SE2, Aufgabenblatt 7 (2 Termine)

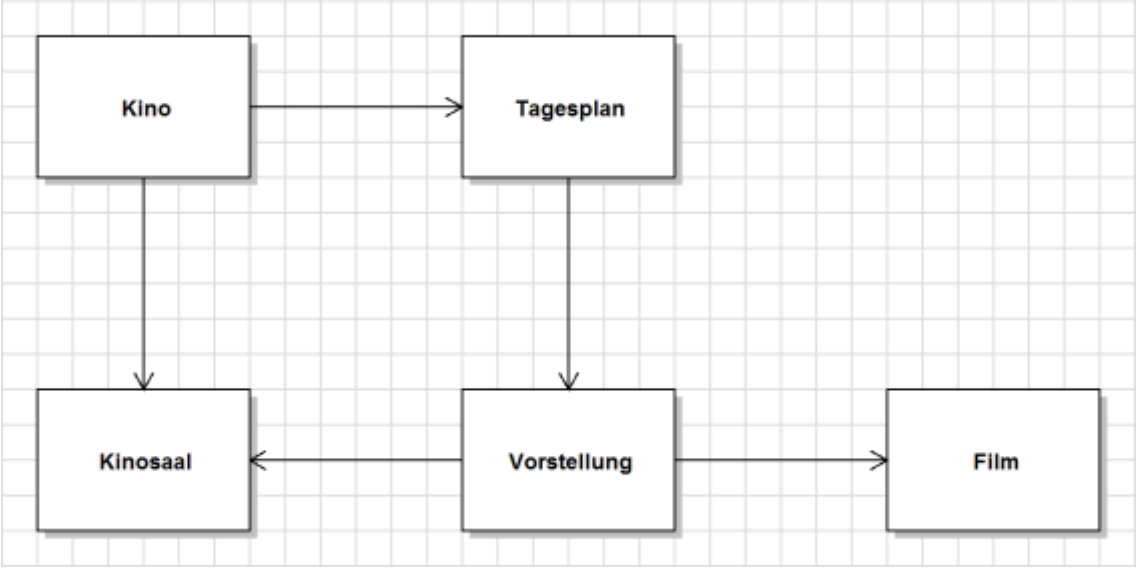
Modul: Softwareentwicklung II – Sommersemester 2017

Beobachtermuster

Moodle-Link ..... moodle.informatik.uni-hamburg.de
Projektraum ..... Softwareentwicklung 2 SoSe2017
Ausgabedatum ..... 26. Mai 2017

Kontext- und Subwerkzeuge mit dem Beobachtermuster verbinden

Auf diesem Aufgabenblatt verwendet Ihr nicht mehr die Mediathek, sondern ein neues Softwaresystem: das Kinoticketsystem. Mit dieser Anwendung sollen an der Kasse eines Kinos Karten für die Vorstellungen verkauft werden. Als Einstiegshilfe in das neue System dient ein Klassendiagramm der Materialien:



Auch das Kinoticketsystem folgt den SE2-Entwurfsregeln, die ihr euch nochmal genau ansehen solltet.

Wenn Ihr das Ausgangssystem startet, werdet Ihr feststellen, dass das System scheinbar bisher noch keine Funktionalität enthält. Das liegt daran, dass die Benutzungsoberfläche sich aus mehreren Subwerkzeugen zusammensetzt, die bisher nicht miteinander verbunden sind. Daher werden Änderungen, die ein Werkzeug bewirkt, nicht an der Oberfläche angezeigt. Eure Aufgabe auf diesem Aufgabenblatt ist es, das zu ändern.

(Termin 1) Aufgabe 7.1 Lösungsvorschlag erarbeiten (2 Stunden)

Das Beobachtermuster ermöglicht es einem Dienstleister, Zustandsänderungen an seine Klienten zu signalisieren, ohne dass er seine Klienten (statisch) kennen muss. Dies erhöht u.a. die Testbarkeit und Wiederverwendbarkeit des Dienstleisters, da er nicht an bestimmte Klienten gekoppelt ist.

In den SE2-Entwurfsregeln wird das Beobachtermuster u.a. dazu eingesetzt, die Subwerkzeuge von ihrem Kontextwerkzeug zu entkoppeln, damit sie nicht nur als Teil eines bestimmten Kontextwerkzeugs funktionieren, sondern (theoretisch) in beliebig vielen Kontextwerkzeugen eingesetzt werden können.

Erarbeitet in Eurer Gruppe einen Lösungsvorschlag, um die Werkzeuge in dem Kinoticketsystem mit Hilfe des Beobachtermusters zu verbinden. Als Ausgangssystem dient das Projekt Kinoticketverkauf_Vorlage_Blatt07. Die Werkzeuge inklusive ihrer Benutzungsoberfläche sind in dem Ausgangssystem bereits vorhanden.

Tipp: Um einen Überblick über ein Projekt zu bekommen, ist die Funktion „Show References“ von Eclipse hilfreich. Sie sucht alle Quelltextstellen, an denen ein Element (z.B. eine Klasse oder Methode) verwendet wird. Setzt dazu den Cursor auf den Klassen- oder Methodennamen und wählt dann Search → References → Workspace bzw. Strg+Shift+G.

