Musterlösung zur Probeklausur „Softwaretechnologie“ 2010/2011

Dr. Günter Kniesel

Institut für Informatik III
Universität Bonn

– 23. Dezember 2010 –

Grundsätzliches

Die Probeklausur ist ein Auszug aus einer tatsächlich in dieser Form stattgefundenen Klausur. Der einzige Unterschied besteht darin, dass Aufgaben aus den in der laufenden Vorlesung noch nicht besprochenen Themenbereichen weggelassen worden sind. Dementsprechend führen wir die Probeklausur in 90 statt 120 Minuten durch.

Der Sinn der Probeklausur ist es, Ihnen eine Vorstellung von dem Ablauf der Klausur sowie der Art der Aufgaben zu vermitteln und Ihnen zu ermöglichen Ihren aktuellen Leistungsstand selbst einzuschätzen.

Versuchen Sie nicht anhand der Probeklausur „Kafffeesatzlesen“ zu betreiben! Sowohl der Schluss, dass etwas in der Klausur nicht vorkommen wird, weil es in der Probeklausur schon dran war, als auch der Umkehrschluss könnte sich als gefährlicher Trugschluss erweisen.

Das Einzige, wovon Sie ausgehen können ist, dass die Klausur

- mehr Aufgaben anbieten wird, als für das Erreichen der maximalen Punktzahl erforderlich wäre, so dass Sie die Möglichkeit haben, mindestens eine Aufgabe wegzulassen und trotzdem die Note 1 zu erreichen
- und in ungefähr die gleiche *Art* von Aufgaben (aber nicht unbedingt die gleichen Inhalte) enthalten wird.

Musterlösung und Benotungshinweise

Die nachfolgende Musterlösung zu Probeklausur soll vor allem eines: Ihnen helfen, Ihre eigenen Ergebnisse inhaltlich zu überprüfen! Wenn Sie Abweichungen zwischen Ihrer Abgabe und der Musterlösung feststellen, schlagen Sie im Skript nach und diskutieren Sie mit Kollegen.

Die beigefügten Hinweise, wie Punkte vergeben werden würden, gelten nur für eine annähernd zur Musterlösung ähnliche Abgabe! Insbesondere bei Modellierungsaufgaben, ist das „Erbsen zählen“ nur eine Hilfe, um *grundsätzlich richtige* Lösungen miteinander vergleichen zu können und eine faire Notenabstufung zu finden. Wenn hingegen die abgegebene Lösung die Aufgabe sinngemäß nicht löst, ist es egal wie viele Assoziationen, Klassen und sonstige Elemente Sie eingezeichnet haben. Voll daneben bleibt voll daneben.

Nachklausur zur Vorlesung „Softwaretechnologie“ 2009/2010

Dr. Günter Kniesel

Institut für Informatik III
Universität Bonn

– 26. März 2010 –

Aufgabe		Maximale Punkte	Erreichte Punkte	Prüfer / Beisitzer
1	Verschiedenes	8,0		
2	Klassen	15,5		
3	Aktivitäts-Diagramm	6,0		
4	Use Case	14,5		
5	Analyse	11,5		
6	Entwurfsmuster	9,0		
7	Entwurfsmuster	11,0		
8	CRC + DBC	9,0		
9	… gestrichen …	5,0		
10	… gestrichen ...	12,5		
Summe		**102,0**		
Note				

Wertung

Für die Note 1.0 reichen 76,5 von 102 Punkten. Diese Regelung ermöglicht Ihnen, Teilaufgaben im Umfang von ca. ¼ der Punkte wegzulassen und trotzdem eine 1.0 zu erzielen!

Setzen Sie also Prioritäten und teilen Sie sich die Zeit richtig ein! *Alle Aufgaben in der verfügbaren Zeit komplett zu bearbeiten dürfte schwierig sein.*

Zuordnung Punkte → Note			
.....– 76,5 → 1,0	68 – 64,5 → 2,0	56 – 52,5 → 3,0	44 – 40,5 → 4,0
76 – 72,5 → 1,3	64 – 60,5 → 2,3	52 – 48,5 → 3,3	40 – 0 → 5,0
72 – 68,5 → 1,7	60 – 56,5 → 2,7	48 – 44,5 → 3,7	

Weitere Hinweise

Folgende Aufgaben sind selbstständig, ohne Verwendung von Hilfsmitteln zu lösen.

