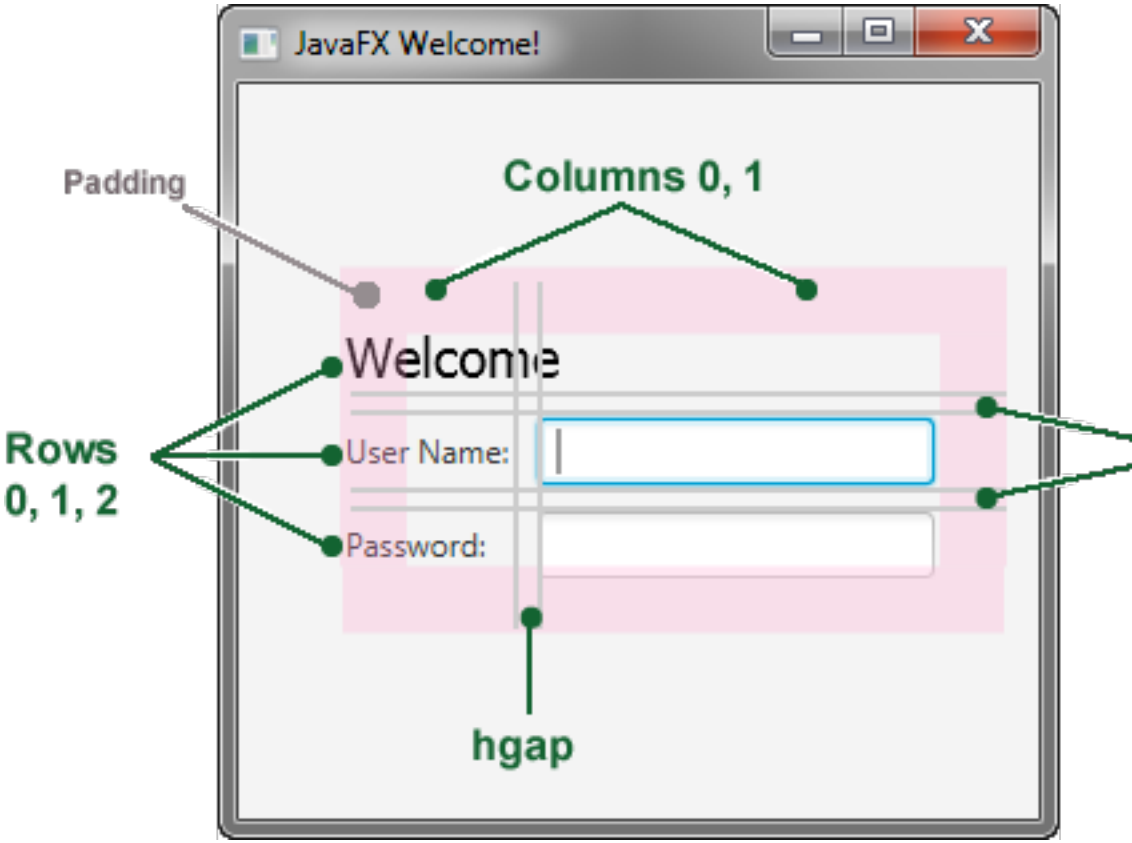
Agenda

1. Überblick und Theorie (20 Minuten)
2. Demo Login Dialog (20 Minuten)
3. Eventverarbeitung mit JavaFX (20 Minuten)

Hochschule **Rosenheim**
University of Applied Sciences

Live Demo – Login Dialog





Hochschule Rosenheim
University of Applied Sciences

Agenda

1. Überblick und Theorie (20 Minuten)
2. Demo Login Dialog (20 Minuten)
3. **Eventverarbeitung mit JavaFX (20 Minuten)**

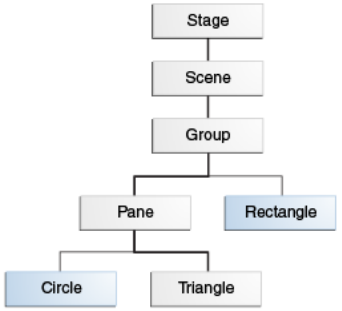
Hochschule **Rosenheim**
University of Applied Sciences

