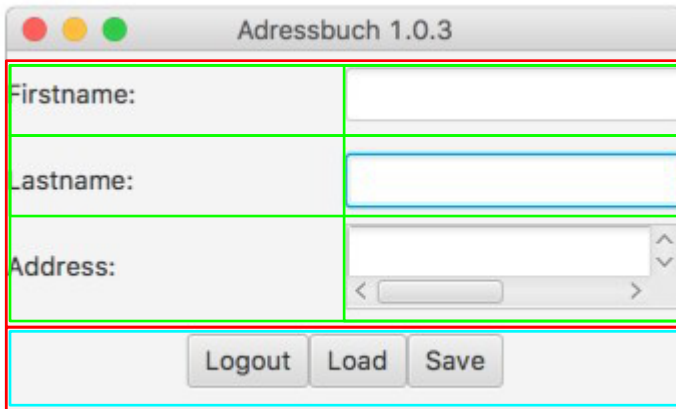
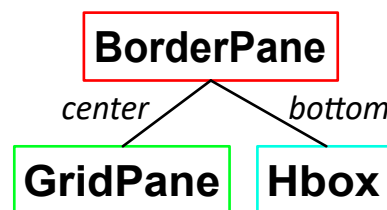


Nachdenkzettel zu GUI Programmierung

1. Welche Layout-Manager würden Sie verwenden, um das folgende Fenster zu realisieren (Abbildung 1)? Zeichnen Sie die Layout-Manager direkt in die Abbildung. Tipp: Folgende Layout-Manager wurden in der Vorlesung besprochen: `BorderPane`, `HBox`, `VBox`, `StackPane`, `GridPane`, `FlowPane`, `TilePane`.



Rot: `BorderPane`
Grün: `GridPane`
Blau: `Hbox`



2. Eventhandler

a) Geben Sie ein Beispiel für die zwei Formen von `JavaFXEventHandler` (alt und seit Java8)

Alt:

```
EventHandler e = new EventHandler< ActionEvent >(){
    @Override
    public void handle(ActionEvent event){
        method();
    }
}
```

Java 8 (Lambda):

```
EventHandler< ActionEvent > e = event -> {
    method();
}
```

b) Welche Möglichkeit gibt es in JavaFX, nach einer bestimmten Zeit einen Handler-Callback zu bekommen? Wozu könnte man das verwenden?

Um einen Handler-Callback zu bekommen, kann man in die `handle(ActionEvent event)` Methode eine z.B. eine `set` Methode schreiben, welche als Parameter einen bestimmten Wert weitergibt, welcher von verschiedenen Aspekten des Handlers und jetzigen Zeitpunkts abhängt.

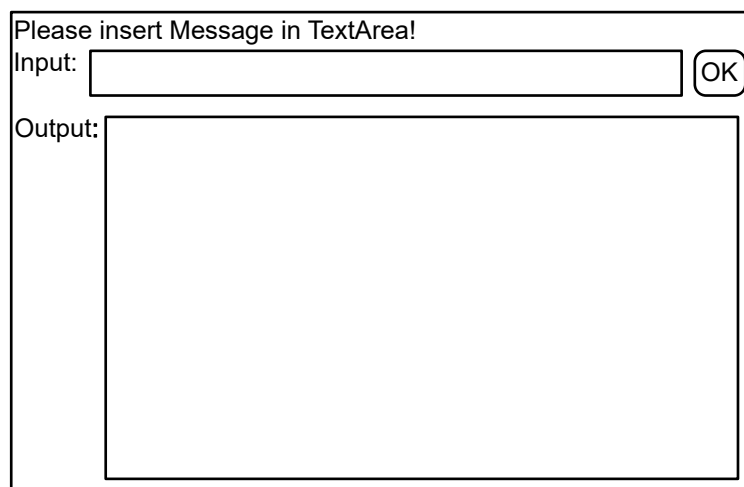
Ein Beispiel wäre z.B. das ein `EventHandler` auf den `EventType MOUSE_DRAGGED` einer Node gelegt wird, welche diese dann verschiebt. Als Parameter werden in der `handle()` - Methode die momentanen Koordinaten der Maus (`event.getX()`...) der `setTranslateX()` und `setTranslateY()` der Node übergeben, was zur Folge hat, dass die Node sich bewegt.

