Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Deni Raco, M. Sc.

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 1

Abgabe: 18.11.2021 16:30 Uhr

Organisatorisches

Die Übungsaufgaben müssen in Gruppen von drei bis vier Personen abgegeben werden. Die Abgabe ist über den RWTHmoodle Lernraum der Vorlesung einzureichen. Alle Gruppenmitglieder müssen auf der Abgabe vermerkt sein, inkl. Matrikelnummer. Die Rückgabe der Ergebnisse erfolgt über den RWTHmoodle Lernraum.

Aufgabe 1.1

Für diese Aufgabe werden **4 Punkte** vergeben.

Gegeben sind folgende Tätigkeiten eines Softwareentwicklungsprozesses:

- Benutzer der Software schulen
- Qualitätssicherung des Pflichtenheftes durchführen
- Gesetzliche Rahmenbedingungen prüfen
- Konzept und Prototyp einer Benutzeroberfläche erstellen
- Entwicklerteam zusammenstellen
- Code eines Programmmoduls debuggen
- Zwei Subsysteme verbinden und testen
- Termine und Kosten des Projektes planen
- Datenstrukturen festlegen
- Vorhandene Altlasten des Kunden analysieren
- Schnittstellen von Programmmodulen definieren
- Leistung der Entwickler bewerten und belohnen
- Software an neue Umgebung anpassen
- Kunden eine Rechnung stellen
- Test-Eingabedaten für ein Programmmodul ermitteln
- Strukturmodell des gesamten Softwaresystems entwerfen
- Dokumentation des Projektablaufes bewerten und archivieren
- Nach bereits vorhandenen, wiederverwendbaren Software-Bibliotheken suchen
- Performance-Prognose des Softwaresystems erstellen

- Programmcode kommentieren

a) Ordnen Sie die Tätigkeiten den fünf Grundaktivitäten des Wasserfallmodells zu. Eine Tätigkeit kann auch mehreren oder keiner Aktivität zugeordnet werden. Begründen Sie kurz Ihre Entscheidung.

b) Ordnen Sie die Tätigkeiten den acht Grundaktivitäten des V-Modells zu. Eine Tätigkeit kann auch mehreren oder keiner Aktivität zugeordnet werden. Begründen Sie kurz Ihre Entscheidung.

c) Warum ist eine starre Zuordnung in beiden Modellen problematisch?

Aufgabe 1.2

Für diese Aufgabe werden **2 Punkte** vergeben.

Nennen Sie die drei Rollen, vier Aktivitäten und drei Artefakte im SCRUM Entwicklungsprozess. Ordnen Sie die Rollen den Aktivitäten und Artefakten zu. Einer Rolle können mehrere Aktivitäten und Artefakte zugeordnet sein.

Aufgabe 1.3

Für diese Aufgabe werden **4 Punkte** vergeben.

Schlüpfen Sie in die Rolle des Softwareingenieurs der Firma *Carmpere*, die sich mit der Entwicklung von Elektroautos befasst. Ihre Unternehmensführung möchte ein deutschlandweites Ladenetz etablieren.

Identifizieren Sie auf Grundlage der folgenden Beschreibung mindestens vier funktionale Anforderungen und mindestens vier nicht-funktionale Anforderungen.

Die Firma Carmpere konzeptioniert die verschiedenen Use Cases zum Laden der Batterien, der von ihnen produzierten Autos mit Elektro- und Hybridmotoren. Es gibt Ladestationen und Schnellladestationen. Ladestationen sind über ein komfortables Interface zu bedienen. Die Schnellladestationen sollen an stark frequentierten Tankstellen auf Autobahnen eingesetzt werden. Das Ladenetz soll eine Ausfallsquote von unter 1% haben. Jede Schnellladestation soll alle Funktionalitäten bereitstellen können, die auch von einer Ladestation bereitgestellt werden. Fahrer interagieren mit Ladestationen. Ladestationen können zusätzlich auch mit Autos interagieren. Ladestationen sollen überprüfen können, ob ein Auto ordnungsgemäß mit der Ladestation über ein Ladekabel verbunden ist. Die Überprüfung erfolgt durch die Übertragung von Fahrzeugdaten. Weiterhin sollen Ladestationen dazu in der Lage sein, die benötigte Lademenge für ein Auto zu berechnen. Ein Fahrer soll einen Ladewunsch an einer Ladestation äußern können. Das Äußern des Ladewunschs hat zur Folge, dass die Ladestation die Lademenge für das Auto berechnet. Wenn die Lademenge berechnet wird, soll auch überprüft werden, ob das Auto ordnungsgemäß mit der Ladestation verbunden ist. Ein Auto soll von einer Ladestation geladen werden können. Schnellladestationen können Autos zusätzlich schnellladen. Das Schnellladen eines Autos hat zur Folge, dass die Batterie des Autos schneller geladen wird als beim herkömmlichen Ladevorgang. Falls bei einer Ladestation ein Lade-vorgang gestartet wird

und die Ladestation eine Schnellladestation ist, dann kann auch der Schnellladevorgang ausgeführt werden. Fahrer können an Ladestationen nicht mit Bargeld bezahlen. Es kann entweder per Carmpere App oder per Kreditkarte bezahlt werden. Der Bezahlvorgang ist verschlüsselt.

