- Musterlösung -

Probeklausur zur Vorlesung „Softwaretechnologie“

Dr. Günter Kniesel

Institut für Informatik III
Universität Bonn

29. Januar 2020

Name: _____

Matrikelnummer: _____

Tutor(in): _____

Aufgabe		Zeit ca.	Maximale Punkte	Erreichte Punkte
1	UML-Aktivitätsdiagramm	10 Min.	11,0	
2	UML-Sequenzdiagramm	10 Min.	9,0	
3	UML-Zustandsdiagramm	10 Min.	9,0	
4	Anwendungsfalldiagramm	11 Min.	10,5	
5	Analysemodell	12 Min.	13,0	
6	Design by Contract	9 Min.	9,0	
7	CRC-Karten	12 Min.	10,5	
8	Stabilität, Abhängigkeit, Ersetzbarkeit	14 Min.	13,0	
9	Entwurfsmuster	15 Min.	14,0	
Summe		103 Min.	99	
Note				

Wertung: 39,0 → **4,0** | 43,5 → **3,7** | 47,5 → **3,3** | 52,0 → **3,0** | 56,5 → **2,7** | 60,5 → **2,3** | 65,0 → **2,0** | 69,5 → **1,7** | 73,5 → **1,3** | 78,0 → **1,0** |

Dies ist eine Probeklausur!

Die Probeklausur, soll Sie auf den Umfang, die Form und die Art der Aufgabenstellungen in der Klausur vorbereiten. Versuchen Sie nicht, aus den Inhalten der Probeklausur auf den Inhalt der Klausur zu schließen. Das Einzige, was Sie generalisieren können, ist die Art der Aufgabenstellungen.

Das Bewertungsschema, das auf der ersten Seite angegeben ist, soll Ihnen nach der Korrektur eine reale Selbsteinschätzung ermöglichen, wo Sie zum heutigen Zeitpunkt stünden, falls das jetzt die echte Klausur wäre. Die laut Bewertungsschema zustande kommende Note hat keinen Einfluss auf die Klausurnote oder die Klausurzulassung und dient nur Ihrer Orientierung.

Weitere Hinweise

Die 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 anderweitig deutlich markieren.*

Sie dürfen auch die Rückseiten der Blätter mitbenutzen. ***Wenn Sie für Ihre Antworten zusätzliche Blätter brauchen, erhalten Sie diese von den Betreuern.***

Bei einigen Aufgaben gibt es eine Lösungsschablone*, die Sie ausfüllen sollen. Sollten Sie einen zweiten Versuch starten wollen, melden Sie sich durch Handzeichen. Sie bekommen dann einen frischen Ausdruck der entsprechenden Schablone. Streichen Sie in diesem Fall die vorherige Lösung eindeutig durch.*

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!*

Bewertungsschema

Bei jeder Aufgabe wird angegeben wofür es (Plus)Punkte gibt.

Von der ermittelten Punktzahl werden anschließend je Kategorie von syntaktischen Fehlern (falsche Pfeile, falsche Kästen, …) 0,5 Punkte abgezogen. Wenn z. B. 5 Linien keine Pfeilspitzen haben, wird nur ein mal 0,5 abgezogen denn das sind wiederholte Fehler der gleichen Kategorie. Wenn es aber zusätzlich einen eckigen Endzustand gibt (neue Fehlerkategorie) wird insgesamt 1 Punkt abgezogen.

Man kann in einer Aufgabe nicht weniger als 0 Punkte erreichen.

Aufgabe 1. *UML-Aktivitätsdiagramm (11 Punkte)* -- *10 Min.*

Anlässlich der Einstellung neuer Kellner hat ein Restaurantbetreiber folgenden Leitfaden erstellt:

Sobald ein neuer Gast das Restaurant betritt, fragen Sie nach, ob er reserviert hat.

Falls ja, suchen Sie im Reservierungsbuch nach dem ihm zugewiesenen Platz. Falls nicht, prüfen Sie nach ob es trotzdem noch freie Plätze gibt.

Anschließend begleiten Sie den Gast zu seinem Platz oder vertrösten ihn, wenn es keinen freien Platz mehr gibt.

Einen Gast, der einen Platz bekommen hat, fragen Sie ob er schon ein Getränk haben möchte und reichen ihm die Karte (in beliebiger Abfolge). Danach geben sie seinen Getränkewunsch an die Küche weiter.

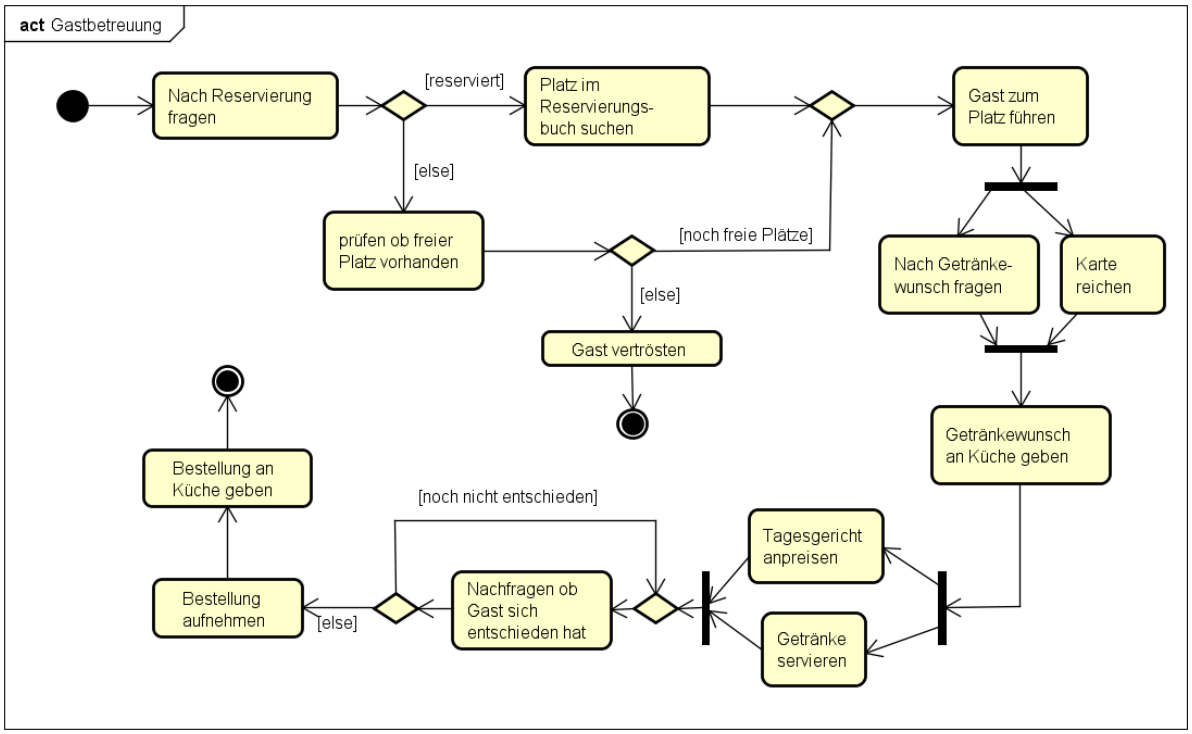
Während Sie ihm die bestellten Getränke servieren, machen Sie ihn noch einmal auf das aktuelle Tagesgericht aufmerksam und fragen Sie danach nach seiner Essens-Bestellung.