Dinge die nicht in die Wertung einfließen sollen *(z.B. Randüberlegungen, als falsch erkannte Antworten, verworfene Versuche) bitte durchstreichen oder sonstwie deutlich markieren.*

Tragen sie Ihre Matrikelnummer oben rechts auf jedem Blatt ein. *Antworten, die auf nicht gekennzeichneten Blättern erfolgen, werden nicht gewertet. Sie dürfen auch die Rückseiten der Blätter mitbenutzen.* ***Wenn Sie für Ihre Antworten zusätzliche Blätter brauchen, erhalten Sie diese von den Betreuern.***

Bei Unklarheiten hinsichtlich der Aufgabenstellung*, melden Sie sich durch Handzeichen. Ein Betreuer kommt dann zu Ihrem Platz.*

Nicht mit Bleistift *sondern mit einem dokumentenechten Stift schreiben!*

Aufgabe 1. *Verschiedenes (8 Punkte)*

Bewerten Sie die Korrektheit der Aussagen mit **Ja** oder **Nein**. Es sind eventuell mehrere Aussagen richtig oder keine Einzige. Bewertungsschema:

- Eine *richtig* angekreuzte Aussage zählt +0,5 Punkte.
- Eine *nicht* angekreuzte Aussage zählt 0 Punkte.
- Eine *falsch* angekreuzte Aussage zählt –0,5 Punkte.

Man kann in der *gesamten* Aufgabe nicht weniger als 0 Punkte erreichen.

a) (2 Punkte)

Eine Referenzversion („Tag“) …		
… darf nur einmal pro Tag erstellt werden.	☐ Ja	☑ Nein
… sollte fehlerfrei sein.	☑ Ja	☐ Nein
… kann in SVN durch den Kopierbefehl erstellt werden.	☑ Ja	☐ Nein
… sollte aufgrund des hohen Speicherplatzbedarfs nicht allzu häufig erstellt werden.	☐ Ja	☑ Nein

b) (2 Punkte)

Refactoring …		
… ist die konsequente Änderung von Code wann immer es erforderlich ist	☐ Ja	☑ Nein
… setzt voraus, dass es automatisierte Tests der veränderten Funktionalität gibt	☑ Ja	☐ Nein
… sichert Verhaltenserhaltung durch Code Reviews	☐ Ja	☑ Nein
… setzt voraus, dass „Bad Smells“ automatisch erkannt werden können	☐ Ja	☑ Nein

c) (2 Punkte)

Zu den essentiellen Eigenschaften von Software-Komponenten gehört, dass sie …		
… auch „benutzte“ Interfaces spezifizieren	☑ Ja	☐ Nein
… alle Abhängigkeiten zu anderen Komponenten explizit machen	☑ Ja	☐ Nein
… Vererbung nutzen können	☐ Ja	☑ Nein
… objekt-orientiert implementiert sein müssen.	☐ Ja	☑ Nein

d) (2 Punkte)

Zu den essentiellen Eigenschaften von „agilen“ Softwareprozessen gehört, dass sie …		
… Kommunikation zwischen Entwicklern und Anwendern fördern	☑ Ja	☐ Nein
… extrem konsequent saubere / aktuelle Dokumentation erstellen	☐ Ja	☑ Nein
… extrem viel Refactoring und automatisierte Tests einsetzen	☐ Ja	☑ Nein
… im Entwurf zukünftige Anforderungsänderungen antizipieren und durch entsprechende Design Patterns vorbereiten	☐ Ja	☑ Nein

Kommentar [g1]: Diese Aufgabe hätte nicht enthalten sein sollen, denn das Thema „Agile Softwareentwicklung“ wird erst im Januar besprochen.

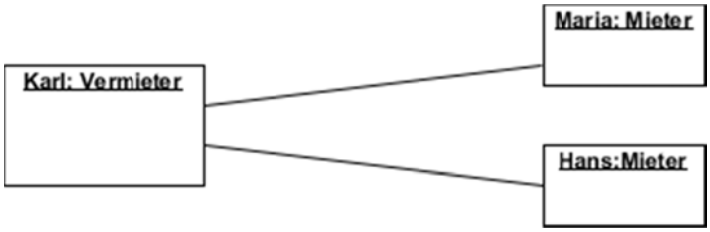
Aufgabe 2. *Klassen- und Objektdiagramme (15,5 Punkte)*

Zeichnen Sie Klassendiagramme, die folgende Sachverhalte modellieren. Notieren Sie auch die Multiplizitäten.

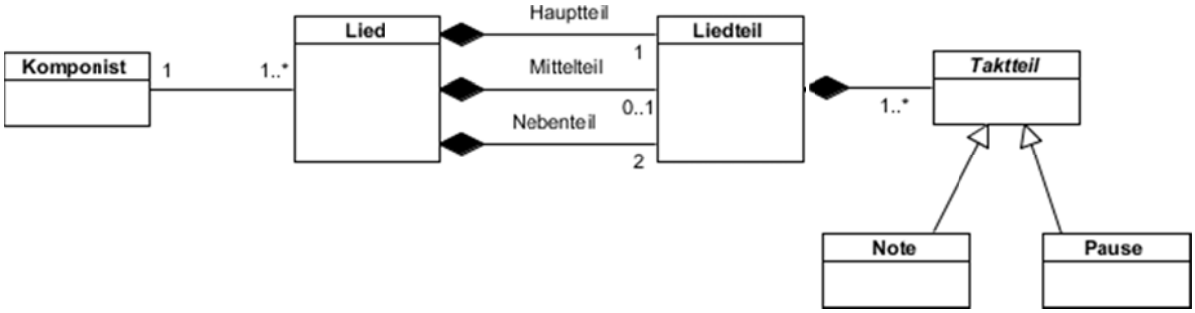
a) (**3 Punkte** – *je 0,5: Klassen, Multiplizitäten, Assoziation und Name*) Zwischen einem Vermieter und einem Mieter besteht in der Regel eine Verbindung, die als „Vertrag“ bezeichnet wird. Mieter können mit einem oder auch mehreren Vermietern einen Vertrag haben, Vermieter können Mieter haben, müssen dies aber nicht.