Benutzer der Software schulen = Wartung – wenn alles erledigt ist, kann man Benutzer schulen (eigentlich aber im Wasserfallmodell nicht vorgesehen)

Qualitätssicherung d. Pflichtenhefts = Analyse – Rahmenbedingungen müssen bereits zu Beginn berücksichtigt werden

Gesetzliche Rahmenbedingungen prüfen = Analyse – Rahmenbedingungen stellen die Basis im Softwareentwicklungsprozess dar

Konzept + Prototyp einer Benutzeroberfläche erstellen = Entwurf, da hier zum ersten Mal etwas „gebastelt" wird

Entwicklerteam zusammenstellen = Analyse, da sonst Projekt schwer umsetzbar

Code eines Programmmoduls debuggen = Implementierung oder Test + Integration, hängt davon ab, wann ein Fehler bemerkt wurde (erst beim Testen oder schon vorher)

2 Subsysteme verbinden + testen = Test + Integration, da die Subsysteme in der Implementierungsphase programmiert werden und das Zusammenführen auf die Integration entfällt

Termine + Kosten d. Projekts planen = Analyse – Kosten sollten prinzipiell am Anfang abgeschätzt werden

Datenstrukturen festlegen = Entwurf, hier wird genauer entschieden, wie später implementiert wird

Vorhandene Altlasten d. Kunden analysieren = Analyse, damit möglichst wenig später angepasst werden muss

Schnittstellen von Programmodulen definieren = Entwurf, hier wird die spätere Architektur bestimmt

Leistung der Entwickler bewerten + belohnen = keiner Aktivität zuordenbar, da dies im Wasserfallmodell nicht vorgesehen ist

Software an neue Umgebung anpassen = Wartung, da der Code geprüft ist und das macht, was er soll, sich aber die Rahmenbedingungen (Umgebung) geändert haben und somit der Code angepasst werden muss. (eventuell auch Test + Integration)

Kunden eine Rechnung stellen = keiner Aktivität zuordenbar, das Wasserfallmodell beschäftigt sich kaum mit dem Kunden (abgesehen von der Analyse + Änderungswünschen d. Kunden)

Test - Eingabedaten für ein Programmodul ermitteln = Test + Integration, da dieser Prozess Teil des Testens ist. Eventuell auch Entwurf, je nach Priorisierung

Strukturmodell ges. Softwaresystem entwerfen: Analyse, da hier das Produkt (die Software) definiert wird

Dokumentation d. Projektverlaufs bewerten + archivieren: keiner Aktivität zuordenbar, da nicht vorgesehen

Nach bereits vorhandenen, wvbaren Softw.-Bibl. suchen: Entwurf, Vorbereitung d. Implementierung

Performance-Prognose d. Softwaresystems erstellen: keiner Aktivität zuordenbar (eventuell Test + Integration, wenn Testergebnisse vorliegen)

Programmcode kommentieren: Implementierung, da Kommentare zu einem guten Programmierstil (also zum Coden selbst) dazugehören

b) Ordnen Sie die Tätigkeiten den acht Grundaktivitäten des V-Modells zu. Eine Tätigkeit kann auch mehreren oder keiner Aktivität zugeordnet werden. Begründen Sie kurz Ihre Entscheidung.

Benutzer der Software schulen = Abnahmetest oder danach (Programm muss existieren)

Qualitätssicherung d. Pflichtenhefts = Analyse, Grob-, Feinentwurf, Implementierung; definierte Testfälle anschauen (ggf. auch Integrationstest)

Gesetzliche Rahmenbedingungen prüfen = Analyse – Rahmenbedingungen stellen die Basis im Softwareentwicklungsprozess dar

Konzept + Prototyp einer Benutzeroberfläche erstellen = Grobentwurf, Systemtest – Konzept wird nach 1. Entwurf erarbeitet +Prototyp

Entwicklerteam zusammenstellen = Analyse, ohne Entwickler restlicher Prozess schwer realisierbar

Code eines Programmmoduls debuggen = Modultest (ggf. auch erst später, hängt davon ab, wann Fehler entdeckt wird)

2 Subsysteme verbinden + testen = Integrationstest, da die Subsysteme in der Implementierungsphase programmiert werden und das Zusammenführen auf die Integration entfällt
auch Feinentwurf, um verbinden zu planen

