Technische Hochschule **Rosenheim**

Modul - Fortgeschrittene Programmierkonzepte

Bachelor Informatik

06 - GUI and JavaFX

Prof. Dr. Marcel Tilly

Fakultät für Informatik, Cloud Computing

Vers. 17.11.2021

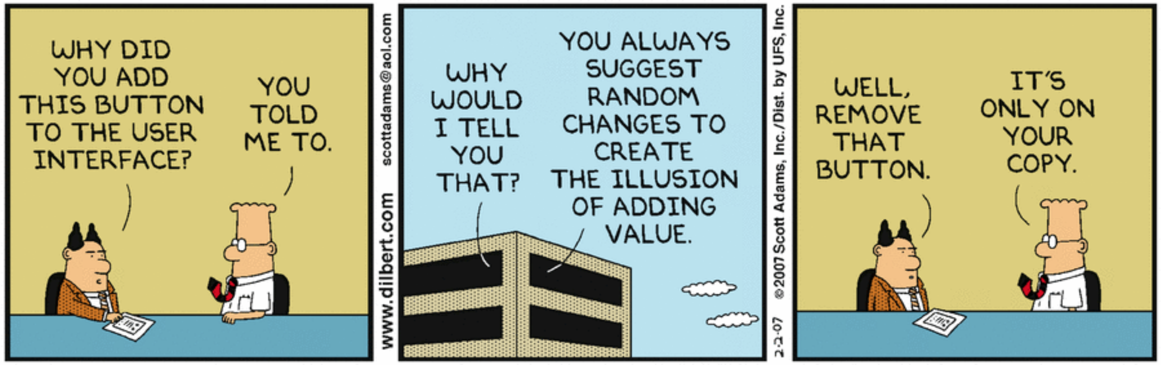
Agenda

- General basics of a GUI application
  - event model, programming approaches
- Java FX
  - Scene Graph and Layouts
- Demo 1:
  - Simple programmed interface, without GUI builder.
- Demo 2:
  - Declarative programmed interface, with GUI Builder
- Advanced
  - CSS

User Interface

Designing a good graphical user interface (GUI) is a challenge

- Intuitive operation
- Consistent design
- Maintainability
- ...


WHY DID YOU ADD THIS BUTTON TO THE USER INTERFACE?
YOU TOLD ME TO.
scottadams@aol.com
www.dilbert.com
WHY WOULD I TELL YOU THAT?
YOU ALWAYS SUGGEST RANDOM CHANGES TO CREATE THE ILLUSION OF ADDING VALUE.
©2007 Scott Adams, Inc./Dist. by UFS, Inc.
2-2-07
WELL, REMOVE THAT BUTTON.
IT'S ONLY ON YOUR COPY.

taken from https://dilbert.com/strip/2007-02-02

Graphical User Interfaces

Once upon the time ...

- **Abstract Windowing Toolkit (AWT)**
  - since JDK 1.x, 1985
  - Window and dialog elements are provided by the OS → no cross-platform, uniform look-and-feel.
  - Limited set of available dialog elements.
- **Swing**
  - Introduction with Java 2, 1998
  - Swing components use only top-level windows from the OS.
  - All other GUI elements are drawn by Swing itself.
- **JavaFX**
  - Release JavaFX 1.0, 2008
  - JavaFX will replace Swing in the medium term.
  - Direct part of Java 8

Java FX

- Java FX is a complete solution:
  - GUI components,
  - Replacement for HTML/CSS/Java Script?
  - Animations
  - Video and audio -Direct access to 2D/3D capabilities of modern graphic cards.

Note

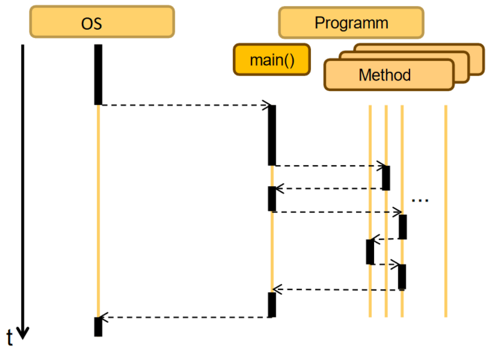
- The following introduction makes partly strong simplifications, important basics like multithreading, interfaces, etc. are still missing.
- Proper "GUI programming" can be found in the subject "Graphical User Interfaces".
- Good source from Oracle: https://docs.oracle.com/javase/8/javase-clienttechnologies.htm