← Fehler: Kardinalitäten müssen umgedreht werden (Vermieter 1..*, Mieter 0..*)

b) (**2,5 Punkte** – *je 0,5: Objekte, Links – je 0,5 Abzug bei konsistent falscher Objektnotation [fehlender Name oder fehlender Typ]*) Zeichnen Sie passend zu Ihrem Klassendiagram das Objektdiagramm welches beschreibt, dass Karl in seinem einzigen Haus je eine Wohnung an Maria und an Hans vermietet hat.



c) (**10 Punkte** – *je 0,5: Klassen, Multiplizitäten die nicht weggelassen werden dürfen, Assoziationen und Namen. Bei Assoziationen der falschen Art nur halb soviel.*) Ein Lied hat einen Komponisten und besteht aus zwei Nebenteilen und einem Hauptteil. Optional kann ein Mittelteil vorhanden sein. Alle Liedteile bestehen aus Taktteilen, die Noten oder Pausen sein können. Liedteile können nicht leer sein.



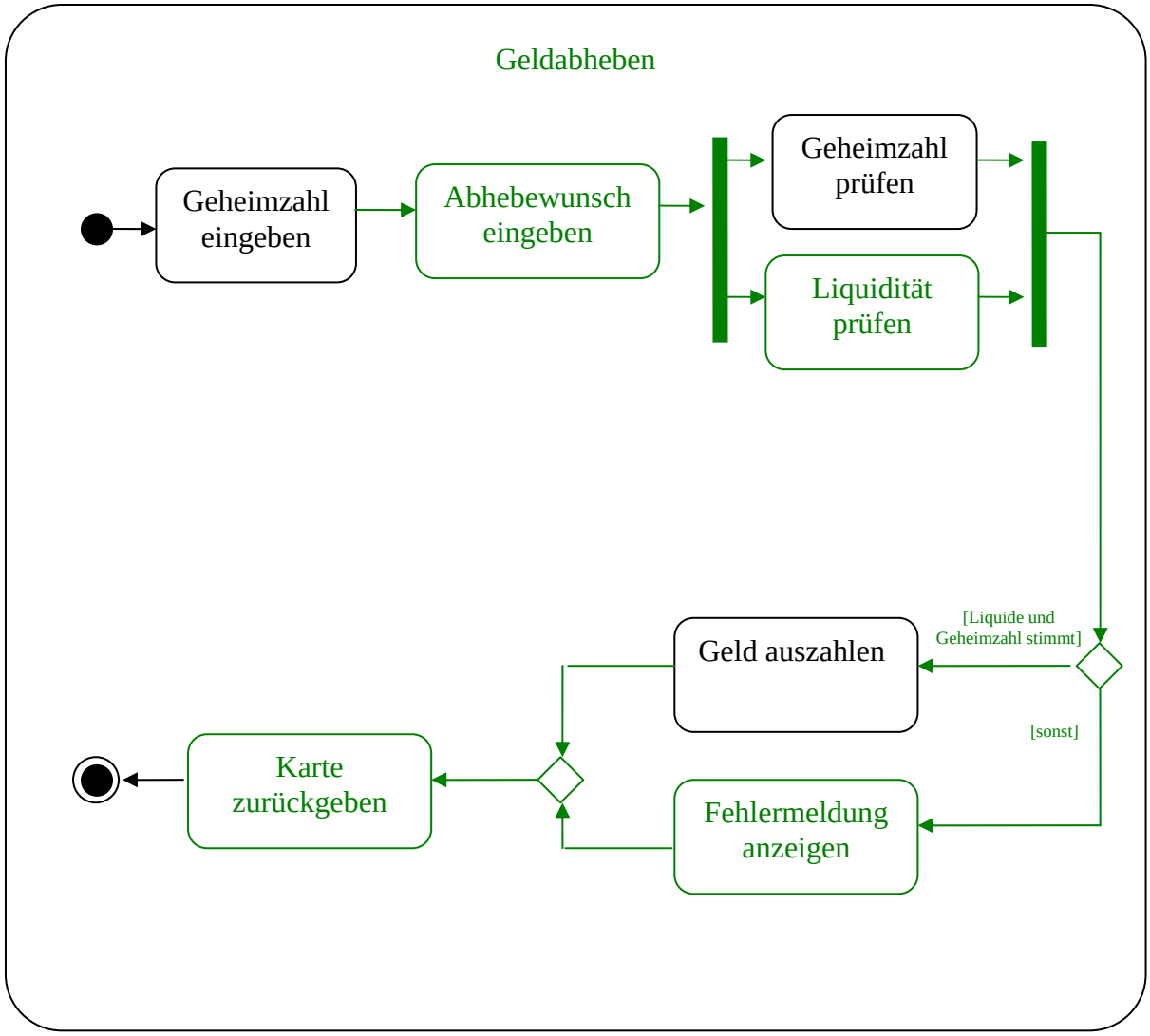
Erläuterungen: *(i) Ein Komponist ist jemand der mindestens ein Lied komponiert hat (daher die Kardinalität 1..* in der Beziehung zum Lied). Wenn man schon ein Komponist wäre weil man sich dafür hält, wäre die Kardinalität * akzeptabel. (ii) Wenn man der Meinung ist, dass Noten und Pausen auch gemalt werden können, ohne dass sie Teil eines Liedes sind, könnte man am Kompositions-Ende (schwarze Raute) der Beziehung zwischen Liedteil und Taktteil die Kardinalität 0,1 spezifizieren.*