Termine + Kosten d. Projekts planen = Analyse – Kosten sollten prinzipiell am Anfang abgeschätzt werden

Datenstrukturen festlegen = Feinentwurf, hier macht man sich genauere Gedanken, wie man im Anschluss implementieren möchte

Vorhandene Altlasten d. Kunden analysieren = Analyse, damit möglichst wenig später angepasst werden muss

Schnittstellen von Programmodulen definieren = Grobentwurf, da äußerst relevant für Feinentwurf

Leistung der Entwickler bewerten + belohnen = keiner Aktivität zuordenbar, da dies im V-Modell nicht vorgesehen ist (nach Abnahmetest denkbar)

Software an neue Umgebung anpassen = Systemtest, da sich dieses geändert hat und man zurück zur Implementierung springen muss (gegebenenfalls)

Kunden eine Rechnung stellen = keiner Aktivität zuordenbar, auch das V-modell beschäftigt sich kaum mit dem Kunden (abgesehen von der Analyse)

Test – Eingabedaten für ein Programmmodul ermitteln = Modultest, hat unmittelbar mit einem Programmmodul zu tun (bei früherer Planung: Grob- oder Feinentwurf)

Strukturmodell ges. Softwaresystem entwerfen: Grobentwurf, da es um einen Überblick geht

Dokumentation d. Projektverlaufs bewerten + archivieren: keiner Aktivität zuordenbar, da Dokumentation durch Tests gegeben

Nach bereits vorhandenen, wvbaren Softw.-Bibl. suchen: Implementierung, hier wird der Code geschrieben (eventuell auch Feinentwurf)

Performance-Prognose d. Softwaresystems erstellen: keiner Aktivität zuordenbar (eventuell Systemtest)

Programmcode kommentieren: Implementierung, da Kommentare zu einem guten Programmierstil (also zum Coden selbst) dazugehören

Aufgabe 1.2

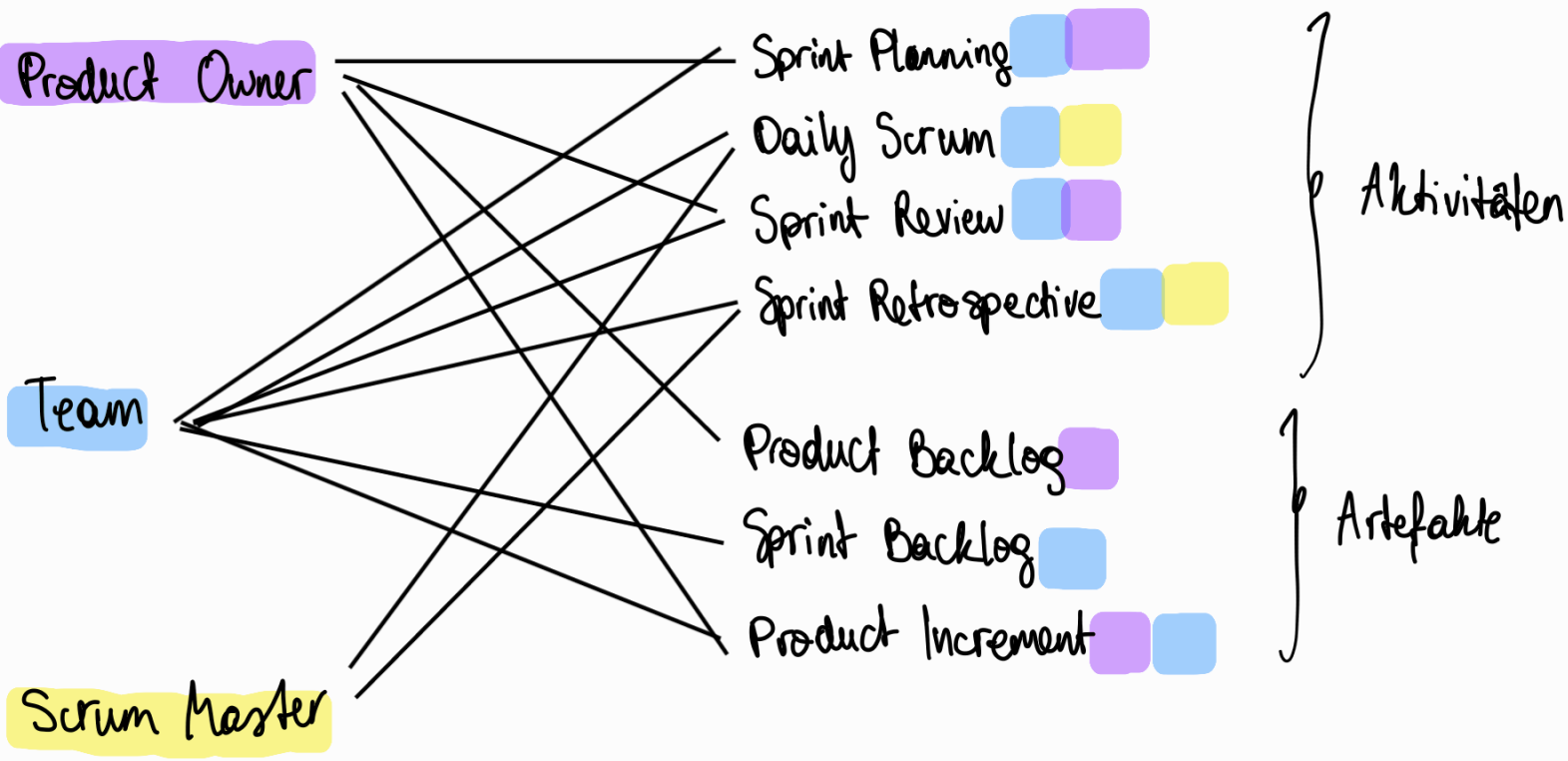
Für diese Aufgabe werden **2 Punkte** vergeben.