Euer Lösungsvorschlag soll die folgenden Fragen beantworten:

- Welche Werkzeuge sind in dem System vorhanden? Stellt die Baumstruktur von Kontext- und Subwerkzeugen dar.
- Auf welche Zustandsänderungen in Werkzeugen müssen welche anderen Werkzeuge reagieren? Wie wird mit Hilfe des Beobachtermusters über diese Änderungen informiert? Stellt dar, auf welchem Weg die Information über eine Änderung weitergegeben wird.
- Stellt die Klassen und Interfaces eures Lösungsvorschlags dar und setzt diese in Beziehung zur allgemeinen Struktur des Beobachtermusters. Welche Klassen bzw. Interfaces eurer Lösung entsprechen welchem Element des Beobachtermusters?
- Ein Kontextwerkzeug kann mehrere Subwerkzeuge enthalten und beobachten. Wie unterscheidet ihr in einem Kontextwerkzeug, in welchem Subwerkzeug ein Ereignis aufgetreten ist? Woher kennt das Kontextwerkzeug den neuen Zustand des Subwerkzeugs?
- Stellt in einem Objektdiagramm den Zustand des geplanten Systems im Zeitpunkt nach der Ausführung der Konstruktoren dar. Verdeutlicht anhand dieses Diagramms die Abläufe im System.

Hinweis: Haltet euch dringend an die UML-Konventionen. Das Objektdiagramm und das Klassendiagramm wurden in SE1 besprochen (Vorlesung V06-Objektgeflechte). Die Folien dazu befinden sich im Moodle von SE1. Schaut euch noch einmal die Folien (Folie 29-40) dazu an, falls ihr nicht mehr wisst, welche Konventionen für diese Diagrammarten gelten (Beispiel: wie stelle ich eine öffentliche Methode im Klassendiagramm dar, wie stelle ich im Klassendiagramm eine Methode richtig dar (Rückgabetyp, Parameter etc.), wie unterscheide ich im Klassendiagramm zwischen Klasse, Interface und einer abstrakten Klasse, welche Pfeilarten benötige ich, wie modelliere ich den aktuellen Zustand eines Objekts mittels eines Objektdiagramms etc.).

(Termin 1) Aufgabe 7.2 Lösungsvorschlag präsentieren (40 Minuten)

Präsentiert euren Lösungsvorschlag im Übungsraum. Hierfür stehen einer ausgewählten Gruppe etwa 20 Minuten zur Verfügung. Die von euch erstellten Entwürfe müssen dabei sinnvoll aufbereitet sein; zum Beispiel können Klassendiagramme an die Tafel gemalt oder Bilder über den Beamer gezeigt werden.

Die Präsentation hat das Ziel, Feedback zu euren Ideen einzuholen. Es ist also wichtig, dass eure Zuhörer euren Entwurf nachvollziehen können. Außerdem sollten Zwischenergebnisse von Diskussionen durch euch explizit festgehalten werden; beispielsweise direkt an der Tafel durch das Verändern von Klassendiagrammen.

(Termin 1) Aufgabe 7.3 Lösungsvorschlag überarbeiten (20 Minuten)

Überarbeitet euren eigenen Entwurf auf Basis der Präsentationen aller Gruppen. Das Ergebnis dieser Aufgabe muss so ausführlich sein, dass ihr euch auch noch in der nächsten Woche daran orientieren könnt.

(Termin 2) Aufgabe 7.4 Lösung implementieren (1,5+ Stunden)

Implementiert euren Lösungsvorschlag soweit, dass die Werkzeuge des Kinoticketsystems sinnvoll miteinander zusammenhängen. Arbeitet hierfür sowohl außerhalb der betreuten Zeit, als auch während des 2. Termins. Haltet dabei die vorgegebenen softwaretechnischen Qualitätsmerkmale (bis auf Tests) ein:

- ☐ Vertragsmodell
- ☐ Schnittstellenkommentare
- ☐ Quelltextkonventionen

Zusatzaufgabe: Schreibt einen Testfall, der testet, ob die Methode informiereUeberAenderung auch wirklich alle angemeldeten Beobachter informiert. Dieser Testfall soll sich nicht auf die schon im System verbauten Werkzeuge beziehen; wir testen in SE2 keine Werkzeuge. Schreibt stattdessen eine neue, leere Unterklasse von Beobachtbar, sowie eine neue Beobachter-Klasse, die die Anzahl der Methodenaufrufe von reagiereAufAenderung mitzählt. Testet anschließend das korrekte Zusammenspiel dieser beiden neuen Klassen in einem Testfall.

Zusatzaufgabe: Schaut Euch die Klasse `java.util.Observable` und das zugehörige Interface `java.util.Observer` an und verwendet diese, um eine zweite Lösung für das Kinoticketsystem zu implementieren.

(Termin 2) Aufgabe 7.5 Lösung vorstellen/Code Review (1 Stunde)

Stellt eure Lösung in eurem Übungsraum vor. Hierfür stehen einer ausgewählten Gruppe etwa 20 Minuten zur Verfügung. Es eignen sich zuerst eine Präsentation des laufenden Programms und danach ein Blick auf den Quelltext am Beamer. Eventuell sollte auch euer anfänglicher Lösungsvorschlag wieder für alle Zuhörer sichtbar sein. Beantwortet dabei folgende Fragen:

- Welche Klassen habt ihr neu implementiert oder erweitert?
- Warum stehen Klassen in der von euch gewählten Beziehung zueinander?
- Konntet ihr euren Lösungsvorschlag 1 zu 1 umsetzen, oder gab es Änderungen aufgrund von unvorhergesehenen Abhängigkeiten?

Es ist das Ziel dieser Aufgabe, die Qualität des von euch geschriebenen Quelltextes zu verbessern.