Aufgabe 3. *Aktivitätsdiagramme (6 Punkte)*

Vereinfacht stellt sich das Geldabheben an einem Automaten wie folgt dar:

- Der Kunde gibt dem Geldautomaten seine Geheimzahl ein.
- Der Kunde gibt an, wie viel Geld er abheben möchte.
- Der Geldautomat überprüft die Geheimzahl.
- Gleichzeitig überprüft der Automat die Liquidität des Kunden bei dessen Bank.
- Wenn der Kunde liquide ist und die korrekte Geheimzahl eingegeben hat, erhält er nacheinander das Bargeld und die Karte vom Automaten.
- Andernfalls erhält der Kunde eine Fehlermeldung angezeigt und danach die Karte zurück.

(**6 Punkte**– *je 1: 4 korrekte Aktivitäten, 1 Synchronisationsklammer, 1 Entscheidungsklammer, -- 1 Punkt Abzug wenn Pfeile nicht gerichtet – Pfeilköpfe nicht wichtig.*) Vervollständigen Sie untenstehendes Aktivitätsdiagramm zu diesem Prozess.

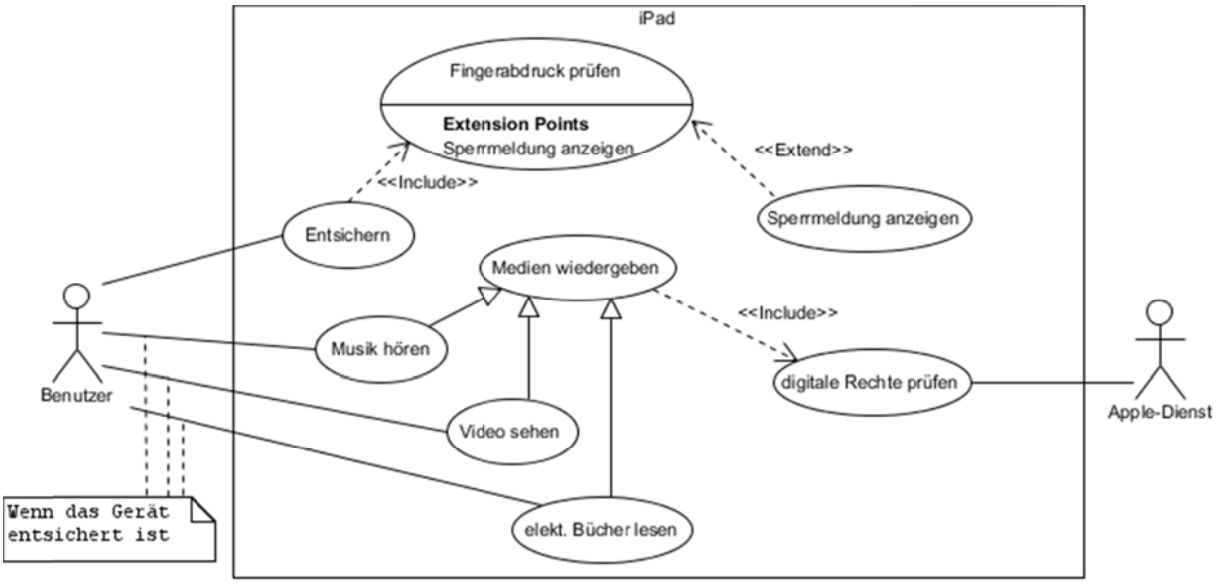


Aufgabe 4. *Anwendungsfälle (14,5 Punkte)*

Sie sollen für den neu vorgestellten tragbaren Computer „iPad“ die Betriebsystem-Software weiterentwickeln. Dazu erhalten Sie folgende Funktionsbeschreibung:

> Der Benutzer entsichert seinen iPad, indem er mit dem Finger darüber streicht. Dabei wird sein Fingerabdruck analysiert. Wenn dabei festgestellt wird, dass der Fingerabdruck nicht dem Besitzer gehört, wird eine Sperrmeldung angezeigt.
>
> Nachdem der iPad entsichert wurde, kann der Benutzer Musik hören, Videos sehen oder elektronische Bücher lesen. In jedem Fall werden vor der Wiedergabe eines Mediums mit Hilfe eines externen Apple-Dienstes die digitalen Rechte geprüft.

a) (**9,5 Punkte** – *je 0,5: Rahmen,*) Erstellen Sie für obige Notizen ein Use Case-Diagramm, das die relevanten Anwendungsfälle darstellt und untereinander in Beziehung setzt. Identifizieren Sie die relevanten Akteure. Wenn es sinnvoll ist, verwenden Sie jede der drei möglichen Beziehungen zwischen Anwendungsfällen. Bedingungen, für die es keine eigene Notation gibt, stellen Sie durch Notizen dar.



Hinweis: Das Diagramm ist unvollständig: An der „extends“-Bezeihung muss auch noch die dazugehörige Bedingung („Falscher Fingerabdruck“) dran stehen.

b) (**4,5 Punkte**) Fertigen Sie für den Anwendungsfall „iPad entsichern“ eine ausführliche textuelle Spezifikation an. Diese soll die folgenden Elemente beinhalten.

Name des Anwendungsfalls: iPad entsichern (0,5)

Akteure:	Benutzer (0,5). Häufiger Fehler: Das iPad ist nicht ein Akteur der mit dem System interagiert, sondern das modellierte System (→ Es taucht somit als Name des Kastens auf, der um die Use-Cases herum gemalt wird – s. vorherige Seite).
Anfangsbedingung:	Das Gerät ist gesichert (0,5).
Ereignisfluss:	1. Der Benutzer streicht mit dem Finger über das iPad (0,5) 2. Der Fingerabdruck des Benutzers wird analysiert (0,5) 3. Das iPad wird entsichert (0,5)
Endbedingung:	Das Gerät ist entsichert (0,5)
Ausnahmen:	Der Fingerabdruck ist falsch (→ keine Entsicherung) (0,5)

Bemerkungen:

(i) Der Ereignisfluss stellt den Ablauf des „Normalfalls“ dar. Das ist hier der Fall, dass der Fingerabdruck korrekt ist.

(ii) Die Anzeige der Sperrmeldung ist eine „Extension“ des Use Case „Fingerabdruck analysieren“, der im hier beschriebenen „Entsichern“ Use Case an Schritt 2 im Ereignisfluss aufgerufen wird. Daher wird sie hier NICHT noch mal beschrieben.

Aufgabe 5. *Analyse (11,5 Punkte)*

Im Rahmen der Anforderungserhebung für das Abspielen einer DVD in einem DVD-Player wurden folgende Objekte identifiziert:

- DVD
- Kontrolltastenfeld
- Abspielsoftware
- Bildschirm

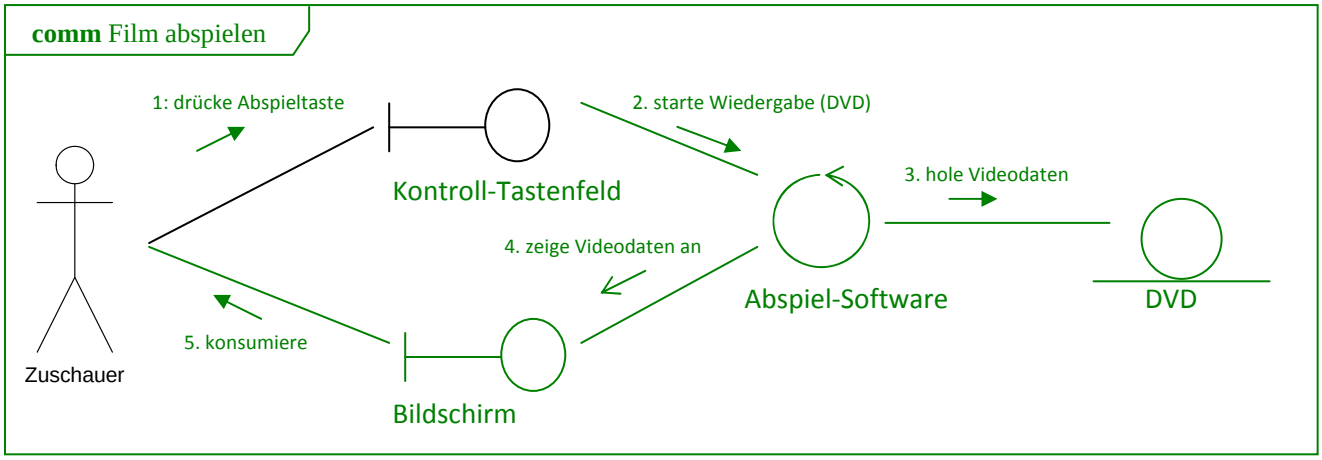
a) (**3,5 Punkte** – *je 0,5*) Klassifizieren Sie die Elemente in die Stereotypen des Analysemodells