Nennen Sie die drei Rollen, vier Aktivitäten und drei Artefakte im SCRUM Entwicklungsprozess. Ordnen Sie die Rollen den Aktivitäten und Artefakten zu. Einer Rolle können mehrere Aktivitäten und Artefakte zugeordnet sein.

3 Rollen:
1. Product Owner = Anwender/Stakeholder
2. Team = Entwickler
3. Scrum Master = Moderator

4 Aktivitäten:
1. Sprint Planning
2. Daily Scrum
3. Sprint Review
4. Sprint Retrospective

3 Artefakte:
1. Product Backlog = To-Do Liste der Requirements
2. Sprint Backlog = Prognose wie funktionstüchtig nächstes Inkrement sein wird / welche Arbeit noch nötig sein wird
3. Product Increment = funktionsfähiges Zwischenprodukt



Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Deni Raco, M. Sc.

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 2

Abgabe: 23.11.2021 10:30 Uhr

Organisatorisches

Die Übungsaufgaben müssen in Gruppen von drei bis vier Personen abgegeben werden. Die Abgabe ist über den RWTHmoodle Lernraum der Vorlesung einzureichen. Alle Gruppenmitglieder müssen auf der Abgabe vermerkt sein, inkl. Matrikelnummer. Die Rückgabe der Ergebnisse erfolgt über den RWTHmoodle Lernraum.

Aufgabe 2.1

Für diese Aufgabe werden **5 Punkte** vergeben.

Formalisieren Sie die folgende textuelle Beschreibung der Kommunikation während des Ladevorgangs zwischen einem Elektroauto und einer Ladesäule des Autoherstellers Prominux (Konkurrent von Carmpere), indem Sie ein geeignetes **Aktivitätsdiagramm** erstellen.

Als Erstes prüft das Elektroauto, ob der Ladestecker gesteckt ist. Steckt der Ladestecker nicht wird der Ladevorgang beendet. Nur bei gestecktem Stecker berechnet das Auto als Nächstes die zu ladende Menge Strom. Dieser Ladewunsch wird als Nächstes an die Ladesäule gesendet. Die Ladesäule prüft den Ladewunsch. Danach bereitet sie den Ladevorgang entsprechend vor und sendet gleichzeitig eine Anfrage nach der Zahlungsmethode an das Elektroauto. Daraufhin bestimmt das Elektroauto die Zahlungsmethode und sendet diese Information an die Ladesäule zurück. Erst nachdem die Zahlungsmethode empfangen wurde und der Ladevorgang vorbereitet ist, prüft die Ladesäule ein letztes Mal, ob der Ladevorgang gestartet werden kann. Nur wenn es bei dieser Prüfung zu keinem Fehler gekommen ist, stellt sie die angeforderte Menge Strom bereit. Ist dies der Fall, beginnt das Elektroauto zu laden. Ist es allerdings zu einem Fehler gekommen, meldet die Ladesäule einen Fehler.

Tipp: Achten Sie darauf, dass Kontrollknoten in Ihrem Aktivitätsdiagramm paarweise auftreten müssen und beachten Sie alle möglichen Akteure, die in der Beschreibung erwähnt werden.