Wiederholen sie den Versuch die Essensbestellung aufzunehmen solange bis sich der Gast entschieden hat und geben diese dann an die Küche weiter.

Ab hier übernimmt dann ein anderer Kellner und Sie sind mit diesem Gast fertig.

Stellen Sie die beschriebenen Sachverhalte in einem UML-**Aktivitätsdiagramm** mit dem Titel „Gastbetreuung“ dar und achten Sie dabei auf vollständige und korrekte UML-Notation.

Lösungsvorschlag:



Bewertung: Je *0,5 Punkte pro*

- *Aktivität,*
- *Paar von Verzweigungsknoten (einschl. zugehörigen Bedingungen),*
- *Paar von Parallelisierungsknoten*
- *Startzustand,*
- *Endzustand,*
- *Diagrammrahmen incl. Beschriftung.*

Aufgabe 2. *UML-Sequenzdiagramm (9 Punkte)* -- *10 Min.*

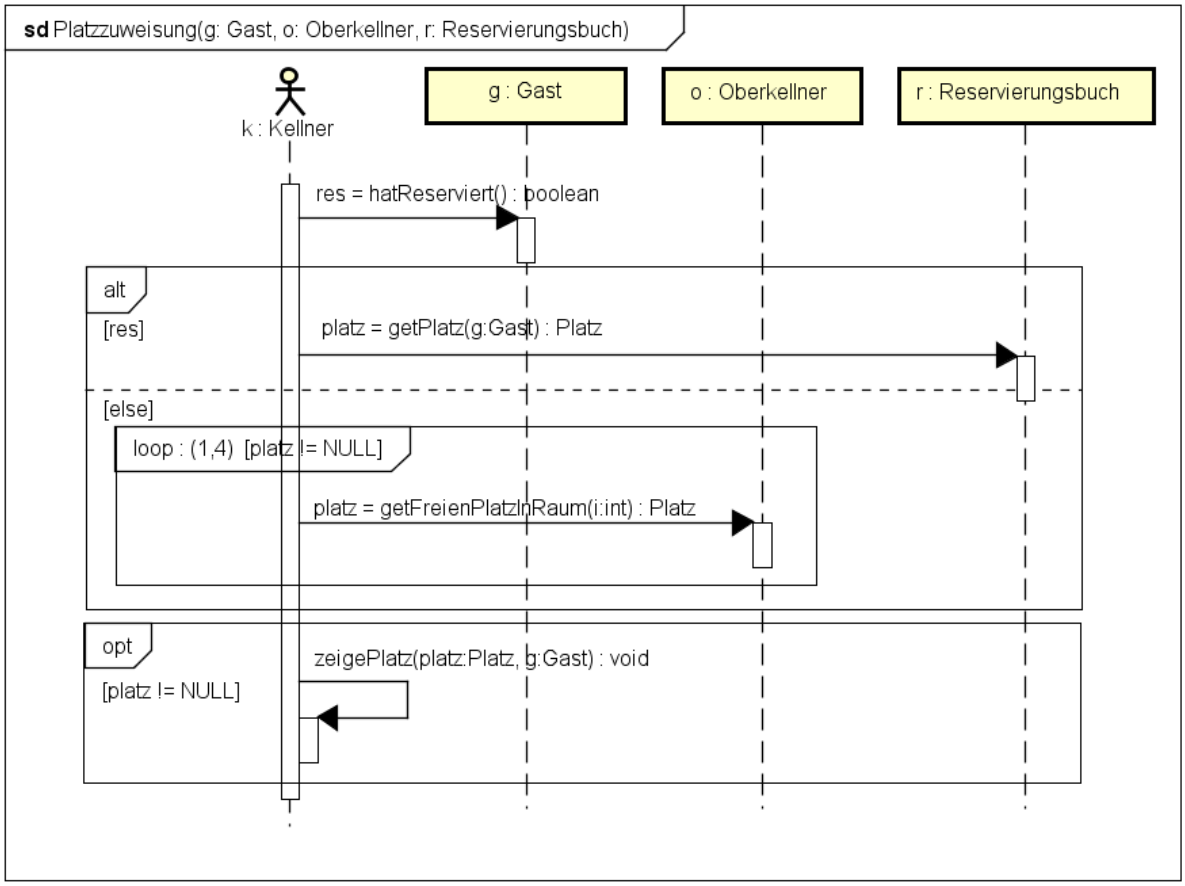
Diese Aufgabe behandelt den in der vorherigen Aufgabe bereits erwähnten Vorgang des Platzzuweisens genauer:

- *Der Kellner fragt zunächst den Gast ob er reserviert hat.*
- *Falls ja, sucht er im Reservierungsbuch nach dem zugewiesenen Platz.*
- *Falls nein, fragt er den Oberkellner ob noch Plätze im ersten Raum frei sind. Solange der Oberkellner dies verneint, fragt er dies für jeden der* ***weiteren 3*** *Räume.*
- *Falls es durch die Reservierung oder den Oberkellner einen freien Platz gibt, führt der Kellner den Gast zu seinem Platz.*

Modellieren Sie den beschriebenen Sachverhalt in einem UML-Sequenzdiagramm in dem der Kellner als *Akteur* auftritt und die Referenzen auf den Gast, das Reservierungsbuch und den Oberkellner als *Parameter* übergeben werden. Achten Sie auf die Angabe sinnvoll *typisierter* Methoden und Parameter und auf eine vollständige Beschriftung aller Teile Ihres Diagramms. Nutzen Sie dabei folgende vereinfachenden Annahmen:

- Der Oberkellner stellt eine Funktion `getFreienPlatzInRaum` zur Verfügung, welche als Parameter eine Raumnummer erwartet und ein Objekt vom Typ `Platz` zurückgibt. (Oder NULL falls es keinen freien Platz gibt).
- Innerhalb einer Schleife können Sie auf die Iterationsvariable i zugreifen.