- Boundary (0,5)
  - ⇨ Kontrolltastenfeld (0,5)
  - ⇨ Bildschirm (0,5)
- Controller (0,5)
  - ⇨ Abspielsoftware (0,5)
- Entity (0,5)
  - ⇨ DVD (0,5)

b) (**8 Punkte** – *je 0,5 pro Element und Beschriftung – keine*) Der Anwendungsfall „Film abspielen“ wird folgenderweise beschrieben:

Name:	Film abspielen
Akteure:	Zuschauer
Anfangsbedingung:	Die DVD befindet sich im Player.
Ereignisfluss:	1. Der Zuschauer drückt die Abspieltaste
	2. Der DVD-Player startet die Film-Wiedergabe
Endbedingung:	Der Film wird auf dem Bildschirm wiedergegeben.
Sonderfälle:	keine.

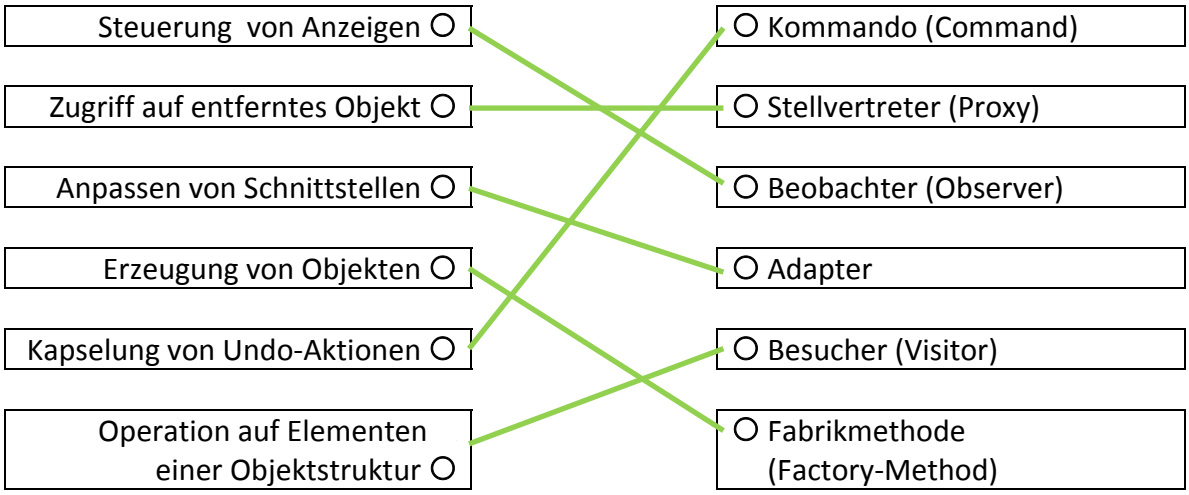
Vervollständigen Sie das folgende Kommunikationsdiagramm, so dass es die Interaktion des Akteurs mit den in der Aufgabenstellung genannten Objekten (DVD, ...) wiedergibt. Geben Sie dabei insbesondere auch die Nachrichten des Ereignisflusses an und verfeinern Sie den Ablauf von „DVD-Player startet die Film-Wiedergabe“ sinnvoll zu einer Interaktion die alle vier Objekte einbezieht.



