**Nachdenkzettel zu GUI Programmierung**

1. Welche Layout-Manager würden Sie verwenden, um das folgende Fens-

ter 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.





2. Eventhandler

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

alt: lineare Programmierung

main 🡪 initialisiere alles 🡪 frage nach user input (scanner) 🡪 male auf den Bildschirm 🡪 sleep(5) 🡪 frage user nach input (scanner) 🡪 male auf den Bilschirm 🡪 …

=> Programm steuert den User linear

Seit Java8: event-getriebene Programmierung

Main 🡪 initialisiere alles: Action Handler werden bei GUI Elementen registriert 🡪 GUI malt erste Szene 🡪 User klickt: GUI entdeckt Button X geklickt. Im Controller: Action Handler dafür wird aufgerufen. Handler ruft game.selectGame("4Gewinnt") im Model. Oder setzt Spielerzahl auf 2. Oder ruft createTask(neue Task) auf. Oder... 🡪 sleep (5) fällt weg! Stattdessen wird ein Timer callback im GUI registriert, der nach 5 Sekunden die Nachricht auf dem Bildschirm löscht 🡪 User klickt wieder auf etwas im Gui und löst Event aus. Oder bewegt die Maus. Oder gibt etwas an der Tastatur ein... 🡪 GUI löst entsprechende Actions aus und das Model wird modifiziert. Anschließend wird die View aktualisiert und der User sieht den neuen Zustand. 🡪 …

=> User steuert Programm durch Events

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

?

3. GUI: 1. Zeichnen Sie das Fenster mit Inhalt, das durch den JavaFX Code beschrieben wird. 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();

}

}



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



<div>

<h1>DocScape – publizieren mit Zukunft</h1>

<img>

</div>

<div>

<h2>+++ einfach in der Anwendung</h2>

<h6> … </h6>

<p> … </p>

<p> … </p>

…

</div>



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>

-die Testklasse muss von der Javaklasse „ApplicationTest“ extenden

-man muss eine start(Stage) Methode implementieren

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>

-müssen Aktionen im Background ausgeführt werden, verwendet man dafür nicht den Main GUI Thread, sondern einen oder mehrere Background Threads

-irgendwann müssen die Background Threads wieder mit dem Main GUI Thread kommunizieren (Ergebnis/ Fortschritt/ Fehler zurückliefern), dafür kann man JavaFX API’s verwenden, die sicherstellen, dass die Interaktion zwischen Background Threads und UI auf dem korrekten Thread stattfindet

-die Task Class ist eine Klasse, von der Klassen erben, die Tasks implementieren, d.h. die Aktionen in Background Threads implementieren