Lösungsvorschlag:



Bewertung:

- 1 Punkt für den Diagrammrahmen mit Beschriftung und Parametern.
- 0,5 Punkte pro beteiligtes Objekt/Akteur einschließlich Lebenslinie.
- 0,5 Punkte pro Methodenaufruf
- 0,5 Punkte pro Rückgabetyp (falls nicht *void*).
- 0,5 Punkte für den *opt*-Kasten mit richtiger Beschriftung und Bedingung
- 1 Punkt für den *loop*-Kasten mit richtiger Beschriftung und Bedingung
- 1 Punkt für den *alt*-Kasten mit richtiger Beschriftung und Bedingung

Je -0,5 Punkte (Abzug) pro Kategorie von syntaktischen Fehlern, aber mindestens 0 Punkte insgesamt in der Aufgabe

Aufgabe 3. *UML-Zustandsdiagramm (9 Punkte)* -- *10 Min.*

Im Folgenden werden die Abläufe und Gemütszustände eines sehr speziellen Gastes näher betrachtet:

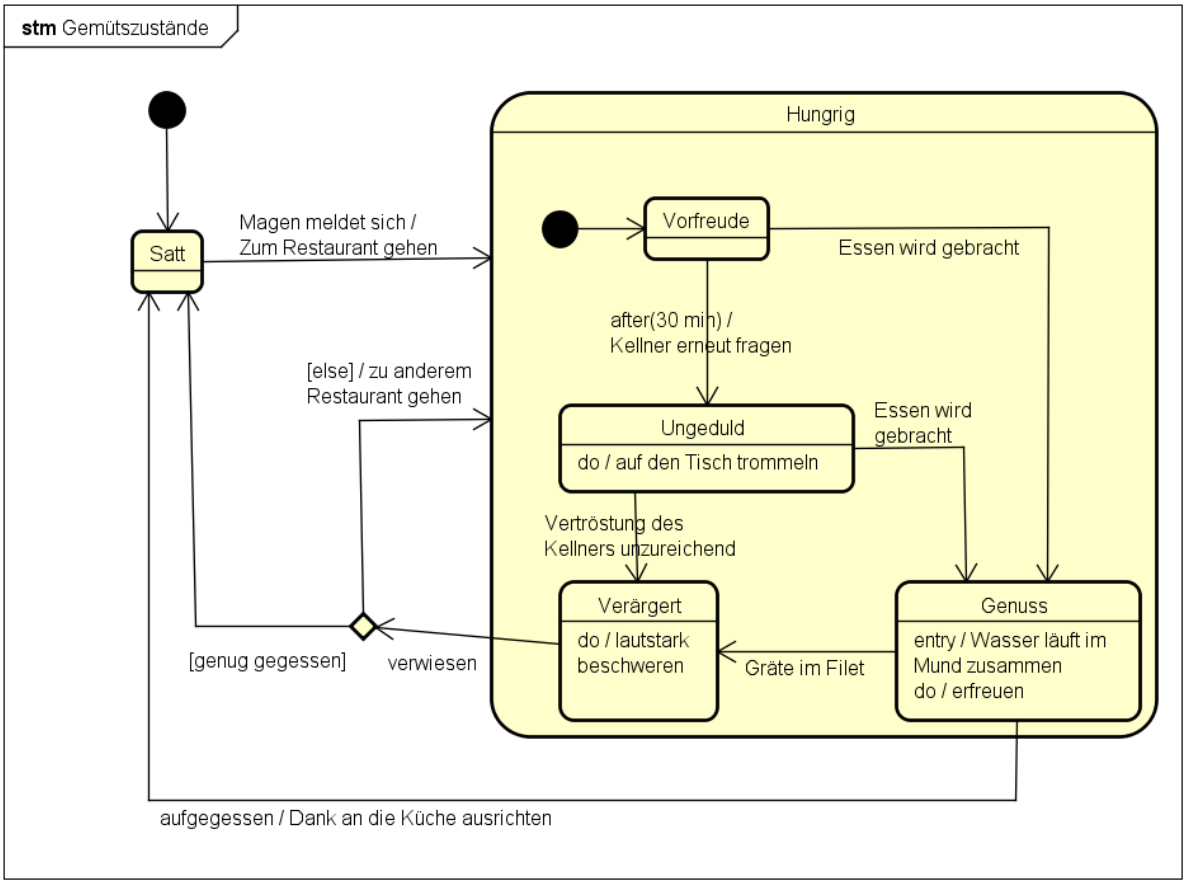
- Hans ist zunächst **satt**.
- Sobald sein Magen sich meldet, merkt er, dass er **hungrig** ist. Daher geht er zu seinem Lieblingsrestaurant und die **Vorfreude** beginnt.
- Wurde sein Seelachsfilet jedoch nach 30 Minuten immer noch nicht gebracht schlägt die Vorfreude in **Ungeduld** um und er fragt den Kellner erneut wann sein Essen kommt. Während er so wartet trommelt er (noch leise) auf den Tisch.
- Wenn das geliebte Seelachsfilet gebracht wurde, beginnt der **Genuss**. Zu Beginn läuft ihm dabei das Wasser im Mund zusammen und während er seinen Fisch genießt erfreut er sich an der leckeren Zubereitung.
- Sind ihm die Vertröstungen des Kellners nicht ausreichend genug während er ungeduldig wartet, oder er findet eine Gräte während er seinen Fisch genießt, schlägt sein Gemütszustand sofort in **Verärgerung** um. Dies äußert sich darin, dass er sich pausenlos lauthals beschwert. Dann hilft es auch nicht mehr, wenn ihm Essen gebracht wird.
- Dies wird den Kellnern jedoch irgendwann zu bunt und **verweisen** Hans aus dem Restaurant. Hatte er bisher schon genug gegessen ist er zwar nicht glücklich (das soll hier nicht modelliert werden) aber immerhin wieder **satt**. Ansonst – da er ja <u>immer noch</u> **hungrig** ist – bleibt ihm nichts anderes übrig, als zu einem anderen Restaurant zu gehen (welches Seelachs anbietet) und die **Vorfreude** beginnt von vorne.
- Hat er jedoch ohne Zwischenfälle seinen Fisch **aufgegessen**, ist Hans danach stets **satt** und lässt seinen Dank der Küche ausrichten.

Modellieren Sie die Gemütszustände mit den dazugehörigen Aktionen in einem vollständig beschrifteten UML-Zustandsdiagramm. Versuchen Sie dabei, redundante Zustände, Übergänge, Bedingungen und Aktionen zu vermeiden. Beschränken Sie Ihre Modellierung auf Angaben, die ausdrücklich im obigen Text aufgeführt sind.