Programm Execution

Technische Hochschule Rosenheim

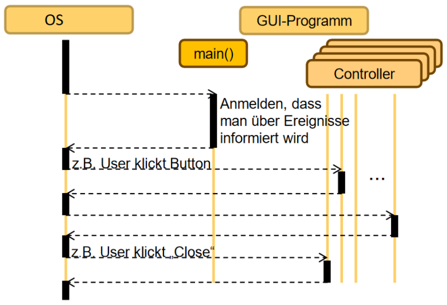
So far:

- Program starts at the beginning of main()
- Program ends at the end of main()
- Control flow is in the program all the time



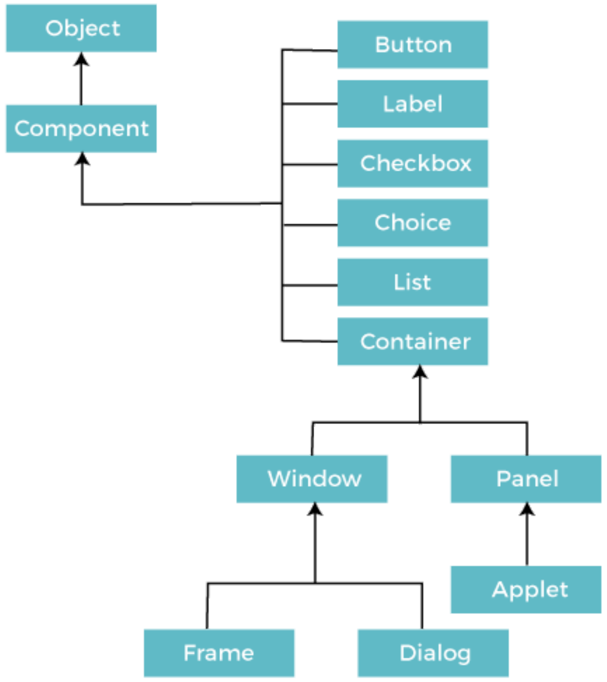
GUI program:

- Program starts at the beginning of main()
- End of main(): control flow goes to BS
- Control flow is handled by OS, GUI program is called "every now and then



Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



Java AWT Sample

Technische Hochschule Rosenheim

```java
// extending Frame class to our class AWTExample1
public class AwtTest extends Frame {

    // initializing using constructor
    public AwtTest() {
        // creating a button
        Button b = new Button("Click Me!!");
        // setting button position on screen
        b.setBounds(30,100,80,30);
        // adding button into frame
        add(b);
        // frame size 300 width and 300 height
        setSize(500,300);
        // setting the title of Frame
        setTitle("This is our basic AWT example");
        // no layout manager
        setLayout(null);
        // now frame will be visible, by default it is not visible
        setVisible(true);
    }

    public static void main(String args[]) {
        AwtTest f = new AwtTest();
    }
}
```

Communication

- Communication between OS and application program takes place in GUI programming, usually by sending messages.
  - Application is informed about events and state changes.
  - **Examples**: Mouse clicks, keyboard inputs or changes in size or position of the window, ...
- Processing of messages
  - **Event sources**: Triggers of the messages
  - Example: Button or window
  - **Event receiver (Event Listener)**: Reacts to messages
  - In order to receive messages from a specific event source, it is necessary to register with source. AWT: `button.addActionListener()`
  - In order for messages to be received, a specific method must usually be implemented. AWT: `public class ClassName implements ActionListener`

Events

- Very many different types of events.
- Most important event: ActionEvent
    - Buttons trigger an action event after the button is pressed and released.

> Outlook: Event handling is based on a modified observer pattern.

Programming Models for GUIs

- **Programmed**
  - Design of the Java user interface == Java code
  - Each GUI component must be created with new.
  - Complex interfaces consist of hard to maintain amounts of program code
  - Small change in layout → large change in source code.
- **Declarative**
  - Design and layout is described in resource file (e.g. XML).
  - Runtime environment reads file and automatically translates into mesh of GUI components and Java code.
  - Result as with programmed interfaces.
  - Easier to maintain.

