SE2, Aufgabenblatt 6 (2 Termine)

Modul: Softwareentwicklung II – Sommersemester 2020

Beobachtermuster

Ausgabedatum ..... 27. Mai 2020

Checkliste Abgabe

Woche 1 (27.05.-10.06.):

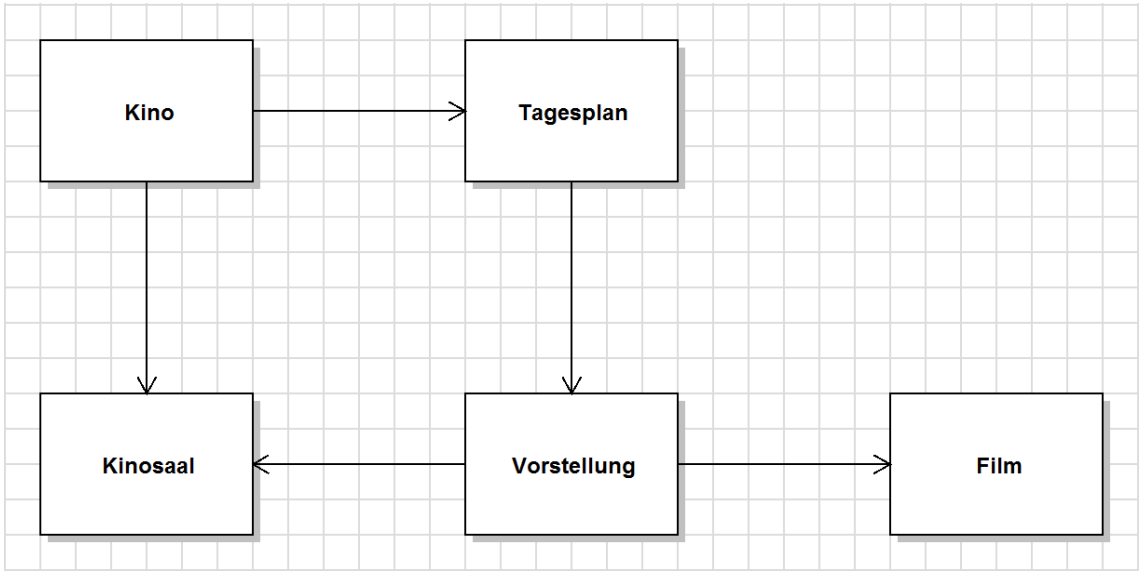
- ☐ **Einmal:** UML und Objektdiagramm im Moodle hochladen
- ☐ **Beide in der Gruppe:** Test zum Lösungsvorschlag Blatt 6 und zu den SE2-Entwurfsregeln

Woche 2 (10.06.-17.06.):

- ☐ **Einmal:** Kinoticketsystem im Moodle hochladen. Geht anhand der Checkliste am Ende des Zettels vor.

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.

(Woche 1) Aufgabe 6.1 Lösungsvorschlag erarbeiten

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. **Beantwortet im** Moodle Fragen zu den Entwurfsregeln.

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_Blatt06. 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.

Der **Lösungsvorschlag** ist am **10.06.** fällig! Die **Implementierung** erst in der Woche darauf, am **17.06.**!

Beantwortet zuerst im Moodle unter anderem folgende Fragen zu eurem Lösungsvorschlag:

- 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?

Außerdem sollt ihr per UML und Objektdiagramm die untenstehenden Aufgaben erfüllen. Arbeitet hier auch gerne mit Notizfeldern in den Diagrammen, um Implementierungsdetails aufzuschreiben.

- Stellt die Klassen und Interfaces eures Lösungsvorschlags in einem **Klassendiagramm** dar und setzt diese in Beziehung zur allgemeinen Struktur des Beobachtermusters. Stellt dar, auf welchem Weg die Information über eine Änderung weitergegeben wird. **Beantwortet schriftlich** welche Klassen bzw. Interfaces eurer Lösung welchem Element des Beobachtermusters entsprechen.
- Stellt in einem **Objektdiagramm** den Zustand des geplanten Systems im Zeitpunkt nach der Ausführung der Konstruktoren dar (also zur Laufzeit). **Verdeutlicht** anhand dieses Diagramms **schriftlich** die Abläufe im System.
- 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? Beantwortet die Fragen **schriftlich**.

Hinweis: Haltet euch dringend an die UML-Konventionen. Das Objektdiagramm und das Klassendiagramm wurden in SE1 besprochen (Vorlesung V06- UML_Syntaktische_Strukturen). Die Folien dazu befinden sich im Moodle. Schaut euch noch einmal die Folien (Folie 4-14) 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.).

(Woche 2) Aufgabe 6.2 Lösung implementieren

Die Implementierung ist erst **am 17.06.** fällig! Implementiert euren Lösungsvorschlag soweit, dass die Werkzeuge des Kinoticketsystems sinnvoll miteinander zusammenhängen. Haltet dabei die vorgegebenen softwaretechnischen Qualitätsmerkmale (bis auf Tests) ein:

- ☐ Vertragsmodell
- ☐ Schnittstellenkommentare
- ☐ Quelltextkonventionen

(Woche 2) Aufgabe 6.3 Lösung ergänzen

Beantwortet zusätzlich zu eurer Implementierung folgende Fragen in einer separaten Datei:

- 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?

Das Java-Projekt sollte wieder folgende Punkte erfüllen:

- ☐ Es wurden alle Aufgaben bearbeitet
- ☐ Es sind keine Fehler mehr vorhanden, d.h. keine Klasse im Packages Explorer hat einen roten Kreis mit weißem Kreuz am Symbol
- ☐ Das Projekt lässt sich kompilieren, d.h. StartUpMediathek_Blatt_04_05 lässt sich ohne Fehlermeldung starten
- ☐ Alle Testfälle laufen ohne Fehler durch. Ihr könnt alle Tests ausführen, indem ihr im Package Explorer Rechtsklick auf den Package-Namen macht und *Run as>JUnit Test* auswählt und alle Tests grün angezeigt werden
- ☐ Projekt und Datei sind umbenannt und mit eurem Namen gekennzeichnet