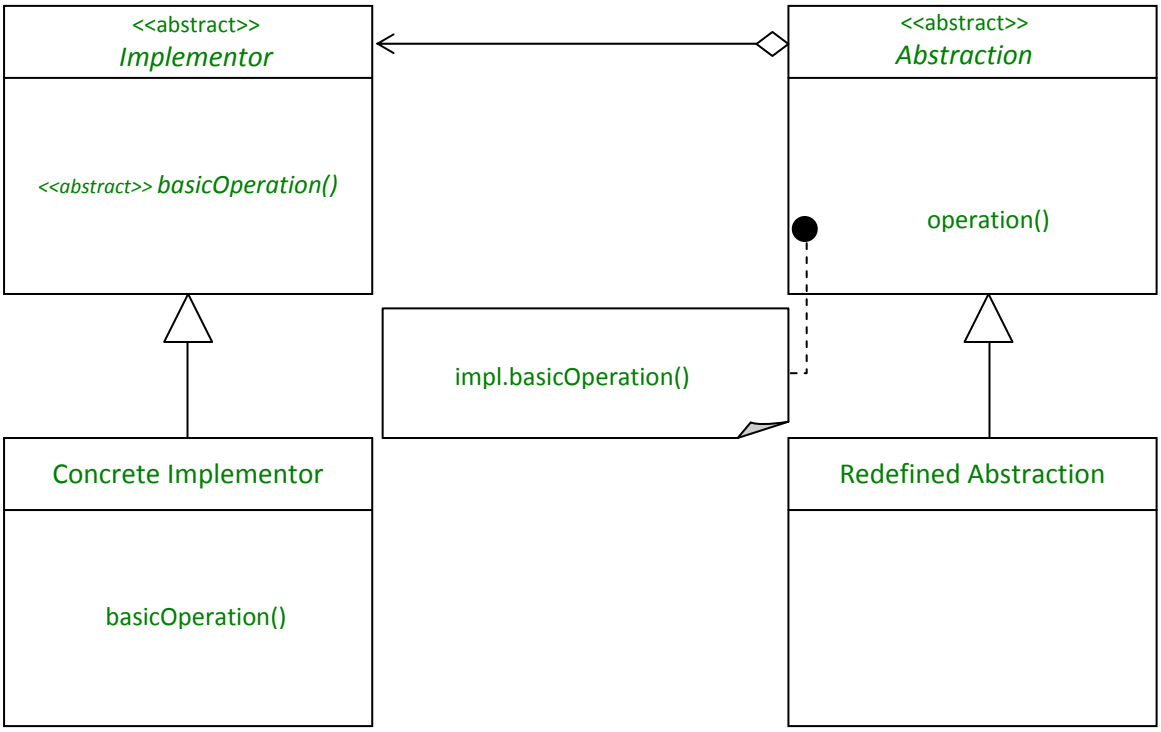
Aufgabe 6. *Entwurfsmuster theoretisch (9 Punkte)*

a) (**3 Punkte** – *je 0,5*) Verbinden Sie die folgenden Anwendungs-Beispiele jeweils mit dem Entwurfsmuster, das sich zur Implementierung anbietet:

- Eine *richtige* Verbindung zählt +0,5 Punkte.
- Eine *fehlende* Verbindung zählt 0 Punkte.
- Eine *falsche* Verbindung zählt –0,5 Punkte.



b) (**6 Punkte** – *je 0,5*) Ergänzen Sie in folgender allgemeiner Darstellung des Bridge-Entwurfsmusters die Klassennamen, Methodensignaturen, Stereotypen und das als Notiz eingezeichnete Codefragment. Jeder Klassen- und Methodennamen soll dabei die Rolle des jeweiligen Elementes im Entwurfsmuster ausdrücken.

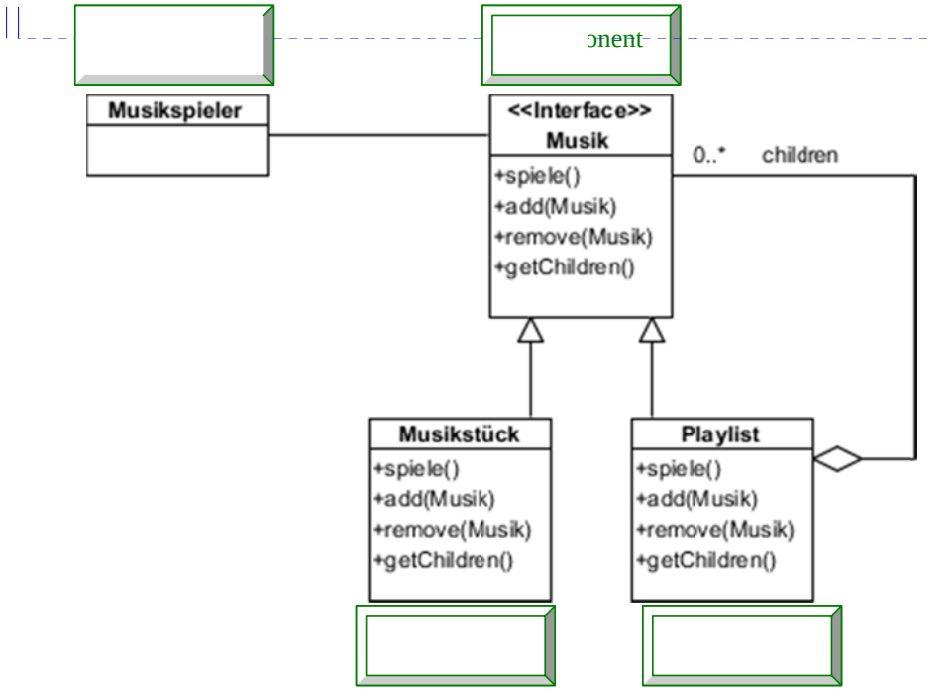


Aufgabe 7. *Entwurfsmuster angewandt (11+1 Punkte)*

Ein digitaler *Musikspieler* kann *Musik* abspielen. Dazu ruft der Musikspieler die *spiele()*-Methode der Musik auf. Musik ist entweder ein einzelnes *Musikstück* oder eine *Abspielliste* (Playlist), die wiederum Musik enthalten kann.