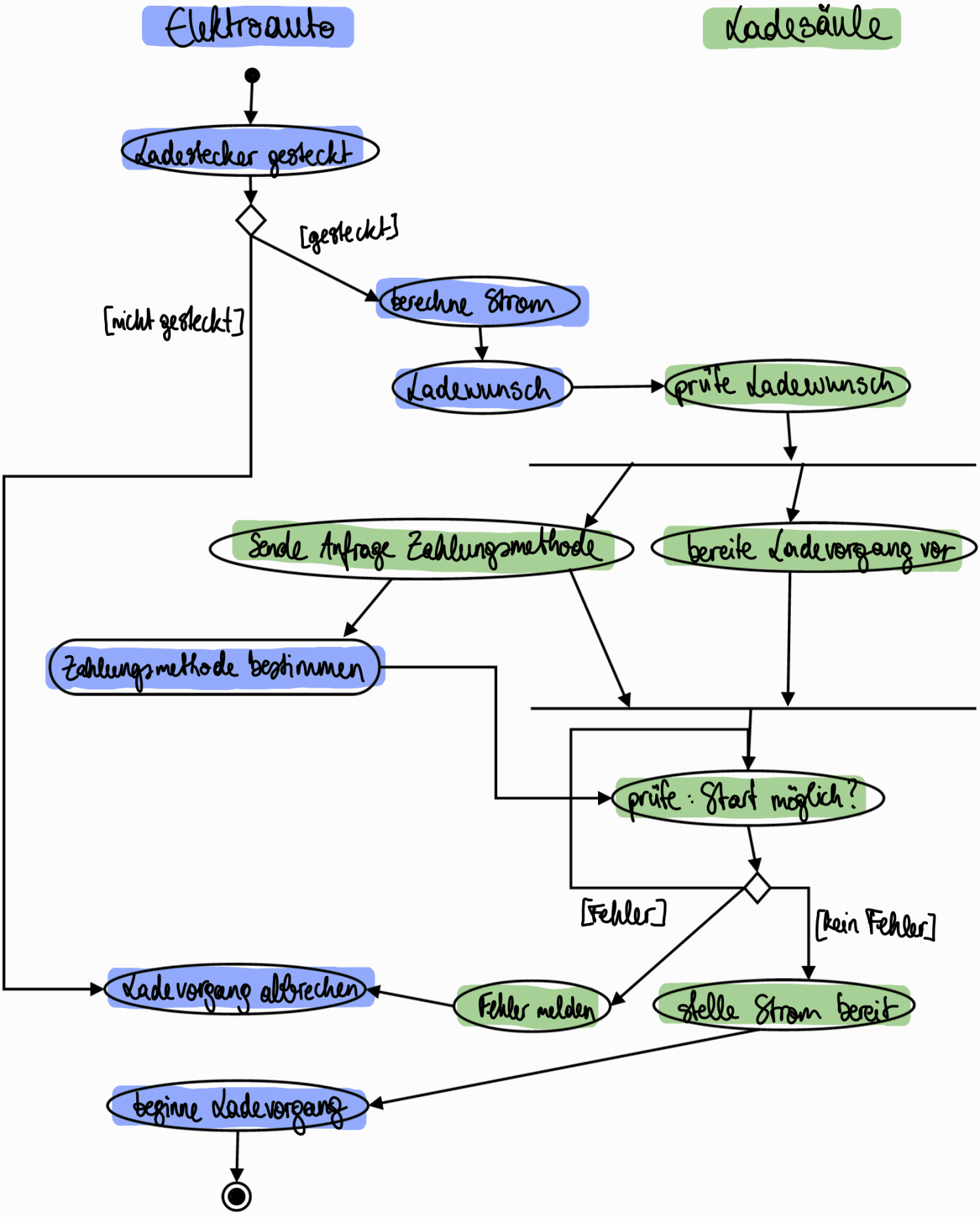
Aufgabe 2.2

Für diese Aufgabe werden **5 Punkte** vergeben.