Nutzen Sie für Ihre Lösung den Platz auf der nächsten Seite.

(Platz für die Lösung zu Aufgabe 3)

Lösungsvorschlag:



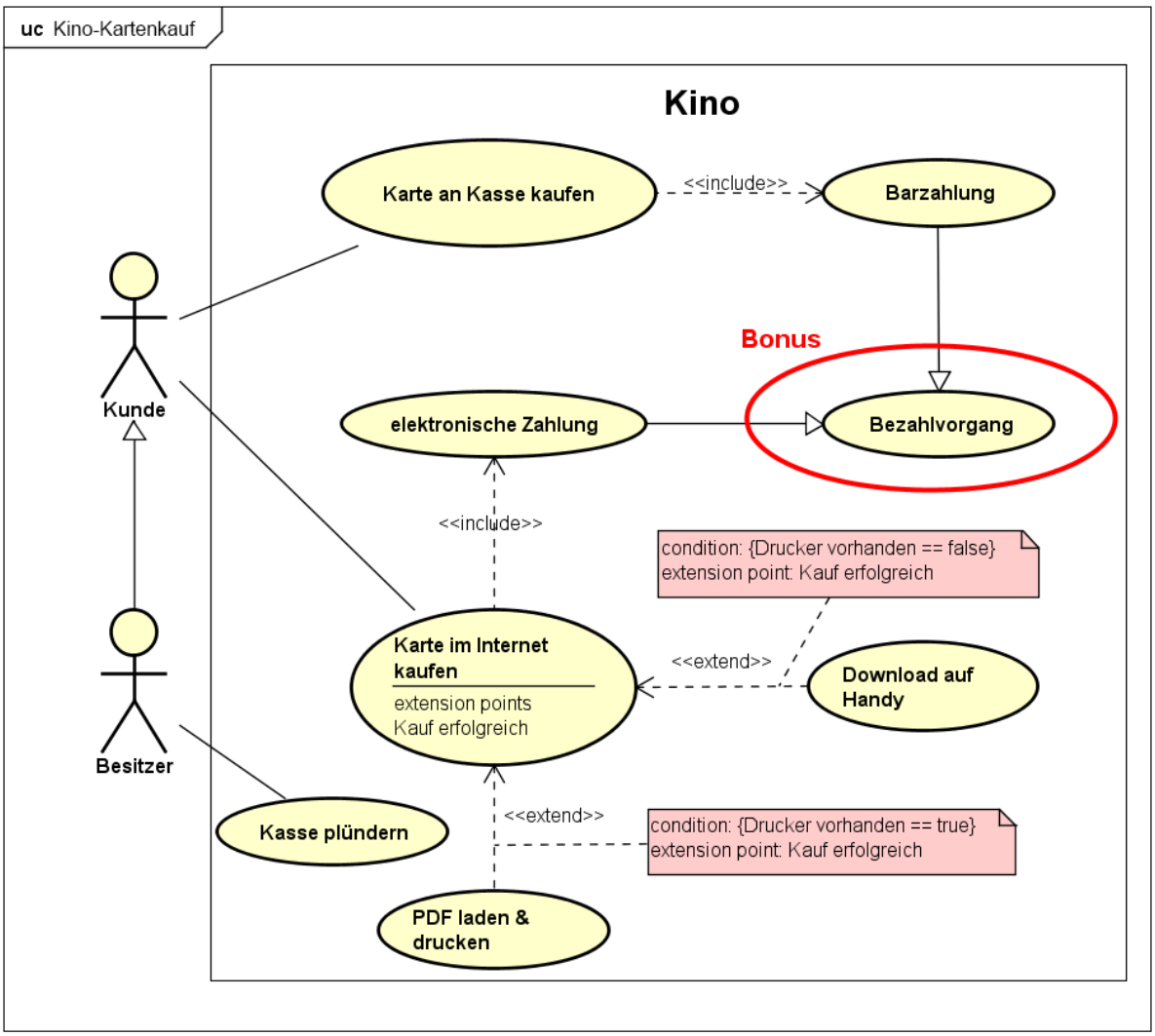
Bewertung: Je 0,5 Punkte pro

- Rahmen und Beschriftung.
- Startzustand (der Startzustand im *Hungrig*-Zustand gibt keine zusätzlichen Punkte)
- Zustand (bei insgesamt 6 Zuständen),
- 8 Ereignisse (für das Ereignis *„after(30 min)“* nur bei Verwendung dieser Notation)
- 2 Bedingung und 8 Aktionen

Aufgabe 4. *UML- Anwendungsfalldiagramm (10,5 Punkte + 1 Bonus)* -- *11 Min.*

Modellieren Sie für das reale System „Kino“ durch ein **Anwendungsfalldiagramm** den Kauf einer Kinokarte. Dafür sollen Sie auf folgende Aspekte achten:

- Kunden können Karten entweder an der Kasse oder im Internet kaufen.
- Der Kauf an der Kasse beinhaltet immer eine Barzahlung.
- Der Kauf im Internet beinhaltet immer eine elektronische Zahlung.
- Der Kunde kann beim Internetkauf, nachdem dieser erfolgreich durchgeführt worden ist, sein Ticket als PDF laden und ausdrucken. Dies ist nur möglich, wenn er einen Drucker besitzt. Falls er keinen hat, lädt er die Karte als QR-Code auf sein Handy.
- Der Besitzer des Kinos kann alle Aktionen ausführen, welche auch vom Kunden ausgeführt werden. Zusätzlich kann er die Kasse plündern.



Bewertung: Je 0,5 Punkte pro

- (1x **Rahmen und Beschriftung.**
- (1x **System (Rahmen)**
- (2 x) **Akteur**
- (7x + 1 Bonus) **Usecase,**
- (2x communication 2x includes, 2x extends + 1x Generalisation, als Bonus 2 x Generalisation) **Beziehung**
- (2x) **Bedingung**
- (1x) **Extension Point**