Events in JavaFX

- Repräsentieren ein Ereignis für das sich Anwendungen registrieren können
- Erweitern die Basisklasse `javafx.event.Event`
- Beispiele: `DragEvent, KeyEvent, MouseEvent, ScrollEvent`
- Die Klasse `javafx.event.Event` hat folgende Attribute
  - Event Type: Die Typinformation (class EventType)
  - Source: Identifiziert den Sender (class Object) (geerbt von java.util.EventObject)
  - Event Target: Identifiziert das Ziel (interface EventTarget)

Hochschule Rosenheim
University of Applied Sciences

Die Event-Zustellung besteht aus vier Schritten



1. Target selection → Auswahl der Zielkomponente
   - Bei Tastatur Ereignissen -> Die Komponente mit dem Focus
   - Bei Maus Ereignissen -> Die Komponente unter dem Zeiger
2. Route construction → Erstellung der Verarbeitungskette
   - Wenn die Zielkomponente ermittelt wurde wird eine Event Dispatch Chain aufgebaut (→ Eine Kette aller beteiligten Komponenten)
3. Event capturing Phase → Durchlauf des Events Bottom/Top
   - Reihenfolge: Vom Wurzelknoten zum Kind (Stage->Triangle)
   - EventFilter greifen in dieser Phase und können verhindern das ein Event zum Kind gesendet wird
4. Event bubbling Phase → Durchlauf des Events Top/Bottom
   - Reihenfolge: Vom Kind zur Wurzel (Triangle -> Stage)
   - EventHandler greifen in dieser Phase
   - Ein Handler kann die Weiterleitung zum Parent mit event.consume() verhindern

Hochschule Rosenheim
University of Applied Sciences

Handler Registrierung (programmatisch, einfache Variante)

- setOn**Event-type**(EventHandler<? super event-class> value)

```
Button btn = new Button(„Exit“);
    btn.setOnAction(new EventHandler<ActionEvent>() {
      public void handle(ActionEvent e) {
            << your code >>
      }
    }
```

- Handler wird normalerweise bei der Controller-Initialisierung erzeugt (siehe interface **Initializable)**
- Einschränkungen
  - Nur ein Event-Handler pro Event-Typ und Node
  - Funktioniert nur für Handler – Nicht für Filter

Hochschule **Rosenheim**
University of Applied Sciences

setOn<<Event>> - Methoden (Button)

```
setOnAction(EventHandler<ActionEvent> value)
setOnContextMenuRequested(EventHandler<? super ContextMenuEvent> value)
setOnDragDetected(EventHandler<? super MouseEvent> value)
setOnDragDone(EventHandler<? super DragEvent> value)
setOnDragDropped(EventHandler<? super DragEvent> value)
setOnDragEntered(EventHandler<? super DragEvent> value)
setOnDragExited(EventHandler<? super DragEvent> value)
setOnDragOver(EventHandler<? super DragEvent> value)
setOnInputMethodTextChanged(EventHandler<? super InputMethodEvent> value)
setOnKeyPressed(EventHandler<? super KeyEvent> value)
setOnKeyReleased(EventHandler<? super KeyEvent> value)
setOnKeyTyped(EventHandler<? super KeyEvent> value)
setOnMouseClicked(EventHandler<? super MouseEvent> value)
setOnMouseDragEntered(EventHandler<? super MouseDragEvent> value)
setOnMouseDragExited(EventHandler<? super MouseDragEvent> value)
setOnMouseDragOver(EventHandler<? super MouseDragEvent> value)
setOnMouseDragReleased(EventHandler<? super MouseDragEvent> value)
```

```
setOnMouseDragged(EventHandler<? super MouseEvent> value)
setOnMouseEntered(EventHandler<? super MouseEvent> value)
setOnMouseExited(EventHandler<? super MouseEvent> value)
setOnMouseMoved(EventHandler<? super MouseEvent> value)
setOnMousePressed(EventHandler<? super MouseEvent> value)
setOnMouseReleased(EventHandler<? super MouseEvent> value)
setOnRotate(EventHandler<? super RotateEvent> value)
setOnRotationFinished(EventHandler<? super RotateEvent> value)
setOnRotationStarted(EventHandler<? super RotateEvent> value)
setOnScroll(EventHandler<? super ScrollEvent> value)
setOnScrollFinished(EventHandler<? super ScrollEvent> value)
setOnScrollStarted(EventHandler<? super ScrollEvent> value)
setOnSwipeDown(EventHandler<? super SwipeEvent> value)
setOnSwipeLeft(EventHandler<? super SwipeEvent> value)
setOnSwipeRight(EventHandler<? super SwipeEvent> value)
setOnSwipeUp(EventHandler<? super SwipeEvent> value)
setOnTouchMoved(EventHandler<? super TouchEvent> value)
```

```
setOnTouchPressed(EventHandler<? super TouchEvent> value)
setOnTouchReleased(EventHandler<? super TouchEvent> value)
setOnTouchStationary(EventHandler<? super TouchEvent> value)
setOnZoom(EventHandler<? super ZoomEvent> value)
setOnZoomFinished(EventHandler<? super ZoomEvent> value)
setOnZoomStarted(EventHandler<? super ZoomEvent> value)
```