3. GUI: 1. Zeichnen Sie das Fenster mit Inhalt, das durch den JavaFX Code beschrieben wird. 2. Was macht der Handler?

```
public class LayoutExample extends Application
{
    private TextField inputArea = new TextField();
    private TextArea outputArea = new TextArea();
    public static void main(String[] args)
    {
        Application.launch(args);
    }
    @Override
    public void start(Stage stage)
    {
        Label headerLbl = new Label("Please insert Message in TextArea!");
        Label inputLbl = new Label("Input: ");
        Label outputLbl = new Label("Output: ");
        Button okBtn = new Button("OK");
        HBox output = new HBox();
        output.getChildren().addAll(outputLbl, outputArea);

        okBtn.addEventHandler(MouseEvent.MOUSE_CLICKED,
            event -> outputArea.appendText("You: " + inputArea.getText() + "\n"));

        BorderPane root = new BorderPane();
        root.setTop(headerLbl);
        root.setRight(okBtn);
        root.setBottom(output);
        root.setLeft(inputLbl);
        root.setCenter(inputArea);
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("SE2 Nachdenkzettel GUI");
        stage.show();
    }
}
```



Please insert Message in TextArea!

Input: OK

Output:

Der Handler bewirkt, dass sobald auf den Button "OK" mit der Maus geklickt (also Maus drücken und loslassen auf dem Button) wird, der Text, welcher im TextField "inputArea" an die TextArea "outputArea" in der Form: "You: " + inputArea.getText() + "\n" angeheftet wird.

4. Strukturen: Beschreiben Sie das Dokument einmal als Baum aus graphischen Nodes und einmal in Form eines serialisierten Textes (wie Html/xml). Die Tags dafür können Sie frei erfinden

DocScope – publizieren mit Zukunft



+++ einfach in der Anwendung

DocScope bringt Ihre Kommunikationsprojekte nach vorne – individuell, innovativ, international

DocScope ist eine regelbasierte Software, die auf vorhandene Datenbanken oder Warenwirtschaftssysteme in Ihrem Unternehmen zugreift.

Ihr Corporate Design für Dokumente ist nichts anderes als eine umfangreiche und präzise Sammlung an Layoutregeln. DocScope übersetzt diese Gestaltungsregeln in eine XML-Notation und führt sie „auf Knopfdruck“ exakt und vollautomatisch aus.

Die Arbeit mit DocScope bedeutet: Änderungen einer Designregel werden in allen Dokumenten und Dokumenttypen automatisch umgesetzt. Auch ältere Versionen von Dokumenten werden von diesen Änderungen erfasst. Damit ist DocScope prädestiniert, die dynamische und nicht zuletzt durch die Expansion in neue Märkte getriebene Entwicklung eines Corporate Designs nachhaltig zu unterstützen. Die Arbeit mit DocScope verhindert darüber hinaus Flüchtigkeitsfehler oder das „bewusste“ Übersehen von CD-Regeln.

Sobald Sie „den Knopf“ drücken und DocScope starten, werden die relevanten Daten (Bilder, Grafiken, Texte, etc.) abgerufen und zu 100% gemäß dem hinterlegten Regelwerk auf den Seiten platziert – inklusive Schrift- und

Tabellensatz und bei Einhaltung aller definierten Stilvorlagen, in jeder beliebigen Sprache (wie z. B. Chinesisch, Arabisch, Japanisch oder Russisch). Ist etwa ein Bildstandsabgleich gewünscht, platziert DocScope die Bilder erst, nachdem alle relevanten Sprachen auf ihre Laufänge hin überprüft und optimiert wurden. Innerhalb kürzester Zeit entsteht so das komplette Print-Produkt.

DocScope erzeugt auf einem gewöhnlichen Rechner (z. B. 1 GHz-Prozessor, 1 GB Speicher) etwa 50 bis 60 Seiten pro Minute! Das Programm arbeitet folglich so schnell, dass Sie selbst kurz vor der Drucklegung noch problemlos auch hochkomplexe Dokumente vollständig neu erzeugen können. Damit reduzieren Sie Fehlerquellen und Prozesskosten.

