

Modul 120

|  |
| --- |
| Benutzerschnittstellen implementieren |
| Dokumentation |

# 1 Theorie

## Begriffe

**Benutzerschnittstelle** // User-Interface

**Software-Ergonomie** // menschliche gestalltung des "Software-Arbeitspaltzes"

**Gestaltungsregelwerk** // Style Guide

*Was?*

Richtlinien und Tipps zur Gestaltung

*Wofür?*

Erscheinungsbild und bedienungseigenschaften (Look an feel) anwendungsübergreifend

einheitlich und effizient gestaltet.

*Welche?*

herstellerspezifisch

unternehmenseigene (corporate Design - CD)

**Interaktive Benutzerschnittstellen**

* mittels Kommandosprachen
* mittels Menüs
* mittels direkter Manipulation (Fenster-orientiert)
* mittels Bildschirmformularen

**DIALOGFÜHRUNG: KONSOLENANWENDUNGEN**

*Pro*

* schnelle effiziente arbeit möglich
* sehr flexibel
* bietet zahlreiche optionen und parameter

*Kontra*

* hoher lernaufwand
* fehleranfällig
* Auswirkung von Fehler können erheblich sein

**DIALOGFÜHRUNG: MENU**

*Pro*

* Schnell verstanden und leicht zu erlernen
* Kein Syntaxfehler (vertippen nicht mˆglich)
* Wenig Aufwand (Eingabe)
* Strukturierung
* Kleinerer lernaufwand

*Kontra*

* Platzbedarf auf dem Bildschirm
* Parameterübergabe aufwendig (Dialogfenster)
* Unübersichtlich und aufwendig (kleine Schritte), wenn viele Alternativen

**DIALOGFÜHRUNG: DIREKTE MANIPULATION**

Wesentliche Objekte sind sichtbar, greifbar. Die Rückmeldung über Erflog (Misserfolg) einer Aktion muss unmittelbar erfolgen.

*Pro*

* Ergebnis direkt erkennbar (sichtbar)
* Kleiner lernaufwand
* Kein Syntaxfehler (vertippen nicht mˆglich)

*Kontra*

* Symbole nicht immer eindeutig --> Fehlhandlungen
* Platzverbrauch

**DIALOGFÜHRUNG: FORMULARE**

Für Eingabe von strukturierten Daten einen Datenbestand (Datenbank).

*Pro*

* Erlernen einfach / intuitiv (Analogie zu Papierformularen)
* Schnell und effizient

**WIMP**

* **W**indwos
* **I**cons
* **M**enus
* **P**ointer

**What You See Is What You Get (WYSIWYG)**

* Alles, was der Benutzer sieht, ist auch vorhanden.
* Nichts von Bedeutung bleibt dem Benutzer verborgen.

**Metapher**

* Abstrakter, unbekannter Begriff mit einer bildhaften Übertragung als Vergleich "übersetzen".
* (Ordner, Blatt, Papierkorb, fliegen, alle Icons)

**Gross bis Klein**

* Stage
* Scene
* Pane
* Label, Button, usw.

**Fensteraufbau (Layout)**

* VBox und HBox // Vertikal oder Horizonzal nebeneinander platziert
* BorderPane // Top, Center, Bottom, Left, Right
* AnchorPane // Einen bestimmten Punkt fixieren > Unten Rechts
* FlowPane // Stellt so viel nebeneinander dar, solang es kein Platz hat > Responsive
* StackPane // Stellt Sachen übereinander dar > vorne oder hinten
* GridPane // Reihen und Spalten, wie eine Tabelle
* ButtonBar // Buttons - Speichern & Abbrechen werden in einer bestimmten Reihenfolge platziert
* Unmanaged Nodes

**Grundsätze der Dialoggestalltung (1/3)**

nach DIN EN ISO 9241 Teil 110 (bzw. 10)

* Aufgabenangemessenheit
  + Der Benutzer wird unterstützt, seine Arbeitsaufgaben effektiv und effizient zu erledigen.
* Selbstbeschreibungsfähigkeit
  + Jeder einzelne Dialogschritt ist verstä‰ndlich oder wird dem Benutzer auf Anfrage erklärt.
* Steuerbarkeit
  + Der Benutzer ist in der Lage, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.