Hochschule **Rosenheim**
University of Applied Sciences

Feld-Injektion zwischen FXML + Java

- Handler müssen in der Regel auf die Attribute der Oberfläche (Eingabefelder, Auswahlelemente) zugreifen. Das Mapping von UI-Elementen zu Code im Controller passiert automatisch. Die Annotation @FXML benötigt man nur bei privaten Feldern.
- Convention over Configuration: Die fx:id in der FXML Datei muss dem Feldnamen im Controller entsprechen

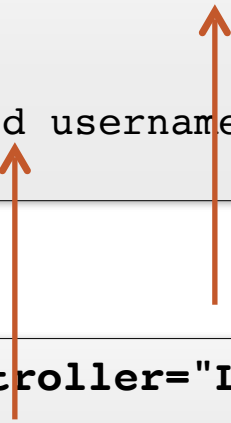
JAVA

```
public class LoginController {

    @FXML
    private TextField usernameField;
    ….
```

FXML

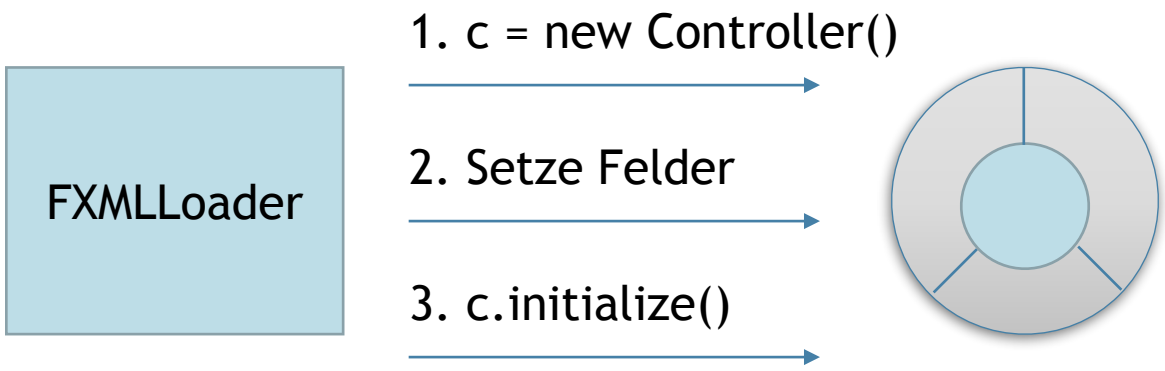
```
<GridPane … fx:controller="LoginController">
     …
    <TextField fx:id="usernameField" onAction="#doLogin" >
```



Hochschule Rosenheim
University of Applied Sciences

Controller implementieren das Initialize Interface

- Erst in der Initialize-Methode sind die injizierten Felder verfügbar
- Im Konstruktor des Controllers sind die Felder noch NULL



Hochschule Rosenheim
University of Applied Sciences

Handler Registrierung (deklarativ mit FXML, einfache Variante)

- Im Controller existiert eine Methode mit einem beliebigen Namen und einem Parameter (Event)
  - Private Methoden können per @FXML aufrufbar deklariert werden. Bei public Methoden kann die Annotation entfallen.