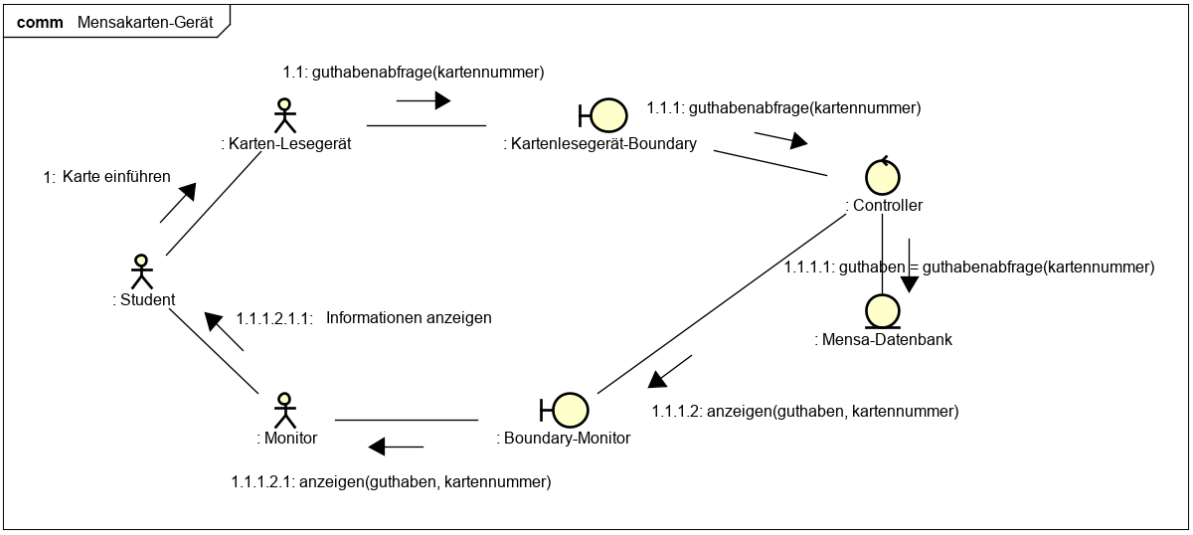
Aufgabe 5. *Kommunikationsdiagramm / Analysemodell* (*13 Punkte*) -- *12 Min.*

Erstellen Sie ein Kommunikationsdiagramm für die Guthabenabfrage am Mensakarten-Gerät und verwenden Sie für die kollaborierenden Objekte die im **Analyse-Modell** üblichen grafischen Stereotypen.
Es sollen folgende Schritte modelliert werden:

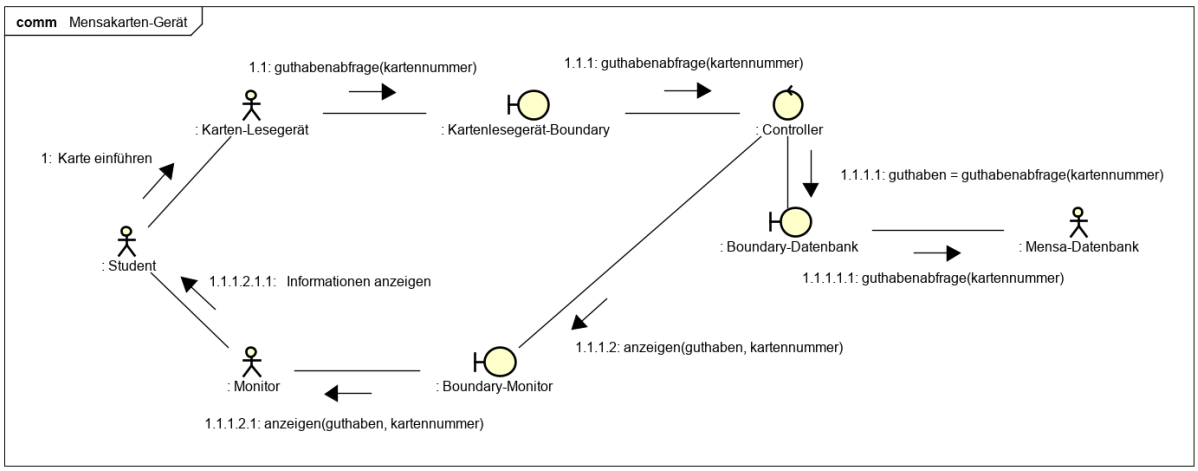
- Nach dem Einführen der Karte in das Karten-Lesegerät wird die Kartennummer ausgelesen.
- Unter Angabe der Kartennummer wird das Guthaben aus der Mensadatenbasis ausgelesen.
- Anschließend wird das momentane Guthaben über den Monitor angegeben.

Sie dürfen annehmen, dass alle Komponenten des Systems bereits initialisiert sind.

Tipp 1: Nummerieren Sie die Nachrichten in Punktnotation.
Tipp 2: Achten Sie darauf, auch den aus obigem Text entnehmbaren Datenfluss durch passende Parameter und Ergebnisse zu modellieren. Zur Zeitersparnis dürfen Sie die *Typen* von Parametern und Ergebnissen weglassen.
Tipp 3: Achten Sie darauf, auch zu modellieren, dass das Karten-Lesegerät und der Monitor aus Sicht der Entwickler des Mensa-Kartengerätes externe Systeme sind.



Alternative-Variante mit Datenbank als externem System:



Bewertung Variante 1: Je 0,5 Punkte pro

- (1x) Rahmen und Beschriftung
- (3 x) Akteur
- (7x) Klasse mit korrektem Stereotyp,
- (7x) Nachricht mit korrekter Beschriftung und Nummerierung
- (1x) Rückgabetyp Guthaben
- (7x) Beziehung

Bewertung Variante 2: Je 0,5 Punkte pro

- (1x) Rahmen und Beschriftung
- (3 x) Akteur
- (6x) Klasse mit korrektem Stereotyp,
- (1x) Datenbank Boundary und Akteur
- (7x) Nachricht mit korrekter Beschriftung und Nummerierung
- (1x) Rückgabetyp Guthaben
- (7x) Beziehung

Aufgabe 6. *Design by Contract (9 Punkte)* -- *9 Min.*

Bei den folgenden Multiple-Choice-Fragen zählen korrekte Antworten 0,5 Punkte, falsche -0,5 Punkte. Man kann pro Teilaufgabe nicht weniger als 0 Punkte erreichen.

a) **(1,5 Punkte)** Legale Zustände werden beschrieben durch…

	Wahr	Falsch
Invarianten	x	
Vorbedingungen		x
Nachbedingungen		x

b) **(1,5 Punkte)** Es gelten die folgenden Verantwortlichkeiten:

	Wahr	Falsch
Vorbedingungen werden vom Aufrufer sichergestellt	x	
Nachbedingungen werden nicht vom Aufrufer sichergestellt	x	
Invarianten werden durch die gleiche Vorbedingung im Aufrufer und im Aufgerufenen umgesetzt		x

c) **(1 Punkt) Vervollständigen** Sie folgende Sätze:

Bei DBC ist eine Assertion … eine logische Bedingung, die wahr sein muss. **(0,5P)**

Bei DBC ist ein Contract … eine Menge von Assertions, die festlegen, wie zwei Partner miteinander interagieren. **(0,5P)**

