MVC

MVP

Aufgabe 1)

Entscheiden Sie ob folgenden Aussagen wahr oder falsch sind.

|  |  |  |
| --- | --- | --- |
| Beschreibung | richtig | falsch |
| Model-View-Presenter ist eine Weiterentwicklung von MVC. |  |  |
| MVP ist nicht für moderne Programmiersprachen ausgelegt. |  |  |
| Bei der Festlegung eines Designs nach dem MVP-Musters sollte zuerst der Presenter implementiert werden. |  |  |
| Es ist nicht sinnvoll für den Presenter eine Schnittstelle (Interface) zu spezifizieren. |  |  |
| Das Model enthält die Geschäftslogik. |  |  |
| Eine View in einer MVP-Anwendung kennt alle anderen Komponenten, wie z. B. das Model. |  |  |
| Die MVP-Anwendung ist nach der Implementierung nicht mehr erweiterbar. |  |  |
| Bei jedem Anforderungsprofil für eine Software ist das MVP-Pattern gegeignet. |  |  |
| Eine passive View verwaltet selbstständig Ihre GUI-Handler (bezüglich der Anwendungslogik). |  |  |
| Ein Presenter verwendet verwaltet konkrete Objekte der einzelnen Models und Views. |  |  |
| Ein Model enthält die Daten. |  |  |
| Ein Presenter ist eine Klasse, die als Bindeglied zwischen Benutzeroberfläche und der Datenhaltung verstanden werden kann |  |  |
| Das MVP-Pattern ist nur eine mögliche Lösung zur Implementierung einer komponentenbasierenden Softwarelösung |  |  |
| Die Realisierung einer Server-/Clientanwendung ist mit MVP nicht möglich |  |  |
| Die Implementierung einer MultiThreading-Umgebung ist in MVP nicht möglich |  |  |

Aufgabe 2)

Es soll ein virtueller Geldbeutel implementiert werden. In unserem Geldbeutel befinden sich zwischen 0 und 100 Euro. Der Geldbestand soll über verschiedene Benutzeroberflächen verändert werden können, die alle auf den selben Geldbeutel zugreifen. Unsere Anwendung soll zusätzlich um weitere Geldbeutel bzw. Benutzeroberflächen erweiterbar sein.



Zeichnen Sie ein Sequenzdiagramm, das den prinzipielle Zusammenspiel der Komponenten untereinander durch eine Veränderung des Geldbestandes durch den Benutzer zeigt.

**Hinweis:**

MVVM