Programming Models for GUIs

- **JavaFX** supports both options
- Declarative variant: **FXML**
  - FXML is based on data description language XML
  - Advantage: With XML hierarchies can be easily mapped.
  - Example: A button is located in a window
  - Class FXMLLoader can then generate a GUI at runtime.
- Declarative variants also in other frameworks:
  - Swing: Swixml, declarative approach but not very common.
  - Microsoft uses XAML

GUI with JavaFX

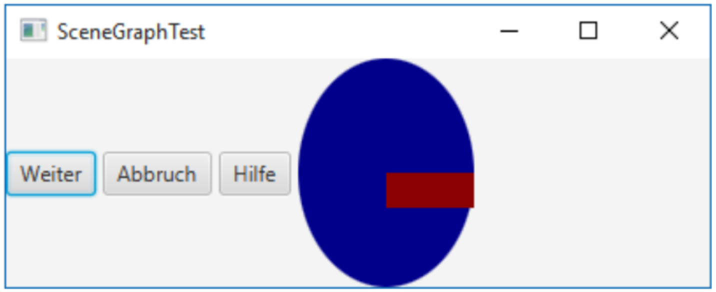
```java
public class Tada extends Application {

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Main title");
        // content
        Label label = new Label("Some wonderful text.");
        Scene scene = new Scene(label);

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

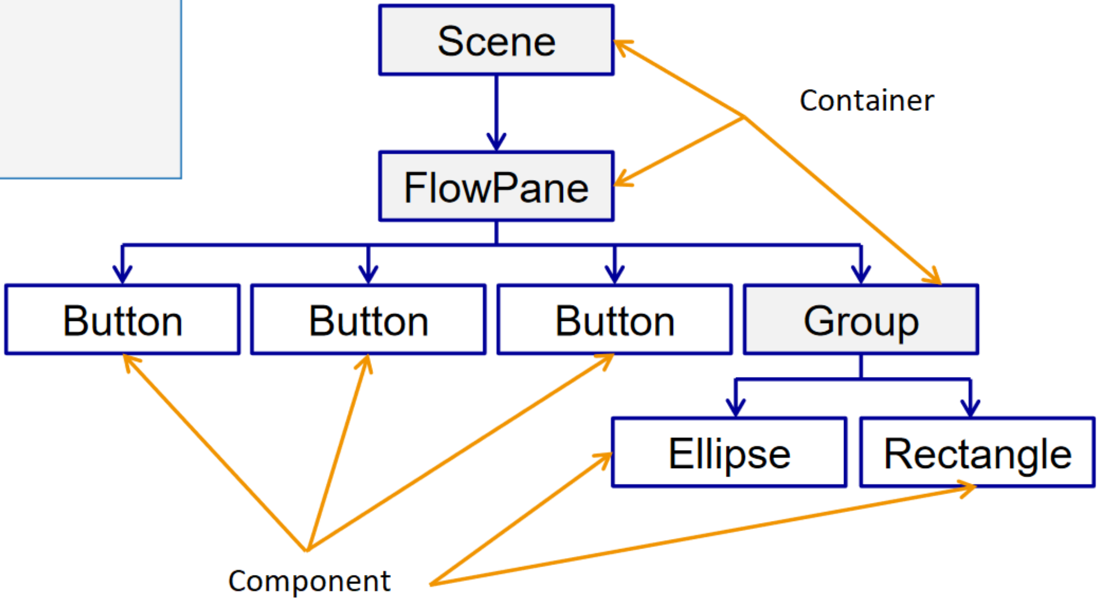
Some Explanations

- **Application**: Superclass that implements GUI functionality.
  - Provides a window with frame, system menu and standard buttons.
- Overridden method `start`:
  - Called when the application is created and specifies the contents of the window
- **Stage**: Top JavaFX container, usually 1 per window.
  - A container can hold other containers (hierarchy) or GUI components.
- **Label**(String text):Textual labe
- **Scene**: Describes the content of the window

JavaFX Scene Graph

The GUI components in a window (pane) are arranged in the form of a tree.

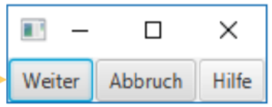




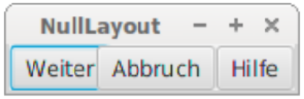
Layoutmanager

Why should GUI components not simply be positioned absolutely?

- Different platforms
- Internationalization
- Interactive resizing of dialogs



Windows 10



Linux Mint

Layout Manager

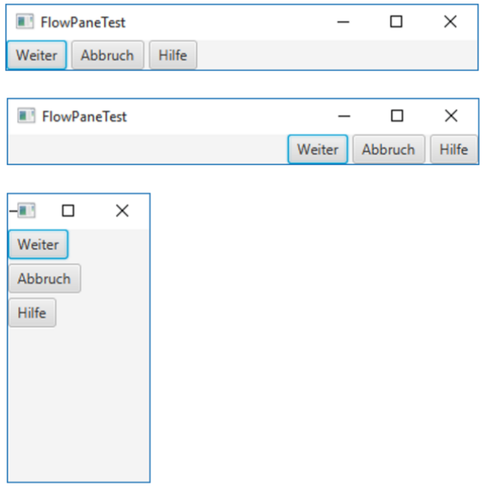
- Container that can hold other elements (e.g. buttons).
- Relative positioning of elements within a container
- Takes care of any necessary scaling of elements.
- In Java FX there are several layout managers.

> Never create a GUI without a layout manager!

Layout: FlowPane

Example: Layout Manager FlowPane

- Placement one after the other in the preferred size of components.
- Spacing between components and their arrangement can be specified in the layout manager constructor.
- Ex: Orientation.VERTICAL or Pos.CENTER_LEFT



Layout: FlowPane

Some Code:

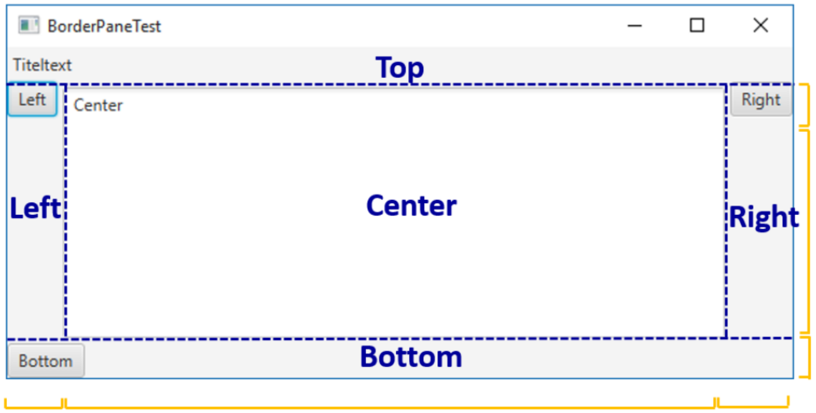
```
FlowPane pane = new FlowPane();
pane.setOrientation(Orientation.VERTICAL);
pane.getChildren().add(label);
```