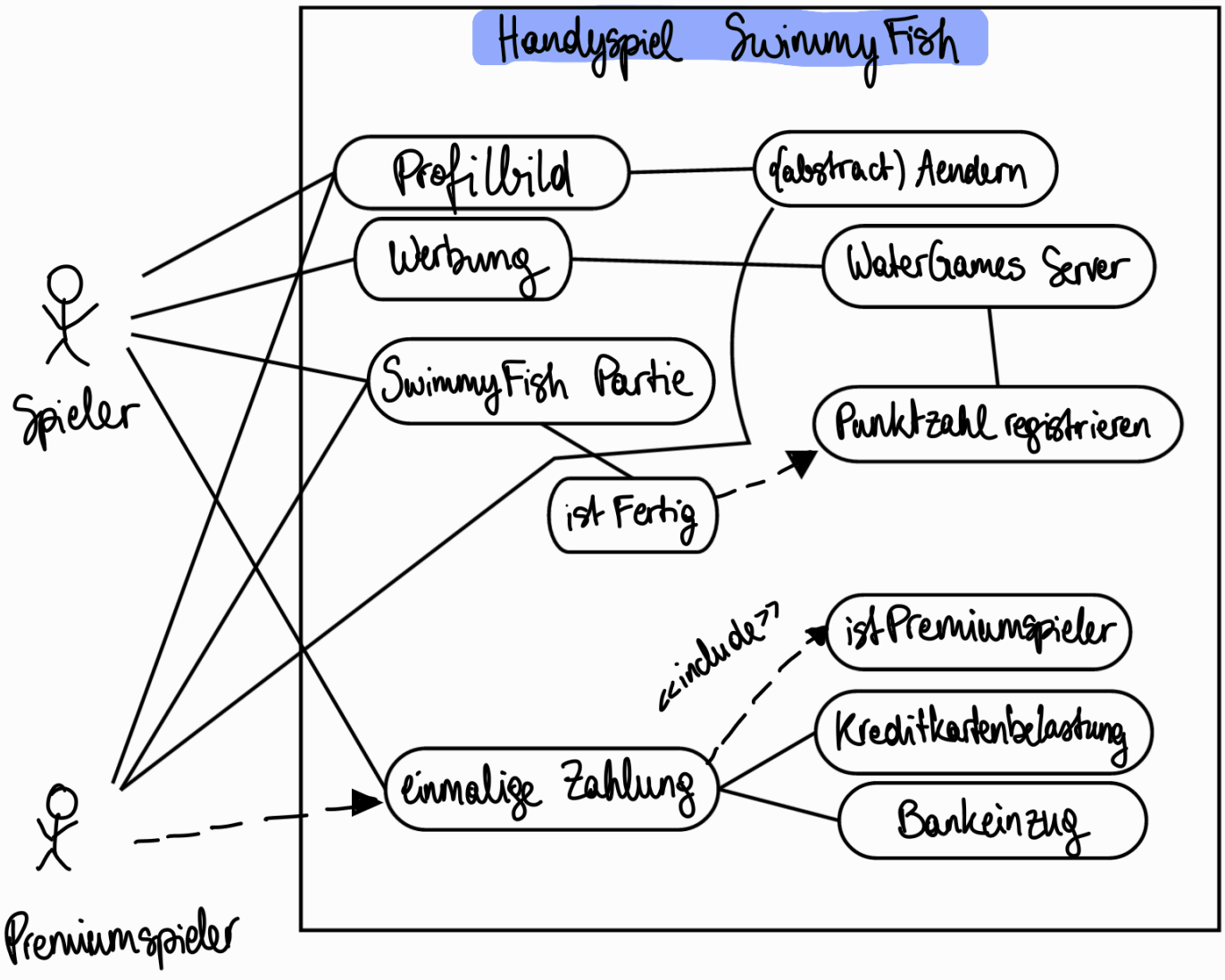
Die Firma WaterGames entwirft das Handyspiel SwimmyFish. Die Nutzer des Spiels können Spieler oder Premiumspieler sein. Spieler sollen zu Premiumspielern werden können, indem sie eine einmalige Zahlung tätigen. Der Preis, um Premiumspieler zu werden, soll nicht zu teuer sein. Die Zahlung soll entweder über Kreditkartenbelastung oder per Bankeinzug erfolgen können. Intern wurde in der Firma diskutiert, ob auch eine Zahlung per Rechnung in Frage kommt. Die Firma hat sich aber dagegen entschieden. Unabhängig von der Bezahlungsart soll beim Bezahlvorgang geprüft werden, ob der Spieler schon ein Premiumspieler ist. Alle Spieler haben ein Profilbild. Während alle Spieler, die keine Premiumspieler sind, das gleiche Profilbild haben, sollen Premiumspieler ihre Profilbilder ändern können. Alle Nutzer sollen SwimmyFish spielen können. Das Spielen von SwimmyFish soll Spaß bereiten. Allen Spielern außer Premiumspielern soll während des Spielens von SwimmyFish Werbung angezeigt werden. Die Werbung soll nicht uninteressant sein. Die zu zeigende Werbung soll vom WaterGames Server angefragt werden. Wenn ein Spieler eine Partie SwimmyFish beendet hat, soll die Punktzahl beim WaterGames Server registriert werden.

Erstellen Sie ein **Use Case Diagramm**, das die für das Softwaresystem des Spiels SwimmyFish relevanten Aspekte darstellt.

Aufgabe 2.1: Aktivitätsdiagramm



Aufgabe 2.2: Use Case Diagramm



SE SOFTWARE ENGINEERING | RWTH AACHEN UNIVERSITY

http://www.se-rwth.de/

Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Dipl.-Inform. Deni Raco

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 3

Abgabe: 30.11.2021 10:30 Uhr

Organisatorisches

Die Übungsaufgaben müssen in Gruppen von drei bis vier Personen abgegeben werden. Die Abgabe ist über den RWTHmoodle Lernraum der Vorlesung einzureichen. Alle Gruppenmitglieder müssen auf der Abgabe vermerkt sein, inkl. Matrikelnummer. Die Rückgabe der Ergebnisse erfolgt über den RWTHmoodle Lernraum.