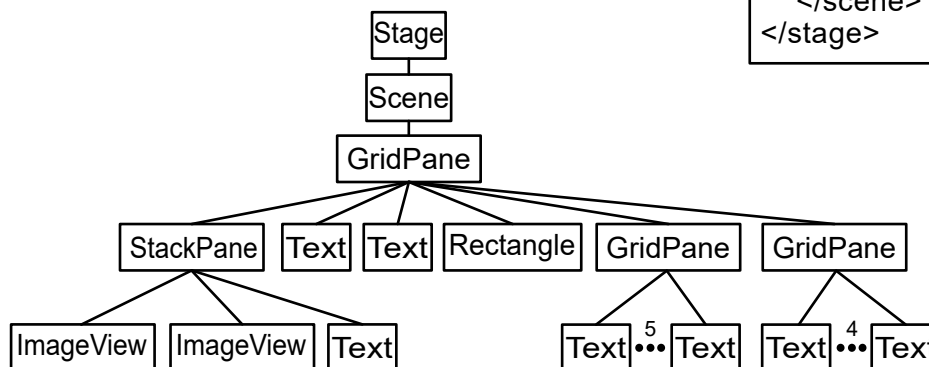
Mit DocScope ist auch die Produktion personalisierter Medien unkompliziert und wirtschaftlich möglich. Ob zielgruppenspezifischer Produktkatalog, individuelle Kundenzeitschrift oder persönliches Angebot – mit DocScope liefern Sie exakt diejenigen Informationen, die Ihre Zielgruppen oder Kunden wünschen. Besser lässt sich Kundenbindung nicht umsetzen. DocScope eröffnet Ihnen neue Dimensionen für das Direct-to-Customer-Marketing!

Sollen individuell gestaltete Seiten oder Objekte aus gängigen Anwendungen, wie z.B. InDesign, eingepflegt werden, ist auch dies problemlos möglich.

```

<stage>
  <scene>
    <gridpane>
      <stackpane>
        <imageview></imageview>
        <imageview></imageview>
        <text></text>
      </stackpane>
      <text></text>
      <text></text>
      <rectangle></rectangle>
    </gridpane>
    <gridpane>
      <text></text>
      <text></text>
      <text></text>
      <text></text>
      <text></text>
    </gridpane>
    <gridpane>
      <text></text>
      <text></text>
      <text></text>
      <text></text>
    </gridpane>
  </gridpane>
</scene>
</stage>

```



5. GUI Thread und andere...

a) Sie wollen Ihre JavaFX Application Unit Testen. Was für ein Problem tritt auf?

Tipp: <https://medium.com/information-and-technology/test-driven-development-in-javafx-with-testfx-66a84cd561e0>

Mit dem normalen JUnit framework, lassen sich GUI Elemente nicht testen, da nicht auf den GUI Thread zugegriffen werden kann, da dieser nicht gestartet wird. Man kann sich jedoch mit der Erweiterung TestFX von JUnit behelfen und darüber Tests schreiben, welche GUI Elemente testen.

b) Sie müssen Dinge im Background machen und können den Main GUI Thread nicht dafür nehmen? Sie müssen Daten zwischen GUI und Restapplikation austauschen?

→ Was macht die Task Class in JavaFX?

Tipp: <https://docs.oracle.com/javafx/2/threads/jfxpub-threads.htm>

Die Task Class in JavaFX extended die standard Java FutureTask Class, welche zum erstellen von Tasks zuständig ist. Das spezielle an den Tasks in JavaFX ist, dass sie zum einen das Worker interface implementieren, mit dem einzelne Stadien der Task abgefragt werden können und zum anderen versichern sie, dass Veränderungen, welche auf dem JavaFX Application Thread ausgeführt werden sollten (wie z.B. EventHandler) auch auf diesem passieren. Außerdem können mit updateProgress(), updateMessage() und updateTitle() in der call() method die korrespondierenden properties im JavaFX Thread geändert werden. So können Daten zwischen GUI und Restapplikation ausgetauscht werden.