**Grundsätze der Dialoggestalltung (2/3):**

nach DIN EN ISO 9241 Teil 110 (bzw. 10)

* Erwartungskonformität
  + Der Dialog ist konsistent und entspricht der Erfahrung des Benutzers und allgemein anerkannten Konventionen.
* Fehlertoleranz
  + Das beabsichtigte Arbeitsergebnis kann troz erkennbar fehlerhafter Eingaben entweder mit keinem ider minimalem korrekturaufwand durch den Benutzer erreicht weden

**Grundsätze der Dialoggestalltung (3/3):**

nach DIN EN ISO 9241 Teil 110 (bzw. 10)

* individualisierbarkeit
  + Das Dialogsystem lässt Anpasssungen an die individuelle
* Lernförderkeit

**MVC (Model-View-Controller)**

* Model // enthält und verarbeitet die Daten
* View // visualisiert die Modelldaten
* Controller // nimmt Benutzeraktionen entgegen und sagt dem Modell welche Daten zu verändern sind.

# 2 Codebeispiele

Ereignisse (Lösung 2)

**import** javafx.application.Application; **import** javafx.event.ActionEvent;

**import** javafx.geometry.Insets; **import** javafx.geometry.Pos;

**import** javafx.scene.Scene; **import** javafx.scene.control.Button;

**import** javafx.scene.control.Label; **import** javafx.scene.layout.BorderPane;

**import** javafx.scene.layout.HBox; **import** javafx.scene.text.Font;

**import** javafx.stage.Stage;

**public** **class** Inkrement **extends** Application {

Label lblZahl;

Double btnBreite = 110.0;

@Override

**public** **void** start(Stage primaryStage) {

primaryStage.setTitle("Auf und Ab!");

BorderPane root = **new** BorderPane();

Scene scene = **new** Scene(root,380,100);

HBox hboxBottom = **new** HBox();

hboxBottom.setAlignment(Pos.***CENTER***);

hboxBottom.setSpacing(10); // Lösung Ereignisse 2.docx / Teil 2 ODER weiter oben --> new HBox(10)

hboxBottom.setPadding(**new** Insets(0, 0, 10, 0));

// Methode 5 in "Ereignisse.docx"

Button btnInkrementieren = **new** Button("\_Inkrementieren"); // Lösung Ereignisse 2.docx

btnInkrementieren.setMnemonicParsing(**true**); / Lösung Ereignisse 2.docx / Teil 2

btnInkrementieren.setDefaultButton(**true**); // Lösung Ereignisse 2.docx / Teil 2

btnInkrementieren.setOnAction((event) -> lblZahl.setText("" + (Integer.*parseInt*(lblZahl.getText()) + 1)) );

hboxBottom.getChildren().add(btnInkrementieren);

// Methode 6 in "Ereignisse.docx"

Button btnReset = **new** Button("\_Reset");

btnReset.setMinWidth(btnBreite);

btnReset.setMnemonicParsing(**true**);

btnReset.setOnAction((event) -> resetPressed(event));

hboxBottom.getChildren().add(btnReset);

// Methode 6 in "Ereignisse.docx"

Button btnDekrementieren = **new** Button("\_Dekrementieren");

btnDekrementieren.setMinWidth(btnBreite);

btnInkrementieren.setMnemonicParsing(**true**);

btnDekrementieren.setOnAction((event) -> dekrementierenPressed(event));

hboxBottom.getChildren().add(btnDekrementieren);

lblZahl = **new** Label("0");

lblZahl.setFont(**new** Font(20));

root.setCenter(lblZahl);

root.setBottom(hboxBottom);

primaryStage.setScene(scene);

primaryStage.show();