a) (**7+1 Punkte** – *1 = Musikspieler und abspielen(), 1 = <<interface or abstract>> Music, 1 = Musikstück und erben von Musik, 1= Playlist und erben von Musik, 0,5 = Aggregation, 0,5 = Rollennamen und Multiplizität, 1 = Body von Playlist::spiele(), 2 = Korrekte 4 methoden*)

Modellieren Sie den angegebenen Sachverhalt mit Hilfe eines geeigneten Entwurfsmusters und zeichnen Sie das entsprechende Klassendiagramm. Geben Sie auch den Code der *spiele()*-Methode der Klasse Playlist an (als Notiz).



Kommentar [g2]: Falls in Ihrem PDF-Viewer der Inhalt der knopfartigen Kästen nicht angezeigt wird: Sie geben die Rolle der jeweiligen Klasse in dem Pattern an. Von links nach rechts und oben nach unten steht: ***Client, Component, Leaf, Composite.***

b) (**2 Punkte** – *je 0,5*) Geben Sie zu jeder Klasse aus (a) die Rollen an, die sie in dem Entwurfsmuster spielt (sie können sie direkt an die jeweilige Klasse als Notiz anfügen).

c) (**2 Punkte** – *je 0,5*) Erläutern Sie allgemein, in welchen Fällen das Entwurfsmuster angewandt werden kann.

Das Composite-Muster kann allgemein eingesetzt werden zur Modellierung

- *rekursiver* (0,5) hierarchischer (0,5) Strukturen,
- in denen sich zusammengesetzte Objekte (0,5) genauso verhalten können wie jedes einzelne Objekt (0,5).

Aufgabe 8. *CRC-Karten und Design by Contract (9 Punkte)*

In der Informatik-Fachschaft können Prüfungsprotokolle in Ordnern ausgeliehen werden. Dabei darf jede Studierende, die durch ihre Matrikelnummer identifizierbar ist, jederzeit nur einen Ordner ausleihen, und jeder Ordner kann jederzeit an maximal eine Studierende ausgeliehen werden.

a) (**7 Punkte** – *je 1 Punkt pro Bereich der CRC-Karte*) Als Objekte wurden bereits *Fachschaft*, *Studierende,* und *Ordner* identifiziert. Unter Einsatz von CRC-Karten soll das Objektmodell um Verantwortlichkeiten und Kollaborationen erweitert werden, und zwar so, dass die Klassen anschließend alles beinhalten, was für folgende Szenarien erforderlich ist:

   a. Erfolgreiches Ausleihen eines Ordners
   b. Verweigertes Ausleihen eines Ordners

Klasse: Fachschaft **Beschreibung:** Verwaltet Prüfungsprotokolle in Ordnern und ermöglicht Studierenden die Ausleihe der Ordner.	
Verantwortlichkeiten (Responsibilities)	**Zusammenarbeit (Collaboration)**
Ausleihe eines Ordners an eine Studierende	Ordner, Studierende

Klasse: Studierende **Beschreibung:** Eine Person, die durch eine Matrikelnummer identifizierbar ist und Ordner entleihen kann.	
Verantwortlichkeiten (Responsibilities)	**Zusammenarbeit (Collaboration)**
Überprüfen ob diese Studierende bereits einen Ordner entliehen hat. *alternativ* Abfragen, welchen Ordner diese Studierende bereits entliehen hat.	---

Klasse: Ordner **Beschreibung:** Ein Element, das Prüfungsprotokolle enthält.	
Verantwortlichkeiten (Responsibilities)	**Zusammenarbeit (Collaboration)**
Überprüfen ob dieser Ordner bereits ausgeliehen ist. *alternativ* Abfragen, an welche Studierende dieser Ordner entliehen ist.	---

b) (**2 Punkte** – *1 = pre, 1= post -- jede Teilbedingung = 0,5*) Geben Sie die Vor- und die Nachbedingung der „Entleihe“-Operation an. Definieren Sie dazu – in Übereinstimmung mit Ihrer Lösung aus (a) – sinnvolle Methodensignaturen und bilden Sie daraus logische Ausdrücke in OCL-Syntax oder Java-Syntax.

context Fachschaft::entleihe(Studierende s, Ordner o)

a. **pre**:
not(o.istAusgeliehen()) && not(s.hatOrdnerEntliehen())

b. **post:**

o.istAusgeliehen() && s.entliehenerOrdner()=o