Aufgabe 3.1 (10 Punkte)

Der Automobilhersteller Carmpere hat seine Spezifikation des Hochvoltspeichers aktualisiert. Modellieren Sie die folgende Spezifikation des Hochfahr- und Herunterfahrprozesses des Speichers in einem passenden Zustandsdiagramm wie in der Vorlesung vorgestellt.

Zu Beginn ist der Speicher ausgeschaltet. Bevor der Speicher angeschaltet ist, muss er hochgefahren werden. Tritt während des Hochfahrens ein Fehler auf, wird eine Fehlermeldung über das Bordnetz gesendet und der Speicher bleibt ausgeschaltet. Wird das Hochfahren erfolgreich beendet, ist der Speicher angeschaltet.

Ein angeschalteter Speicher ist nach dem Hochfahren immer entladen. Wird ein Ladewunsch geäußert, so wird der Ladevorgang gestartet. Nachdem der Ladevorgang beendet wurde ist der Speicher aufgeladen. Ist der Speicher aufgeladen, wird er komplett entladen sobald ein Entladewunsch geäußert wird.

Soll ein angeschalteter Hochvoltspeicher ausgeschaltet werden, so muss er zunächst heruntergefahren werden. Nur ein entladener Speicher kann heruntergefahren wer-den. Tritt beim Herunterfahren ein Fehler auf, wird eine Fehlermeldung ausgegeben und der Speicher geht zurück in den angeschalteten Zustand. Nachdem das Herunterfahren erfolgreich beendet wurde, ist der Speicher ausgeschaltet.

Hinweis: Nutzen Sie einen Oberzustand an angemessener Stelle.

Aufgabe 3.2 (10 Punkte)

Sie befinden sich im Entwurf eines Campusverwaltungssystems zur Verwaltung von Personen, Veranstaltungen und Räumen. Dabei machen Sie sich genauere Gedanken über die Datenstruktur des Systems.

Teilaufgabe a)

Formalisieren Sie die folgende textuelle Beschreibung des Systems, indem Sie ein geeignetes Klassendiagramm erstellen.

Das System besteht aus einer Menge an Personen und Veranstaltungen, die es verwaltet. Jede Person hat einen Namen. Eine Person ist entweder ein Student oder ein Mitarbeiter. Studenten haben eine Matrikelnummer, Mitarbeiter eine ID. Die vom System verwalteten Veranstaltungen sind Vorlesungen und Seminare. Es können aber noch andere Arten von Veranstaltungen verwaltet werden. Jede Veranstaltung hat einen Namen und einen Turnus, der entweder „SS“, „WS“ oder „Dual“ sein kann und das Semester beschreibt, in dem die Veranstaltung stattfindet.

Des Weiteren verwaltet das System Räume. Ein Raum wird von genau einem System verwaltet. Jeder Raum hat eine Nummer und eine Anzahl an Plätzen, die er bietet. Veranstaltungen finden in mindestens einem Raum statt. Eine Veranstaltung wird von mindestens einem Mitarbeiter betreut. Ein Mitarbeiter kann maximal drei Veranstaltungen betreuen. Studenten können beliebig viele Veranstaltungen besuchen.

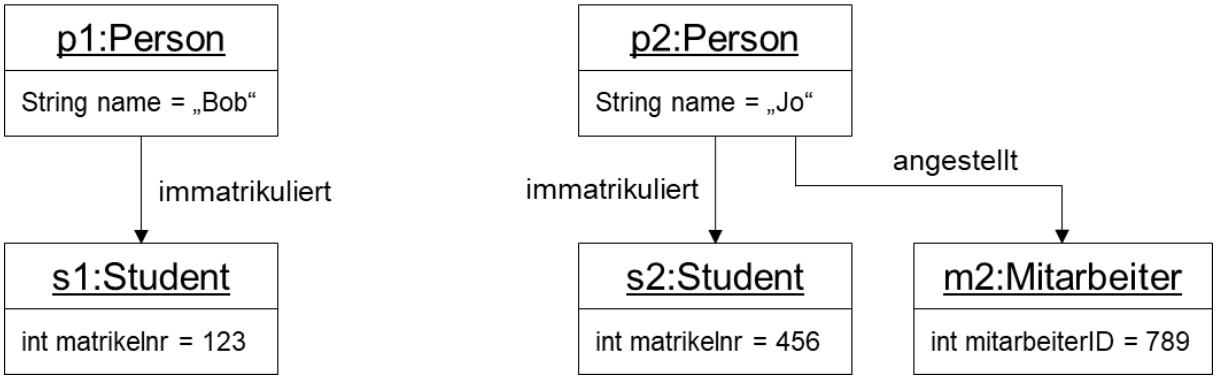
Teilaufgabe b)

Um die Terminplanung im System zu verbessern, soll es geändert werden. Dabei ist es wichtig, dass Veranstaltungen zu unterschiedlichen Terminen in unterschiedlichen Räumen stattfinden können.