}

**private** **void** dekrementierenPressed(ActionEvent event) {

lblZahl.setText("" + (Integer.*parseInt*(lblZahl.getText()) - 1));

}

**private** **void** resetPressed(ActionEvent event) {

lblZahl.setText("" + 0);

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}}

# Methode 1 - Event Handling

**package** ereignisse;

**import** javafx.application.Application;

*usw.*

**public** **class** ereignis1 **extends** Application {

@Override

**public** **void** start(Stage primaryStage) {

BorderPane root = **new** BorderPane();

Scene scene = **new** Scene(root,200,100);

HBox hboxBottom = **new** HBox();

hboxBottom.setAlignment(Pos.***CENTER***);

hboxBottom.setPadding(**new** Insets(0, 0, 10, 0));

Button btnInkrementieren = **new** Button("Inkrementieren");

hboxBottom.getChildren().add(btnInkrementieren);

btnInkrementieren.setOnAction(**new** Ereignishandler());

Label lblZahl = **new** Label("0");

lblZahl.setFont(**new** Font(20));

root.setCenter(lblZahl);

root.setBottom(hboxBottom);

primaryStage.setScene(scene);

primaryStage.show();

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

**class** Ereignishandler **implements** EventHandler {

@Override

**public** **void** handle(Event arg0) {

System.***out***.println("Hello Bassersdorf");

}

}

# Methode 2 - Event Handling

Ereignisbehandlung mit **anonymer Klasse**

**package** ereignisse;

**import** javafx.application.Application;

**import** javafx.event.ActionEvent;

**import** javafx.event.EventHandler;

**import** javafx.geometry.Insets;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Button;

**import** javafx.scene.control.Label;

**import** javafx.scene.layout.BorderPane;

**import** javafx.scene.layout.HBox;

**import** javafx.scene.text.Font;

**import** javafx.stage.Stage;

**public** **class** ereignis1 **extends** Application {

@Override

**public** **void** start(Stage primaryStage) {

BorderPane root = **new** BorderPane();

Scene scene = **new** Scene(root,200,100);

HBox hboxBottom = **new** HBox();

hboxBottom.setAlignment(Pos.***CENTER***);

hboxBottom.setPadding(**new** Insets(0, 0, 10, 0));

Button btnInkrementieren = **new** Button("Inkrementieren");

hboxBottom.getChildren().add(btnInkrementieren);

btnInkrementieren.setOnAction(**new** EventHandler<ActionEvent>() {

**public** **void** handle(ActionEvent event) {

System.***out***.println("Hello Bassersdorf");

}

});

Label lblZahl = **new** Label("0");

lblZahl.setFont(**new** Font(20));

root.setCenter(lblZahl);

root.setBottom(hboxBottom);

primaryStage.setScene(scene);

primaryStage.show();

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

# Methode 3 - Event Handling

Ereignisbehandlung mit **Lambda expression** (ab Java 8)

Mit separater Methode

**package** ereignisse;

**import** javafx.application.Application;

*usw.*

**public** **class** ereignis1 **extends** Application {

@Override

**public** **void** start(Stage primaryStage) {

BorderPane root = **new** BorderPane();

Scene scene = **new** Scene(root,200,100);

HBox hboxBottom = **new** HBox();

hboxBottom.setAlignment(Pos.***CENTER***);

hboxBottom.setPadding(**new** Insets(0, 0, 10, 0));

Button btnInkrementieren = **new** Button("Inkrementieren");

hboxBottom.getChildren().add(btnInkrementieren);

btnInkrementieren.setOnAction(event -> buttonPressed(event));

Label lblZahl = **new** Label("0");

lblZahl.setFont(**new** Font(20));

root.setCenter(lblZahl);

root.setBottom(hboxBottom);

primaryStage.setScene(scene);

primaryStage.show();

}

**private** **void** buttonPressed(ActionEvent event) {

System.***out***.println("Hello Welt");

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}

}