Beantworten Sie die folgenden Fragen:

d) **(0,5 Punkte)** Wie beziehen Sie sich in OCL auf den Wert einer Variablen *vor* Ausführung einer Operation? value@pre

e) **(0,5 Punkte)** In der Formulierung welcher Art(en) von Assertions wird der Ausdruck aus Aufgabe d) verwendet? In der Nachbedingung.

f) **(1Punkt)** Ist das Abfangen von AssertionErrors sinnvoll? Begründen Sie Ihre Antwort.
Es ist nicht sinnvoll, da Abfangen zum Aufblähen des Codes führt (denn Errors können überall auftreten) und außerdem Assertions nur zur Fehlersuche gedacht sind → Stack Trace

g) **(1 Punkt)** Wie würden Sie die Logik einer Assertion in Java ausdrücken, wenn es Assertions nicht als Sprachelement gäbe?
Bedingungen per if-Abfragen und eigene Exceptions werfen.

h) **(1 Punkt)** Welchen Vorteil bietet die Verwendung von Assertions gegenüber Ihrer Lösung zu g)?
Lesbarer, ausschaltbar

i) **(0,5 Punkte)** Nehmen Sie an, es gibt eine Klasse „Maths“ und darin die Methode „sqrt“ für das Ziehen der Wurzel aus einer positiven reellen Zahl. Formulieren Sie eine sinnvolle OCL-Vorbedingung für diese Methode.
context Maths::sqrt(float x) **pre**: x >= 0

j) **(0,5 Punkte)** Setzen Sie obige Vorbedingung als Java-Assertion um.
assert x >= 0: ‘‘Die Wurzelfunktion ist nur für x >= 0 definiert‘‘;

Aufgabe 7. *CRC-Karten (10,5 Punkte)* -- *12 Min.*

a) **(3 Punkte)** Nennen Sie die wesentlichen Ziele der Nutzung von CRC-Karten.
- Reduktion von zu vielen Verantwortlichkeiten an einer Stelle **(0,5 P)**
- Fokus auf Schnittstelle statt Daten **(0,5 P)**
- Identifikation von Operationen, Parametern, Ergebnistypen und evtl. Assoziationen **(2,0 P)**

b) **(1 Punkt)** In welcher Aktivität werden CRC-Karten eingesetzt?
In der Anforderungsanalyse und dem frühen Objektentwurf

c) **(6,5 Punkte)** Entwickeln Sie zu folgendem Sachverhalt CRC-Karten für die Klassen „Smart-TV“ und „HDMI-Anschluss“. Sie brauchen für andere Klassen keine eigenen CRC-Cards zu entwickeln, sollten sich aber trotzdem Gedanken machen, welche anderen Klassen es geben muss, damit Sie die Kollaborationen von „Smart-TV“ und „HDMI-Anschluss beschreiben können. Lesen Sie sich den Text gründlich durch!

- Ein internetfähiger Smart-TV hat drei HDMI-Anschlüsse. An jeden dieser Anschlüsse können Spielekonsolen oder Rechner angeschlossen werden.
- Wenn ein solches Gerät angeschlossenen wird, meldet der HDMI-Anschluss dies dem Smart-TV, der daraufhin den Benutzer fragt, ob er den aktuellen Eingang auf den Ausgang des neu angeschlossenen Gerätes umschalten soll.
- Der Smart-TV kann das am aktuell selektierten HDMI-Anschluss anliegende Signal am Bildschirm anzeigen.
- Smart-TVs, Rechner und Spielekonsolen können ein- und ausgeschaltet werden.
- Um mit dem Smart-TV ins Internet zu gehen, muss dieser sich zunächst mit einem Router verbinden.

Class: Smart-TV	
Responsibilities	**Collaboration**
Bild zeigen (0,5P)	HDMI-Anschluss (0,5P)
Anschluss selektieren (0,5P)	HDMI-Anschluss (0,5P)
Online gehen (0,5P)	Router (0,5P)
Ein-/Ausschalten (0,5P)	-

Class: HDMI-Anschluss	
Responsibilities	**Collaboration**
Anschließen (0,5P)	Rechner, Spielekonsole (0,5P + 0,5P)
Angeschlossenes Gerät melden (0,5P)	Smart-TV (0,5P)

1P für korrekte Form (Class: Klassenname, Responsibilities (Verantwortlichkeiten), Collaborations (Kollaborationen))

Aufgabe 8. *Stabilität, Abhängigkeit, Ersetzbarkeit (13 Punkte)* -- *14 Min.*

a) **(1 Punkt)** Erklären Sie, wann ein Typ B ein Subtyp von A ist. Gehen Sie dabei auch auf Vor- und Nachbedingungen ein.

Instanzen von B haben alle Methoden von A **(0,5P)**, mit gleichen oder stärkeren Nachbedingungen und gleichen oder schwächeren Vorbedingungen. **(0,5P)**

b) **(2,5 Punkte)** Erklären Sie die folgenden Begriffe sinngemäß (keine Formeln):

Efferente Abhängigkeiten:
Eigene Abhängigkeit von Anderen **(0,5P)**

Instabilität einer Klasse:
Wie leicht ein Typ änderbar ist **(0,5P)**, hohe Instabilität = leicht änderbar **(0,5P)**.

c) **(4 Punkte)** Nennen **und erklären** kurz Sie die vier Entwurfsprinzipien von Robert C. Martin.

Dependency Inversion Principle **(0,5P)**, Abhängigkeit nur zu gleich abstraktem oder Abstrakterem **(0,5P)**
Stable Dependencies Principle **(0,5P)**, Abhängigkeit nur zu gleich oder stärker Stabilem **(0,5P)**
Stable Abstractions Principle **(0,5P)**, Abstraktheit und Stabilität soll proportional sein **(0,5P)**