```
FlowPane pane = new FlowPane();
pane.setOrientation(Orientation.HORIZONTAL);
pane.getChildren().add(label);
```

Layout: Borderpane

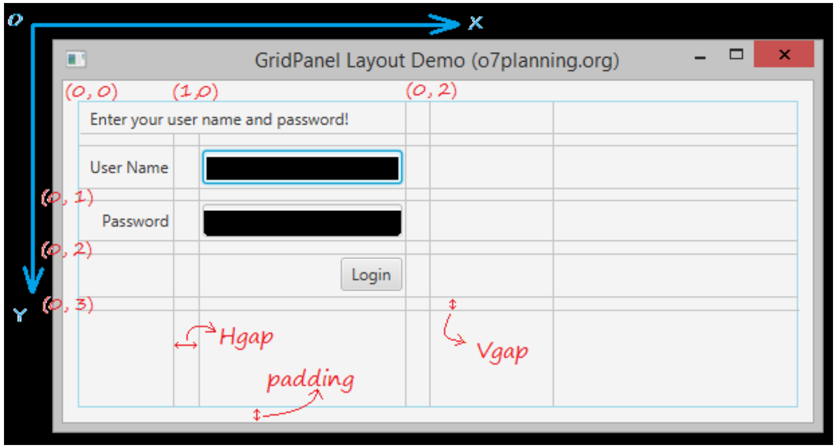
5 areas, into each of which 1 component can be entered.