JAVA

```
@FXML
private void doLogin(ActionEvent e) { … }
```

- Im SceneBuilder ist der Methodenname einem Event zugeordnet

FXML

```
<TextField fx:id="usernameField" onAction="#doLogin" >
```

- Die Registrierung (addEventHandler) passiert automatisch

Hochschule Rosenheim
University of Applied Sciences

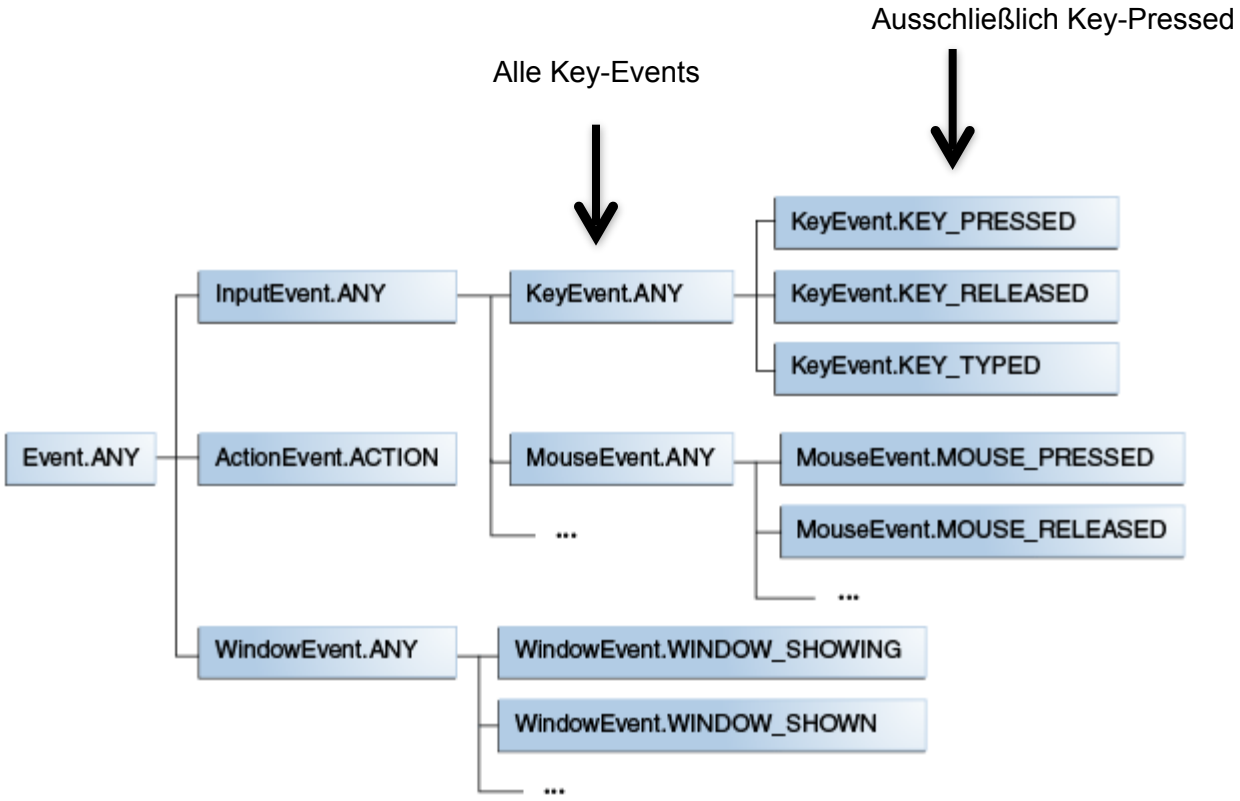
Handler Registrierung (Rich-API)

```
node.addEventHandler(MouseEvent.MOUSE_CLICKED,
      new EventHandler<Event>() {
                  @Override
                 public void handle(Event event) {
                       System.out.println("Handler2:" + event);
                 }
            }
);
```