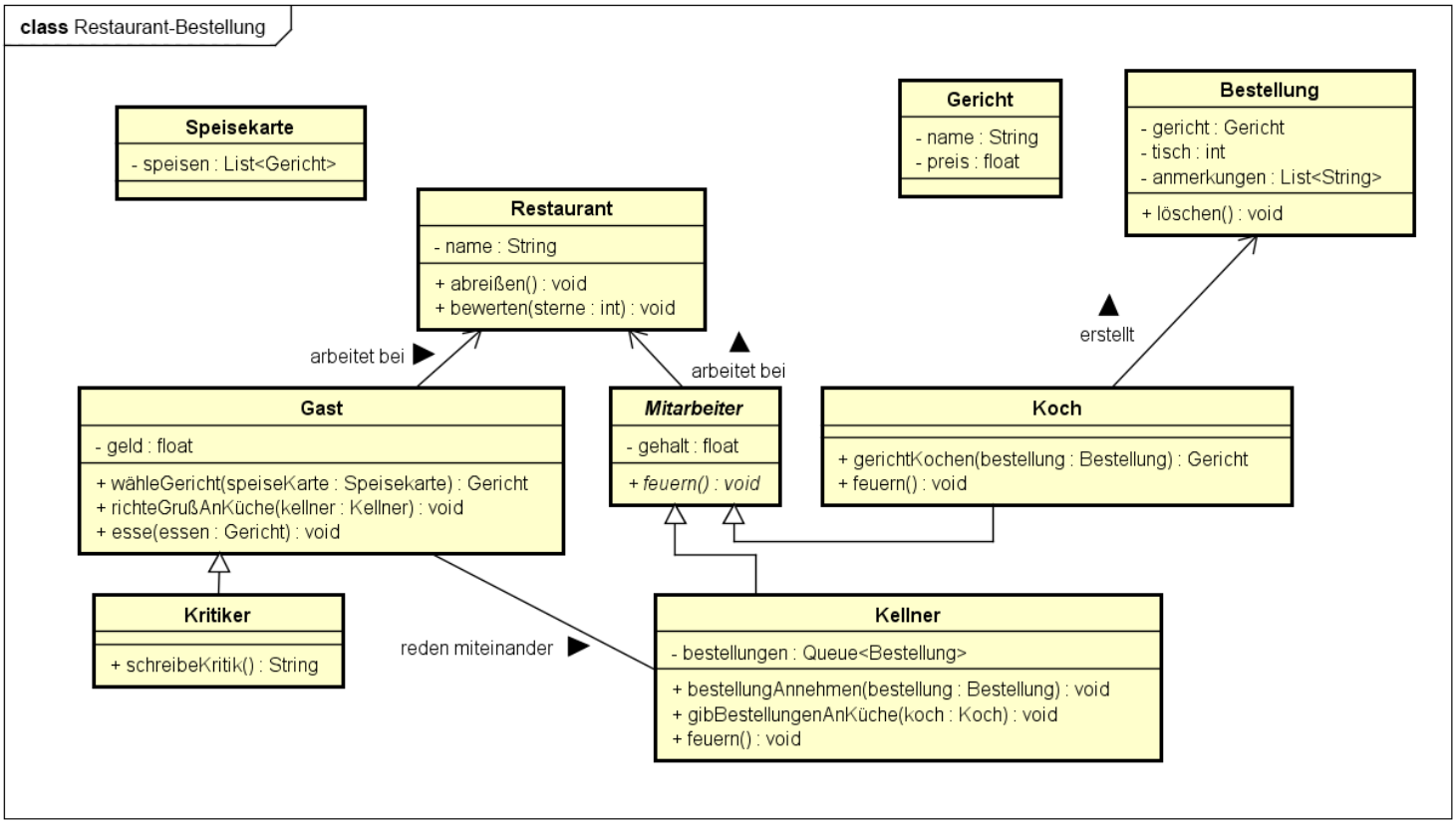
Acyclic Dependencies Principle **(0,5P)**, keine gegenseitigen Abhängigkeiten **(0,5P)**

d) **(1 Punkt)** Kann eine Klasse nur vollständig abstrakt oder vollständig konkret sein, oder gibt es auch hier Zwischenwerte? Begründen Sie Ihre Antwort.

Es gibt auch Zwischenwerte: Man berechnet das Verhältnis der Anzahl der abstrakten und konkreten Methoden.

e) **(4,5 Punkte)** Berechnen Sie zu folgendem Klassendiagramm für die Klassen „Restaurant“, „Mitarbeiter“ und „Bestellung“ die **Abstraktheit** und **Instabilität**. Füllen Sie dazu die unten vorgegebene Tabelle aus.

- Fangen Sie an, indem Sie in den Spalten **Afferent** und **Efferent** die Namen der Klassen eintragen zu denen entsprechende Abhängigkeiten bestehen.
- Dann berechnen tragen Sie in die Spalten **Instabilität** und **Abstraktheit** die Formeln ein die sie im jeweiligen konkreten Fall ausrechnen (also mit schon eingesetzten Zahlen entsprechend des jeweiligen Falles), sowie das Ergebnis.
- Abschließend tragen Sie in die Spalte Bewertung ein, ob Sie entsprechend der berechneten Werte die Klasse für gut wiederverwendbar, schlecht wiederverwendbar, nutzlos oder unwartbar halten.



Klasse	Afferent	Efferent	Instabilität	Abstraktheit	Bewertung
Restaurant	Gast, Mitarbeiter	String	1/(1+2) = 1/3	0/2 = 0	unwartbar
Mitarbeiter	Kellner, Koch	Restaurant	1/(1+2) = 1/3	1/1 = 1	gut wiederverwendbar
Bestellung	Koch, Kellner	Gericht, String, List	3/(3+2) = 3/5	0/1 = 0	schwer wiederverwendbar

Pro korrekter Instabilität 0,5P, pro korrekter Abstraktheit 0,5,
pro korrekter Bewertung 0,5P → 3 x 1,5 = 4,5 Punkte

Aufgabe 9. Entwurfsmuster (14 Punkte) -- *15 Min.*

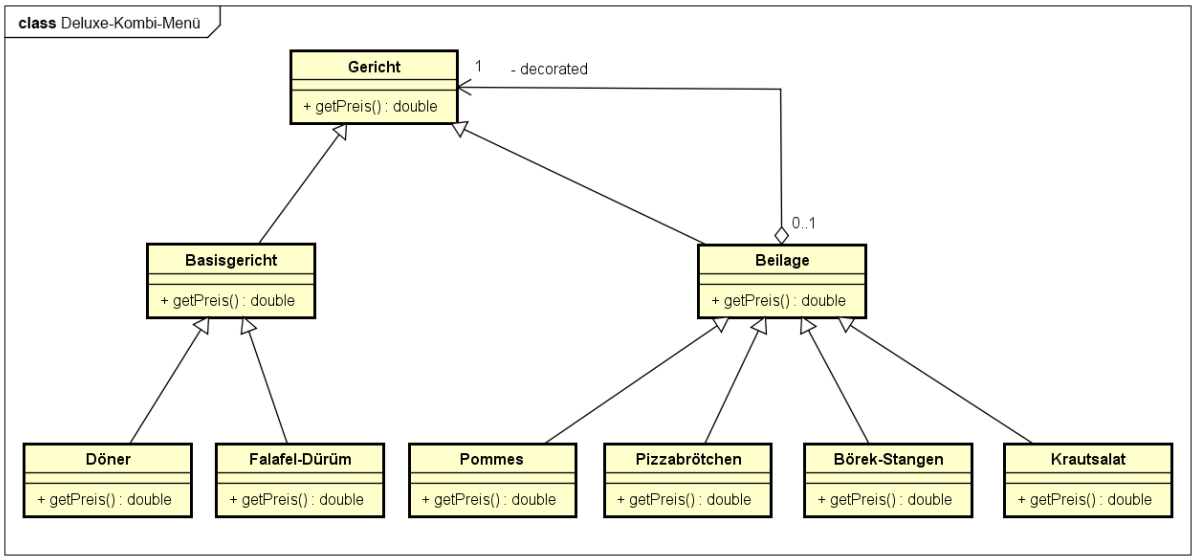
Gegeben ist folgendes Szenario: Der neue Döner-Laden um die Ecke möchte sein beliebtes Deluxe-Kombi-Menü, bei dem jeweils ein Basisgericht mit einer beliebigen Anzahl an Beilagen kombiniert werden kann, modellieren. Der Gesamtpreis für das Deluxe-Kombi-Menü soll die Summe der Preise der einzelnen Gerichte sein, wobei ein Basisgericht 3€ und jede Beilage 1,50€ kostet. Aktuell stehen als Basisgerichte nur Döner mit Hähnchenfleisch und Falafel-Dürüm auf der Karte, ebenso wie an Beilagen momentan nur Pommes, Pizzabrötchen, Krautsalat und Börek-Stangen verfügbar sind.