```
BorderPane borderPane = new BorderPane();

BorderPane.setAlignment(label, Pos.CENTER_LEFT);
BorderPane.setMargin(label, new Insets(4.0, 4.0, 4.0, 4.0));
borderPane.setBottom(label);
```



Layout: Gridpane

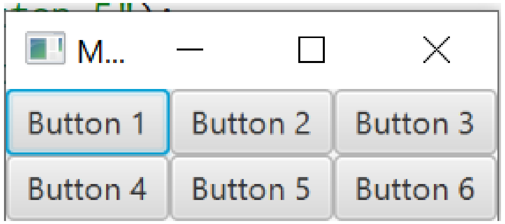
- A JavaFX GridPane is a layout component which lays out its child components in a grid.
- The size of the cells in the grid depends on the components displayed in the GridPane, but there are some rules.
- All cells in the same row will have the same height, and all cells in the same column will have the same width.
- Different rows can have different heights and different columns can have different widths



Layout: Gridpane

Code

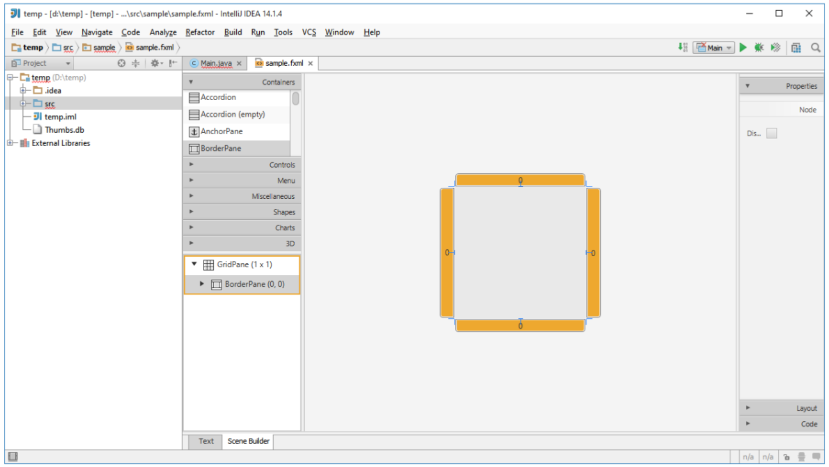
```
Button button1 = new Button("Button 1");
Button button2 = new Button("Button 2");
Button button3 = new Button("Button 3");
Button button4 = new Button("Button 4");
Button button5 = new Button("Button 5");
Button button6 = new Button("Button 6");

GridPane pane = new GridPane();

pane.add(button1, 0, 0, 1, 1);
pane.add(button2, 1, 0, 1, 1);
pane.add(button3, 2, 0, 1, 1);
pane.add(button4, 0, 1, 1, 1);
pane.add(button5, 1, 1, 1, 1);
pane.add(button6, 2, 1, 1, 1);
```



Scene Builder

Practice: Assembling the GUI with GUI Builder

- Drag&Drop
- Generates FXML file
- This can then be loaded at runtime (declarative style)
- GUI Builder: Scene Builder works together with IntelliJ.



Hello World

```
public class HelloWorld extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        primaryStage.setTitle("Hello World");

        FlowPane flowPane = new FlowPane(Orientation.VERTICAL);
        Label label = new Label("Das ist ein Label");
        Button button = new Button("OK");
        flowPane.getChildren().addAll(label, button);
        flowPane.setAlignment(Pos.TOP_LEFT);

        Scene scene = new Scene(flowPane);

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

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

Class Diagrams

As anonymous inner class:

```
Label label = new Label("Das ist ein Label");
Button button = new Button("OK");
button.setOnAction(new EventHandler<ActionEvent>() {
                       @Override
                       public void handle(ActionEvent event) {
                           label.setText("Hello World!");
                       }
                   });
```

or as **lambda**

```
Label label = new Label("Das ist ein Label");
Button button = new Button("OK");
button.setOnAction(event -> label.setText("Hello World!"));
```

