


Objektorientierte Programmiertechniken / SA 2 Prof. Dr. Ursula Oesing	Hochschule Bochum Bochum University of Applied Sciences 
Praktikum Nr. 3 : Singleton und Observer	

3.1 Vorstellung der Aufgaben der Praktika

In den Praktika zur Veranstaltung Objektorientierte Programmiertechniken / SA 2 führen Sie Schritte zur Erstellung eines Softwareprodukts unter Berücksichtigung von Aspekten der Objektmodellierung durch. Diese sind unter anderem deshalb notwendig, um die Wartbarkeit und Erweiterbarkeit eines umfangreichen Softwareprodukts zu gewährleisten.

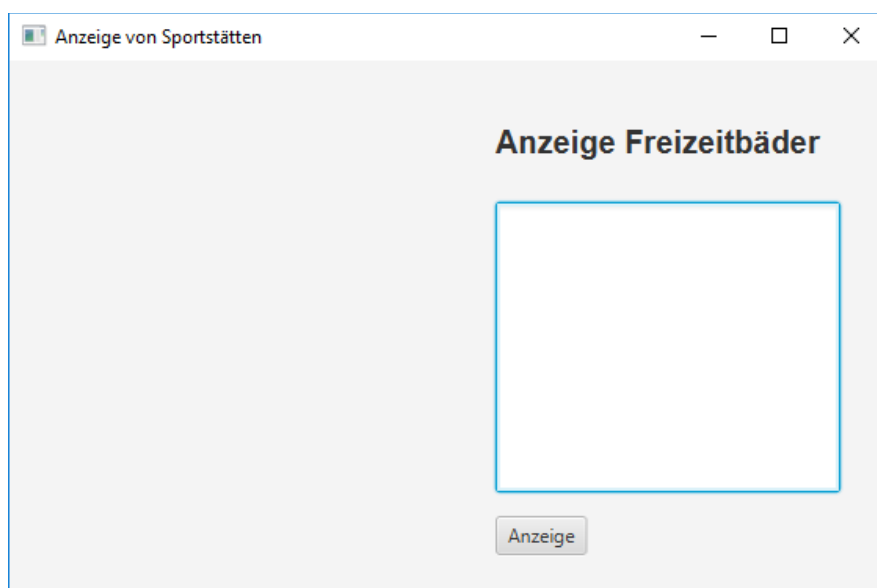
Heute behandeln Sie das objektbasierte Erzeugungsmuster *Singleton* und das objektbasierte Verhaltensmuster *Observer*.

Es steht eine Vorgabe zu *Singleton* zur Verfügung.

3.2 Aufgaben zur Vorbereitung unter Berücksichtigung von MVC

Grundlage ist das Ergebnis des letzten Praktikumstermins.

Ändern Sie die Anwendung aus dem zweiten Praktikumstermin ab. Legen Sie innerhalb des packages *gui* ein package *guiFreizeitbaeder* an und verschieben Sie die bisherigen Klassen des package *gui* mittels Drag an Drop nach *guiFreizeitbaeder*. Legen Sie ein package *guiSportstaetten* mit zwei Klassen *SportstaettenView* und *SportstaettenControl* innerhalb des packages *gui* an. Benutzen Sie für *SportstaettenView* die Vorgabe, die Sie noch ergänzen müssen.



Das Fenster zu Sportstaetten soll ebenfalls auf *FreizeitbaederModel* zugreifen.

Erzeugen Sie dann in der Klasse *Main* sowohl das Fenster zu Freizeitbädern, als auch zu Sportstätten.

```
@Override
public void start(Stage primaryStage) {
    // Fenster zu Freizeitbaedern
    new FreizeitbaederControl(primaryStage);
    // Fenster zu Sportstaettten
    Stage fensterSportstaetten = new Stage();
    new SportstaettenControl(fensterSportstaetten);
}
```

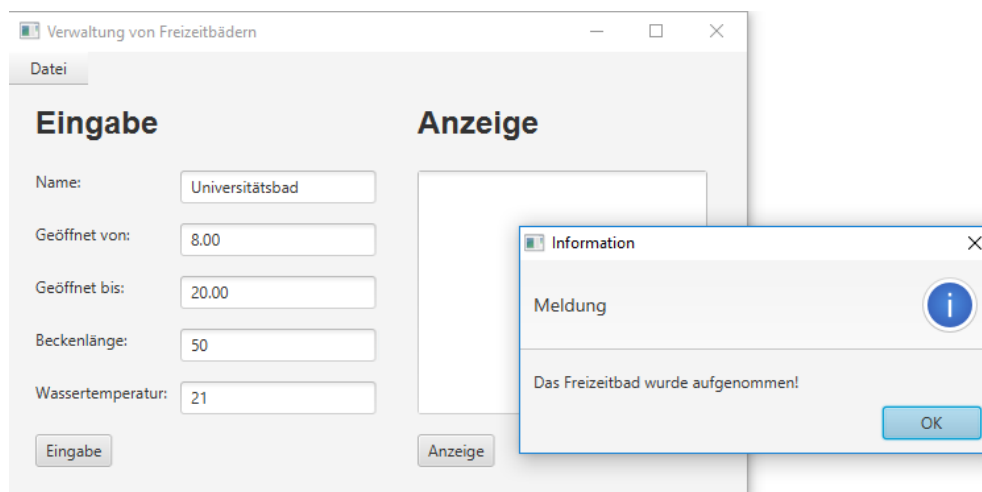
3.3 Realisierung von Singleton

Bisher werden für die beiden Fenster jeweils ein eigenes *FreizeitbaederModel* – Objekt erzeugt. Das erkennen Sie daran, dass ein Freizeitbad, welches in dem Fenster zu Freizeitbädern eingegeben wurde, in dem Fenster zu Sportstätten nicht angezeigt wird. Testen Sie das aus, indem Sie im Fenster zu Freizeitbädern ein neues Bad anlegen, dieses dort anzeigen lassen und dann im Fenster Sportstätten ebenfalls auf den Button *Anzeige* klicken!

