Die Lösungen zu den einzelnen Aufgaben finden sich unter:

Aufgabe 1:

im Klassenmodell unter Main
Links ist das Fachkonzept: alle Klassen erben von Observable.
In der Mitte ist die Fassadenklasse.
Rechts in die GUI-Schicht: alle Klassen implementieren Observer.

Aufgrund der Übersichtlichkeit wurden die Observer-Design-Pattern – Assoziationen weggelassen.

Aufgabe 2:

Die ausführliche Dokumentation des Observer-Design-Patterns soll exemplarisch für alle Klassen der Anbindung der GUI-Schicht an die Fachkonzeptschicht dienen, da immer dasselbe Prinzip mit denselben Attributen vorherrscht.
Die Funktion des Observer-Patterns ist als Sequenzdiagramm mit Notiz unter *Logical View -> general_observer_pattern* dokumentiert.

Aufgabe 3:

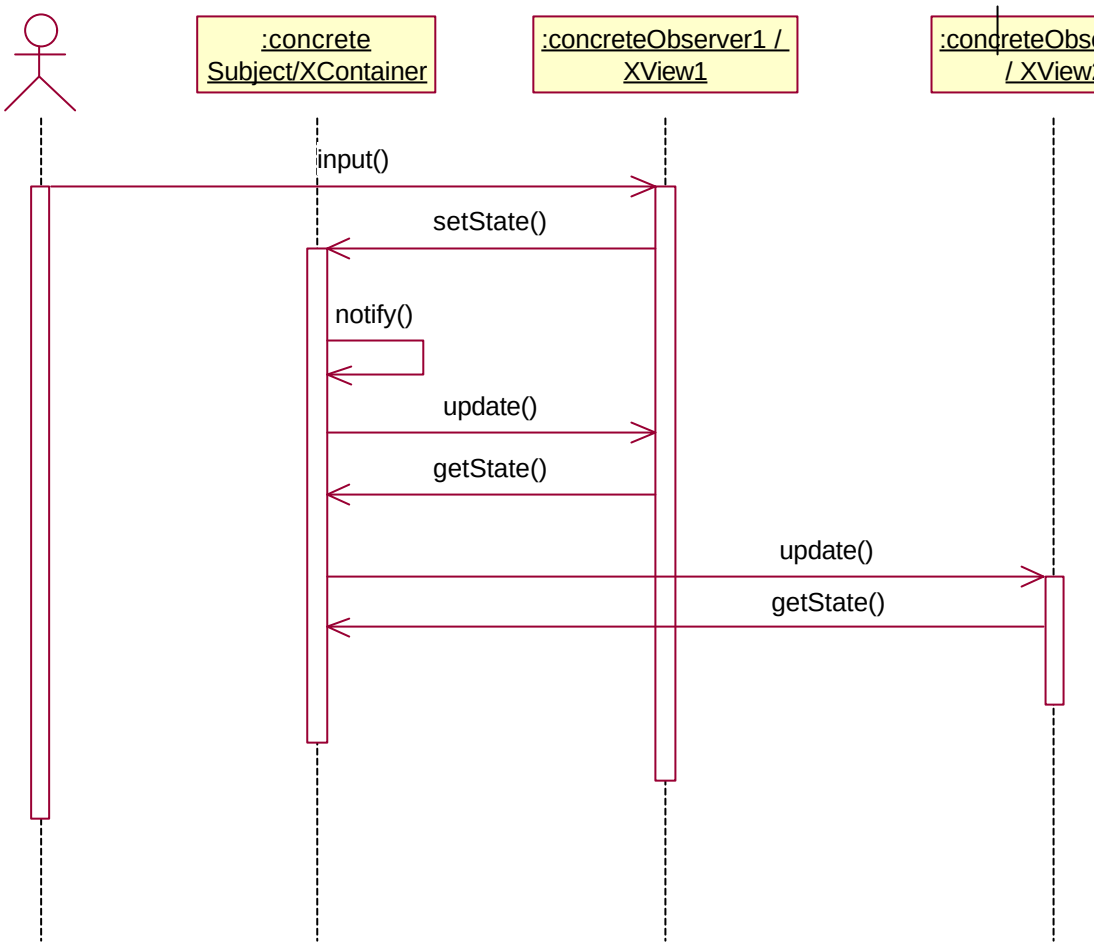
im Klassenmodell sind die Sequenzdiagramme der vorkommenden komplexen Operationen

Aufgabe 4:

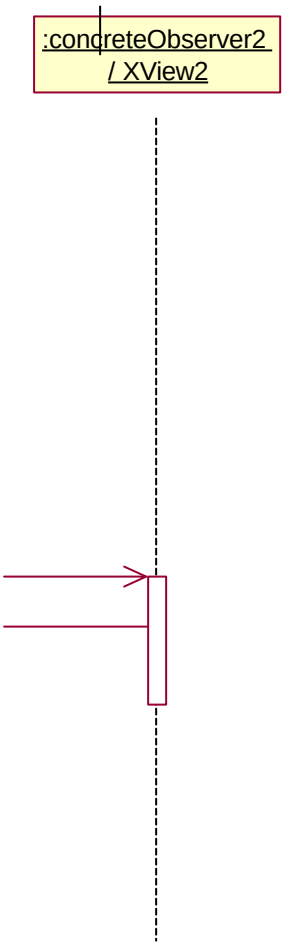
Diese Aufgabe wurde implizit bei der Erstellung des großen Rose-Modells erfüllt. Die Verfeinerung der Assoziationen / Attribute / Vererbungsstruktur wurde schon berücksichtigt, die Dokumentation der komplexen Funktionalitätsimplementierung ist in den Diagrammen von Aufgabe 3 enthalten.

Aufgabe 5:

a) Fassadenklasse ist im großen UML-Modell.
b) Alle GUI-Klassen wurden mittels Reverse-Engeneering in das Modell eingefügt.
c) Die Verwaltungsoperationen wurden in der Fachkonzeptschicht erstellt.



Werden Daten geändert (im View) wird über setState() die Änderung im Container vorgenommen. Über notify() wird die update() - methode in den observern aufgerufen, die dann mit getState() die Änderungen aus dem Container geholt und dann anzeigt.
Werden umgekehrt die Daten geändert, ohne dass ein Observer beteiligt ist, muss der Container (das concrete Subject) direkt notify() aufrufen. der rest ist gleich.


:concreteObserver2
/ XView2