Using FXML

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

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<GridPane alignment="center" hgap="10" vgap="10"
    xmlns="http://javafx.com/javafx/11.0.1"
    xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller">
   <children>
       <FlowPane>
           <Label text="Hello World!"/>
           <Button mnemonicParsing="false" text="Button" />
       </FlowPane>
   </children>
</GridPane>
```

Use FXML in Code

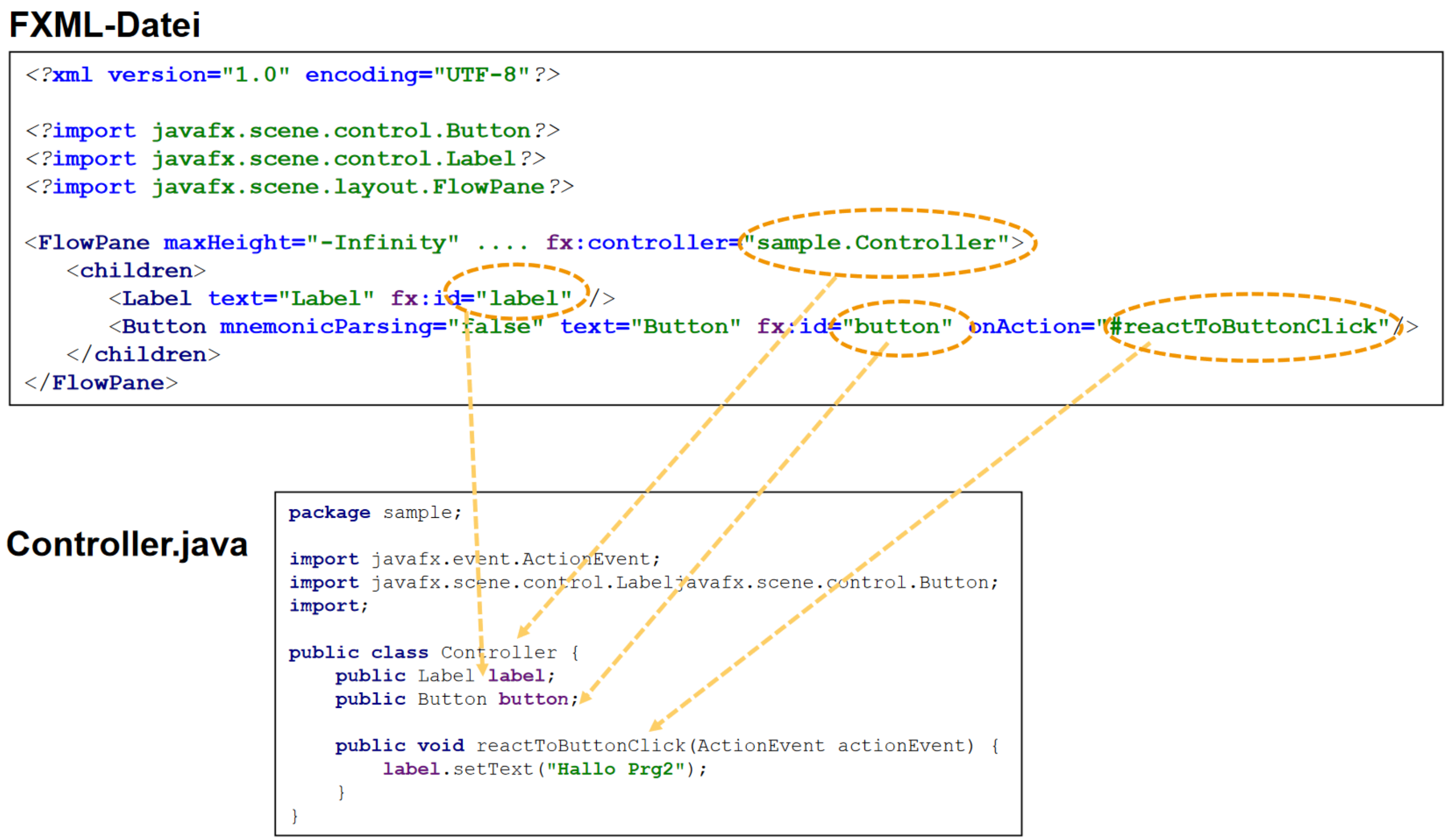
Use via

```
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Hello World");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }


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

Handle Events

- **Recommended**: Reaction to events within separate controller class
  - Not in `Main.java`
  - Here: `Controller.java`
  - Code can be generated automatically for the most part.
- Possible approach: Edit FXML file in IntelliJ
  - Set reference to controller class
    - `<FlowPane ... fx:controller=sample.Controller">`
  - Assign a unique fx-id for GUI components to be accessed with custom code, e.g.:
    - `<label text="label" fx:id="label" />`
    - `<Button text="Button" fx:id="button"/>`
  - Code generation for accessing GUI components within the controller class.
  - Click on GUI element in FXML file, then Alt+Enter, then "Create Field".