- Die Methode **addEventFilter(EventHandler)** ist analog
- Eine Node -> viele Handler eines Typs möglich (**Multicasting**)
- Viele Nodes -> ein Handler möglich
- Handler für Gruppen von Events entlang der Vererbungshierarchie

Hochschule **Rosenheim**
University of Applied Sciences

Hierarchische Event-Typen ermöglichen feingranulare Kontrolle bei der Eventbehandlung



Hochschule Rosenheim
University of Applied Sciences

Das Interface EventTarget

- Jede Node ist ein EventTarget (Node implements EventTarget)
  - buildEventDispatchChain() bindet ein UI-Element an den Event-Dispatcher an

javafx.event

Interface EventTarget

All Known Implementing Classes:

Accordion, AnchorPane, Arc, AreaChart, Axis, BarChart, BorderPane, BubbleChart, Button, ButtonBase, Canvas, CategoryAxis, Cell, Chart, CheckBox, CheckBoxListCell, CheckBoxTableCell, CheckBoxTreeCell, CheckBoxTreeItem, CheckMenuItem, ChoiceBox, ChoiceBoxListCell, ChoiceBoxTableCell, ChoiceBoxTreeCell, Circle, ColorPicker, ComboBox, ComboBoxBase, ComboBoxListCell, ComboBoxTableCell, ComboBoxTreeCell, ContextMenu, Control, CubicCurve, CustomMenuItem, Ellipse, FlowPane, GridPane, Group, HBox, HTMLEditor, Hyperlink, ImageView, IndexedCell, Label, Labeled, Line, LineChart, ListCell, ListView, MediaView, Menu, MenuBar, MenuButton, MenuItem, Node, NumberAxis, Pagination, Pane, Parent, PasswordField, Path, PieChart, Polygon, Polyline, Popup, PopupControl, PopupControl.CSSBridge, PopupWindow, ProgressBar, ProgressBarTableCell, ProgressIndicator, QuadCurve, RadioButton, RadioMenuItem, Rectangle, Region, ScatterChart, Scene, ScrollBar, ScrollPane, Separator, SeparatorMenuItem, Service, Shape, Slider, SplitMenuButton, SplitPane, StackedAreaChart, StackedBarChart, StackPane, Stage, SVGPath, Tab, TableCell, TableColumn, TableRow, TableView, TabPane, Task, Text, TextArea, TextField, TextFieldListCell, TextFieldTableCell, TextFieldTreeCell, TextInputControl, TilePane, TitledPane, ToggleButton, ToolBar, Tooltip, TreeCell, TreeItem, TreeView, ValueAxis, VBox, WebView, Window, XYChart

---

```
public interface EventTarget
```

Represents an event target.

Method Summary

Methods

Modifier and Type	Method and Description
EventDispatchChain	buildEventDispatchChain(EventDispatchChain tail) Construct an event dispatch chain for this target.

Hochschule Rosenheim
University of Applied Sciences

Erzeugen eigener Events

- Node hat eine *fireEvent(Event)* Methode
- Während der Eventverarbeitung von Low-Level Events können High-Level Events erzeugt werden
- Low-Level Events (Basisklasse InputEvent)
  - Mouse, Keyboard, Touch …
- High-Level Events:
  - Action, Window, TreeModificationEvent …

Hochschule **Rosenheim**
University of Applied Sciences

Esg gibt verschiedene Wege Events zu behandeln

- 1) Controller und Handler in einer Klasse

```
class Controller implements EventHandler<ActionEvent> { … }
```

- 2) Controller und Handler als getrennte Klassen

```
class Controller {…}
class Handler implements EventHandler<ActionEvent>{…}
```

- 3) Handler als innere Klasse im Controller
  - static inner class: Zugriff aus dem Handler auf statische Attribute
  - Non static inner class: Zugriff auf dem Handler auf alle Attribute des Controller
- 4) Handler als anonyme Klasse
- 5) Handler als Lambda Expression (ab Java 8)

Hochschule Rosenheim
University of Applied Sciences