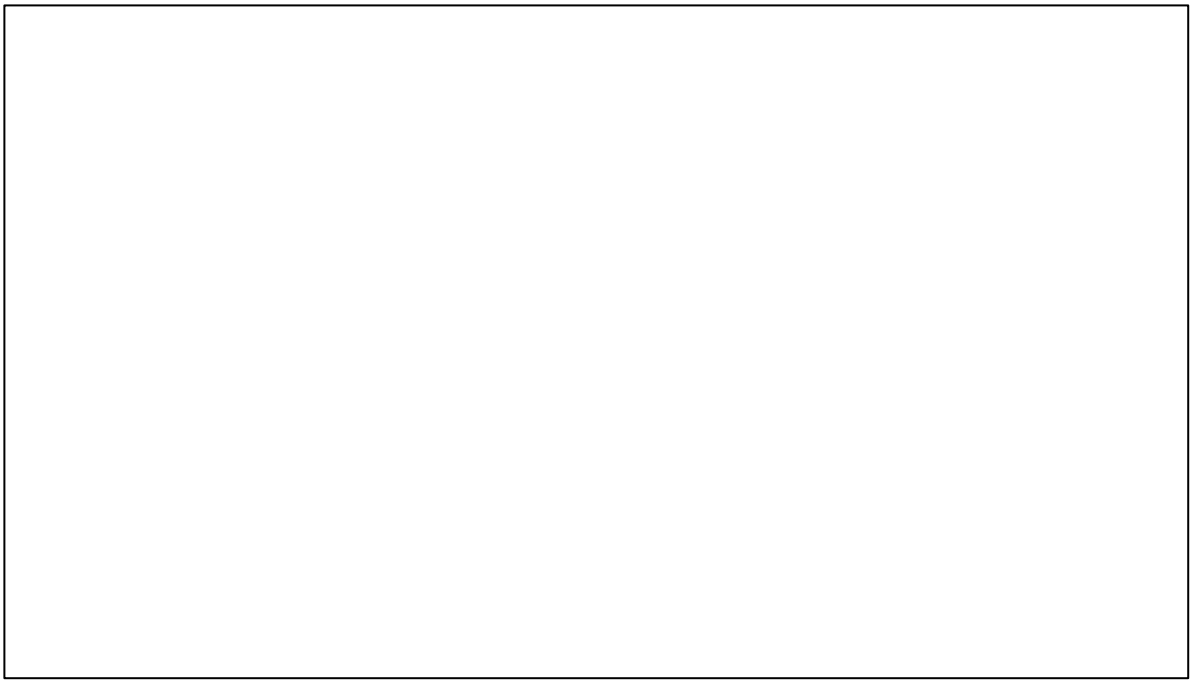
Observer Pattern

Aufgabe 1)

Klassendiagramm FAZ-Verlag + Observer Pattern



MVC: Model View Controller

Aufgabe 1)

Beschreibung	**richtig**	**falsch**
Die View verwaltet die Geschäftslogik.		
Bei der Pull-Methode holt sich die View vom Model per Getter die aktuellen Daten.		
Wenn eine Änderung am Model durchgeführt werden, schickt der Controller Daten an die View.		
Controller sind das Bindeglied zwischen View und Model.		
Das Model beinhaltet die Logik der Daten und deren interne Strukturen .		
Die Push-Methode ist immer die bessere Variante wenn das Subjekt alle Details der Observer kennt.		
Es gibt eine eindeutige, allgemeingültige Definition von MVC.		

MVP: Model View Presenter

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.		
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 geeignet.		
Eine passive View verwaltet selbstständig Ihre GUI-Handler (bezüglich der Anwendungslogik).		
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 denselben Geldbeutel zugreifen. Unsere Anwendung soll zusätzlich um weitere Geldbeutel bzw. Benutzeroberflächen erweiterbar sein.

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