Controller FXML



FXML-Datei

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.FlowPane?>

<FlowPane maxHeight="-Infinity" .... fx:controller="sample.Controller">
   <children>
      <Label text="Label" fx:id="label" />
      <Button mnemonicParsing="false" text="Button" fx:id="button" onAction="#reactToButtonClick"/>
   </children>
</FlowPane>
```

Controller.java

```
package sample;

import javafx.event.ActionEvent;
import javafx.scene.control.Labeljavafx.scene.control.Button;
import;

public class Controller {
    public Label label;
    public Button button;

    public void reactToButtonClick(ActionEvent actionEvent) {
        label.setText("Hallo Prg2");
    }
}
```

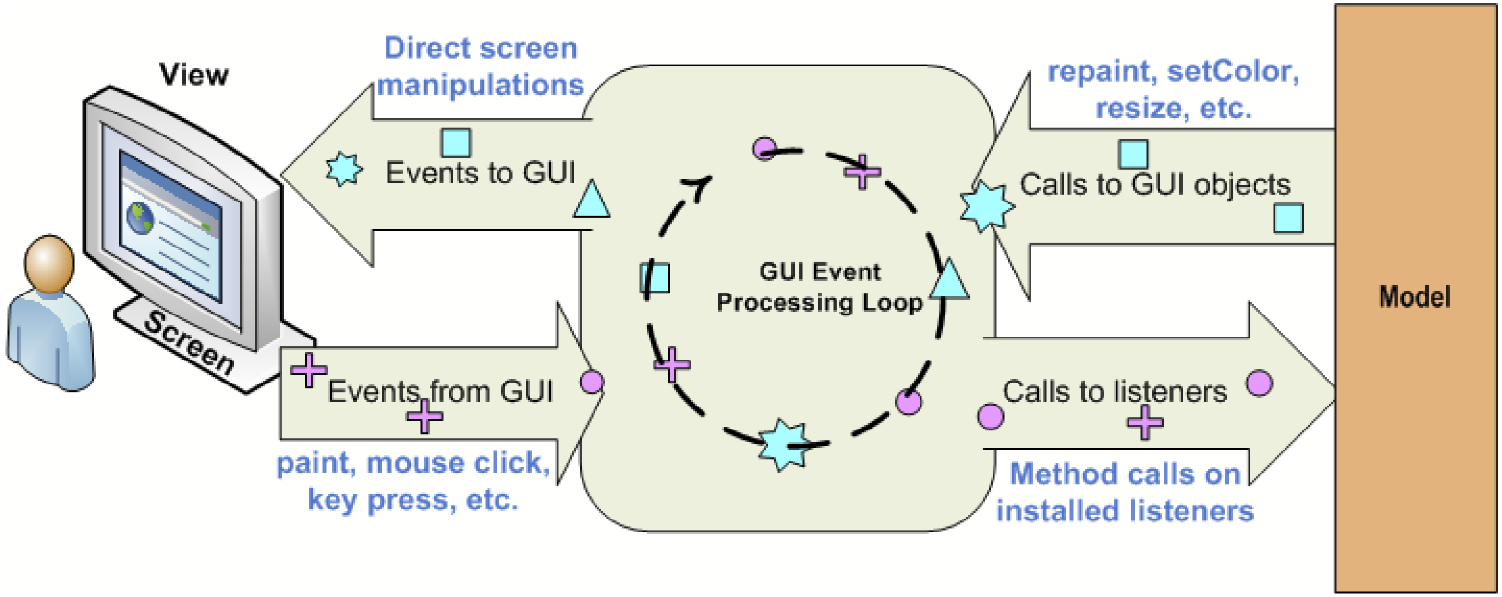
Cascading Style Sheets (CSS)

- Adjustment of the display of control elements.
- Directly in the code or in the Scene Builder or via CSS files
- Advantage of CSS files
    - Designers can change the design of a GUI without knowing or changing the implementation.

> Example:
>
> - Change background color of FlowPane in view.
> - Add attribute `stylesheets="@view2.css"` in the FXML file.

GUI Event Schedule

- Typically the UI is single threaded!
- Updating the UI from another thread is tricky!


View
Screen
Direct screen manipulations
Events to GUI
Events from GUI
paint, mouse click, key press, etc.
GUI Event Processing Loop
