ENTWICKLUNGSPORTFOLIO TEILKOMPONENT CLIENT

MODUL 326 OBJEKTORIENTIERT ENTWERFEN UND IMPLEMENTIEREN

LUKAS NYDEGGER INF13F
VERSION 1.0
16.11.2016

ENTWURFPRINZIP 1

PROBLEMSTELLUNG

Wir mussten uns für ein Architektur Muster unserer JavaFX Applikation entscheiden.

AUSWAHL DER PASSENDEN ENTWURFSPRINZIPS / -MUSTERS.

Da wir den Client entwickeln haben wir uns für die JavaFX Variante entschieden. Für beide ist es das erste Mal, dass wir mit JavaFX arbeiten. Deshalb haben wir uns eingelesen um einen möglichst guten Aufbau unserer Applikation sicher zu stellen.

Als Muster haben wir uns aus folgenden Gründen für die MVC Architektur entschieden:

- JavaFX sieht vor, dass es in verschiedene Views unterteilt wird, zwischen denen man wechseln kann.
- Für jede View kann auf einen Controller verwiesen werden um Aktionlistener zu implementieren und Benutzereingaben zu verarbeiten.

Diese wichtigen Eigenschaften von JavaFX lassen sich wunderbar mit dem MVC Muster umsetzen.

ANWENDUNG AUF DAS URSPRÜNGLICHE PROBLEM

Durch die angewandte MVC Architektur lässt sich unsere Applikation in verschiedene View mit entsprechenden Controller umsetzten. Das vereinfacht die Umsetzung des ganzen Clients erheblich, denn alle Teile lassen sich unabhängig voneinander entwickeln. Jede Klasse hat somit seine eigene Aufgabe.

LERNPROZESS

Das Wählen einer geeigneten Architektur seiner Applikation, sollte immer als erstes geschehen. Die Umsetzung wird erheblich einfacher. Anpassungen wie auch Wartungen solcher MVC Architekturen sind einfacher, da sich die Klassen unabhängig voneinander austauschen lassen.

Für uns als Team ist es von Nutzen wenn alle nach demselben Muster programmieren. Der Austausch und die Lesbarkeit des Codes ist einfacher.

ENTWURFPRINZIP 2

PROBLEMSTELLUNG

Bei der nicht Anwendung, des Single Responsibility Prinzips kann es sein, dass in unserer Applikation eine Redundanz der Funktionalität zweier Klassen entstehen kann. Dadurch wird die Lesbarkeit nicht gewährleistet.

AUSWAHL DER PASSENDEN ENTWURFSPRINZIPS / -MUSTERS.

Das Passende Entwurfsprinzip ist das Single Responsibility Prinzip, es sagt vor, dass eine Klasse nur einen Zeck hat. Die Funktionen in der Klasse sollte auch nur diesem Zweck dienen. So kann keine Funktionalität einer Klasse Doppelt oder auf mehrere verteilt werden.

ANWENDUNG AUF DAS URSPRÜNGLICHE PROBLEM

Bei unserer Applikation haben wir das Single Responsibility Prinzip angewendet und beim Erstellen einer Klasse genau überlegt, welchem Zweck es dienen soll und was für Funktionen sie haben soll.
Falls der Zweck schon zu einer bestehenden Klasse gehört hat, haben wir die Funktionen in dieser Implementier oder die Klasse gegebenenfalls unterteilt.

LERNPROZESS

Das Angewendete Entwurfsprinzip finde ich Persönlich sehr sinnvoll.
Sie lässt sich leicht umsetzten und dadurch ist die Motivation seine Klassen dementsprechend zu gestalten und allenfalls die Architektur nochmal zu überdenken gross.

Was mir am meisten aufgefallen ist, ist dass die Lesbarkeit des Codes meines Partners viel einfach wurde. Man weiss, was die Klasse macht, meist schon an dem Namen und deshalb weiss man auch wo man ansetzten muss.

Bei dem Entwurfsprinzip finde ich es wichtig, dass man seine Klassen demnach passend benennt. Dabei ist es auch wichtig die gleichen Begriffe zu benutzen und sich nicht abzuwechseln.