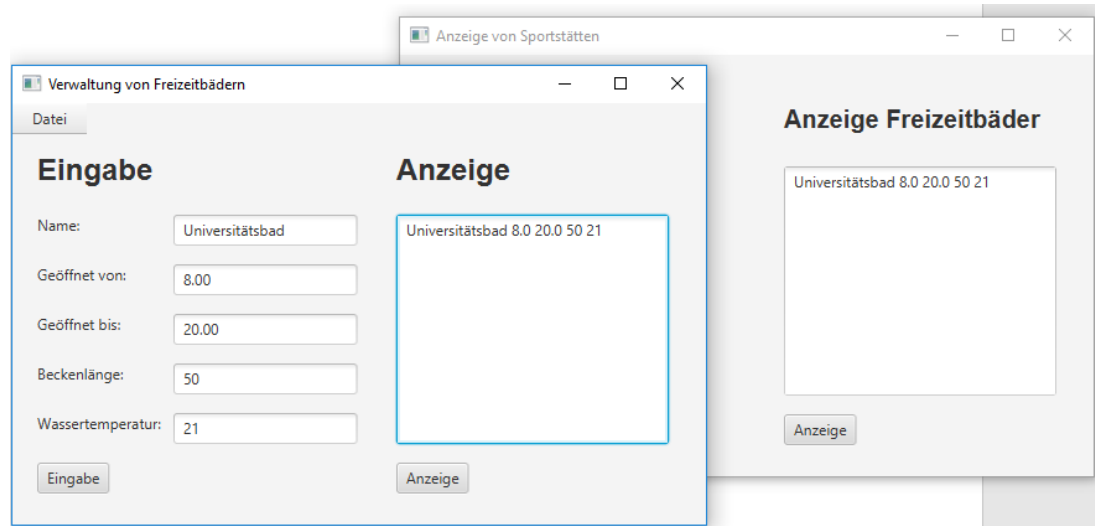
Ändern Sie die Anwendung ab. Die Klasse *FreizeitbaederModel* soll Singleton sein. Singleton gewährleistet hier, dass nur ein *FreizeitbaederModel*-Objekt erzeugt werden kann. Testen Sie am Anschluss, ob ein Freizeitbad, welches in dem Fenster zu Freizeitbädern eingegeben wurde, in dem Fenster zu Sportstätten angezeigt wird.

3.4 Realisierung von Observer mit eigenen Klassen

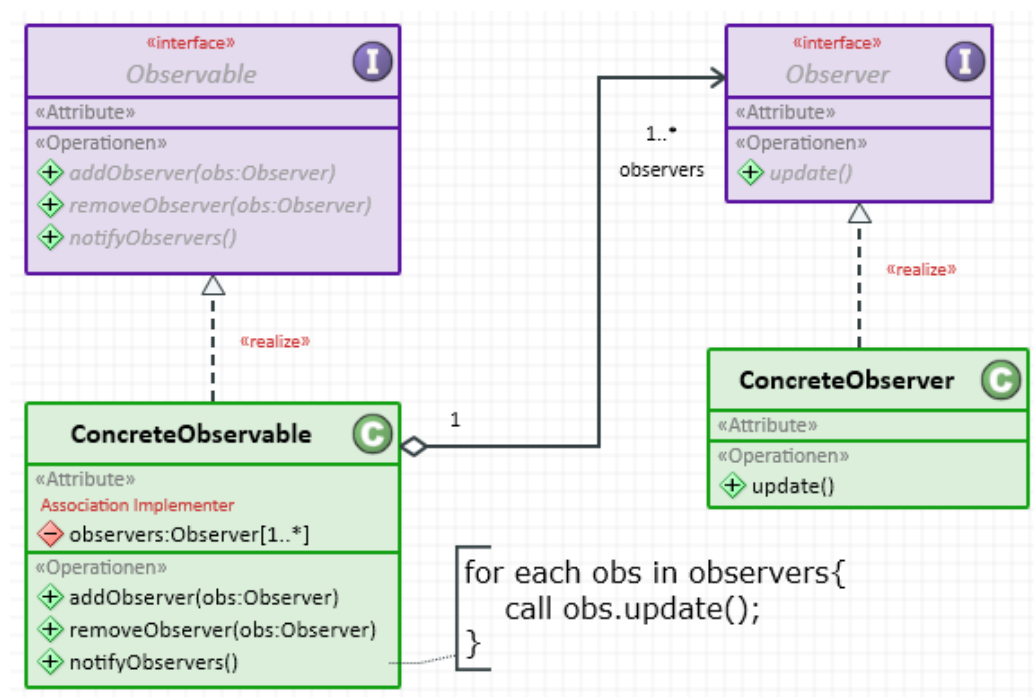
Ändern Sie die Anwendung aus 3.3 ab. Es soll bei der Eingabe eines Freizeitbades nicht wie bisher das Informationsfenster erscheinen,



sondern die Anzeige des Fensters zur Verwaltung von Freizeitbädern aktualisiert werden. Dazu soll das **Model** ein *ConcreteObservable* werden und das **Control** zu den Freizeitbädern ein *ConcreteObserver*. Die Anzeige von Freizeitbädern des Fensters zu den Sportstätten soll ebenfalls aktualisiert werden.



Erstellen Sie dazu eigene Interfaces *Observer* und *Observable* und realisieren Sie das Pattern, wie es im Buch *Patterns kompakt* vorgestellt wird.



Karl Eilebrecht, Gernot Starke, Patterns kompakt