Sobald sich der Laden etabliert hat, möchte die Besitzerin jedoch ihr Angebot erweitern. Dafür muss das implementierte System ebenfalls erweitert werden. Das soll aber möglich sein, ohne irgendeine der vorhandenen Klassen zu verändern.

a) **(0,5 Punkte)** Welches Entwurfsmuster bietet sich hier an?
Decorator-Pattern

b) **(2 Punkte)** Ordnen Sie die typischen Rollen des Entwurfsmusters den Klassen, die Sie modellieren würden und für geeignet halten, zu.
Component: Gericht
Concrete Component: Basisgericht
Decorator: Beilage
Concrete Decorators: Pommes, Pizzabrötchen, Krautsalat, Börek-Stange

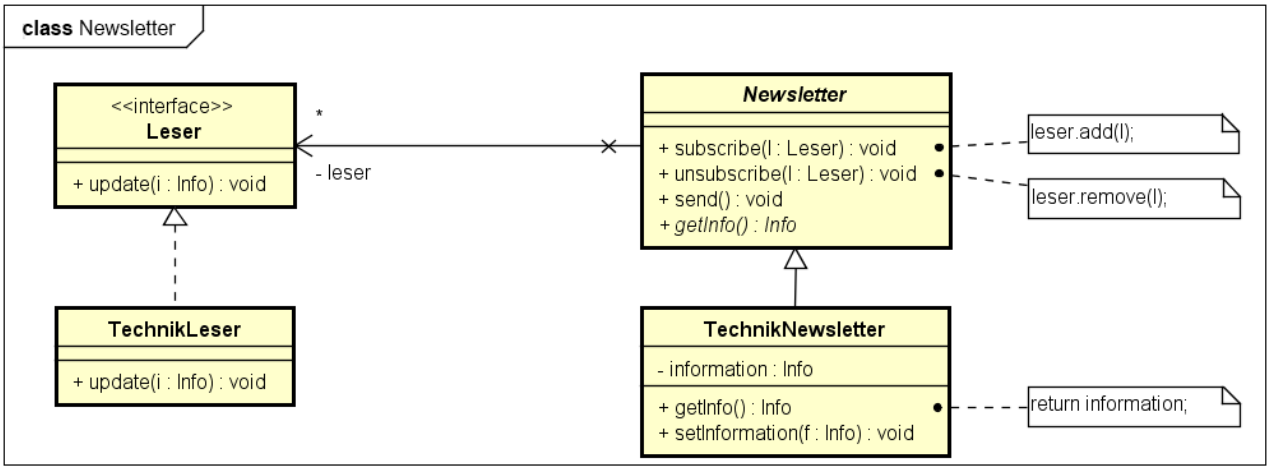
c) **(7 Punkte)** Zeichnen Sie das zugehörige Klassendiagramm mit **allen** Klassen, die Sie modellieren würden.



Bewertung:

- (1x 0,5P) Rahmen und Beschriftung
- (9x 0,5P) Klassen
- (1x 1P) 8 Vererbungspfeile
- (1x1P) Aggregation mit Multiplizität und Feld

d) **(2 Punkte)** Sie sehen hier die Anwendung eines Push-Observer-Patterns für Newsletter:



Das Attribut „leser“ des Newsletters sei als ArrayList implementiert. Die Implementierung der Methoden „getInfo“, „subscribe“ und „unsubscribe“ ist ebenfalls bereits vorgegeben.
Ergänzen Sie nun den Inhalt der „send“-Methode von Newsletter und der „setInformation“-Methode von TechnikNewsletter als korrekten Java-Code. Sie brauchen nur das Nötigste aufzuschreiben. Zusätzliche Checks sind hier nicht gefordert.

```
public void send() {
        for(Leser l : leser) {                  // 0,5 P
                l.update(this.getInfo());       // 0,5 P
        }
}

public void setInformation(Info f) {
        information = f;                        // 0,5 P
        send();                                 // 0,5 P
}
```

e) **(4,5 Punkte)** Gehen Sie davon aus, dass die Objekte „susi“ (eine Instanz von TechnikLeser) und „tech“ (eine Instanz von TechnikNewsletter) bereits instanziiert sind und stellen Sie sich folgenden Ablauf vor:

- „susi“ abonniert „tech“.
- „tech“ versendet neue News.

- „susi“ beendet ihr Abonnement bei „tech“.

Sehen Sie sich nun die in der nachfolgenden Tabelle aufgelisteten Methodenaufrufe an und

- markieren Sie Aufrufe die unmöglich sind mit „f“
- markieren Sie Aufrufe die für die korrekte Ausführung des oben beschriebenen Ablauf *nicht* erforderlich sind mit „—“

Ordnen Sie die verbleibenden Aufrufe in der Reihenfolge, in der sie aufgerufen werden müssen, damit der obige Ablauf wie erwartet funktioniert und tragen Sie zu jedem Aufruf seine Nummer in der korrekten Reihenfolge ein.

Nr.	Aufgerufene Methode
—	tech.setInformation(null);
1	tech.subscribe(susi);
5	tech.unsubscribe(susi);
f	Newsletter.send();
2	tech.send();
4	susi.update(information);
f	tech.update(information);
f	TechnikLeser.update(susi);
3	tech.getInfo();