repaint, setColor, resize, etc.
Calls to GUI objects
Calls to listeners
Method calls on installed listeners
Model

Updating the UI Thread

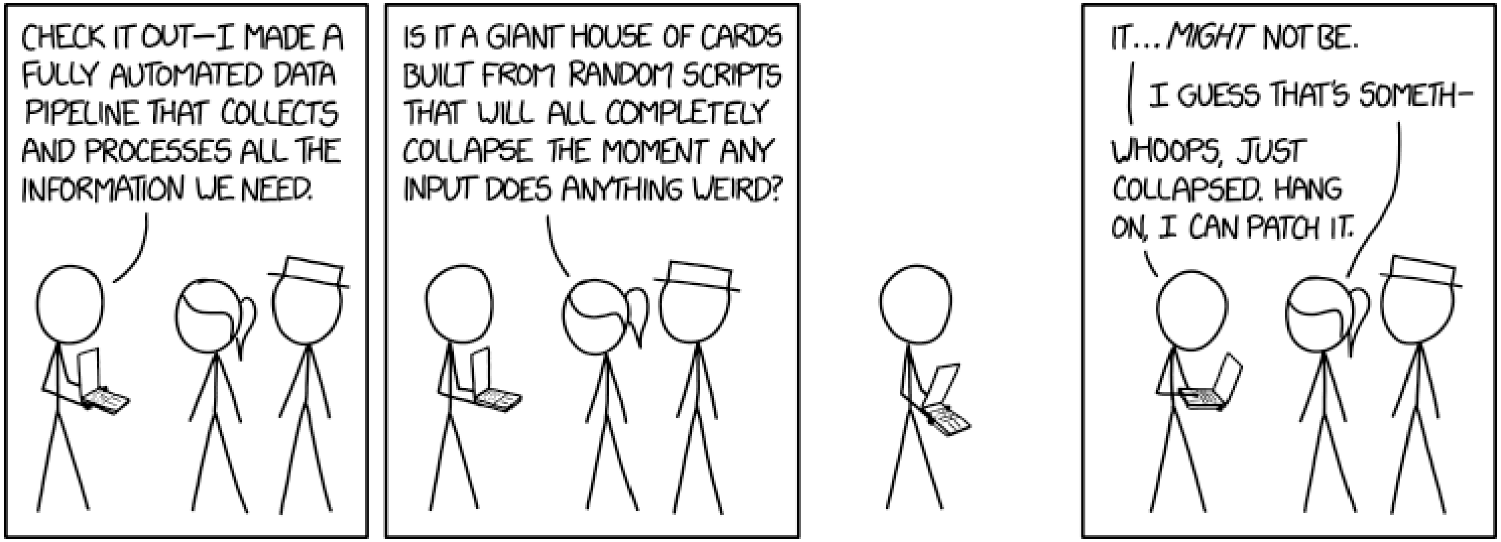
- Using `Platform.runLater(...)` is an appropriate approach for this.
- The trick to avoid flooding the FX Application Thread is to use an atomic variable to store the value you're interested in.
- In the `Platform.runLater` method, retrieve it and set it to a sentinel value.
- From your background thread, update the atomic variable, but only issue a new `Platform.runLater` if it's been set back to its sentinel value.

```
        Platform.runLater(new Runnable() {
                      @Override
                      public void run() {
                          long value = 0;
                          label.setText(value + "");
                      }
                  });
```

Summary

- GUI programming with Java
  - ... is hell and not super clear
- Basics, components and events
- Layout manager
- FXML and scene builder

Final Thought!


CHECK IT OUT—I MADE A FULLY AUTOMATED DATA PIPELINE THAT COLLECTS AND PROCESSES ALL THE INFORMATION WE NEED.
IS IT A GIANT HOUSE OF CARDS BUILT FROM RANDOM SCRIPTS THAT WILL ALL COMPLETELY COLLAPSE THE MOMENT ANY INPUT DOES ANYTHING WEIRD?
IT... *MIGHT* NOT BE.
I GUESS THAT'S SOMETH–
WHOOPS, JUST COLLAPSED. HANG ON, I CAN PATCH IT.