1) Nennen Sie das Entwurfsmuster, das sich anbietet, um das System entsprechend zu verbessern. Begründen Sie zusätzlich kurz, wodurch dieses einen Vorteil für das entwickelte System bietet.
2) Modellieren Sie im Folgenden den Teil Ihres Klassendiagramms aus Teilaufgabe a) entsprechend des Musters um.

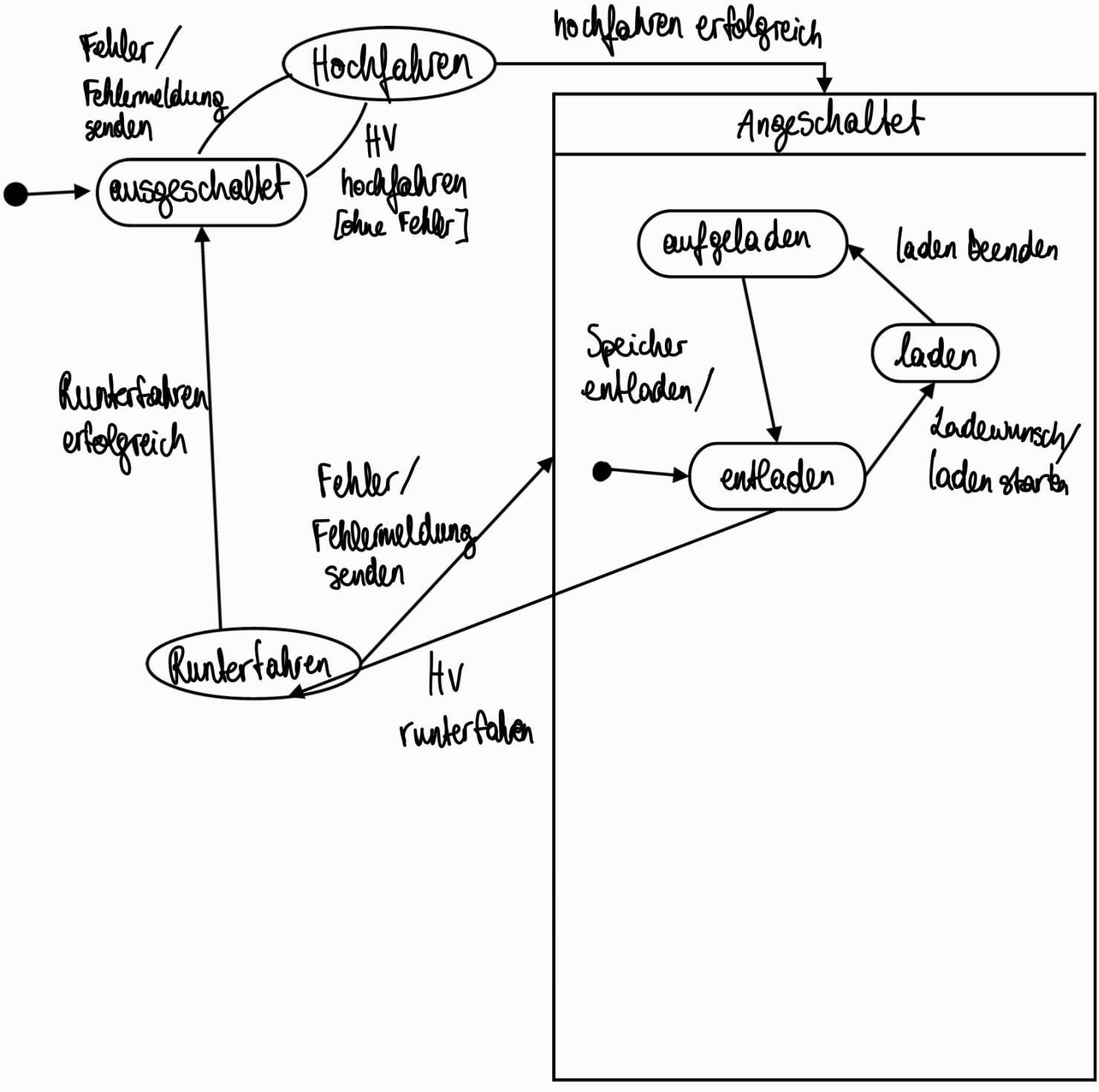
Teilaufgabe c)

Eine Ihrer Kolleginnen hat Ihr Klassendiagramm aufgrund einer Änderung in den Anforderungen modifiziert, so dass folgendes Objektdiagramm eine gültige Instanz des Klassendiagramms ist:



1) Welches Entwurfsmuster hat Ihre Kollegin umgesetzt?
2) Wie lautet die Anforderung, die Ihre Kollegin umgesetzt hat?

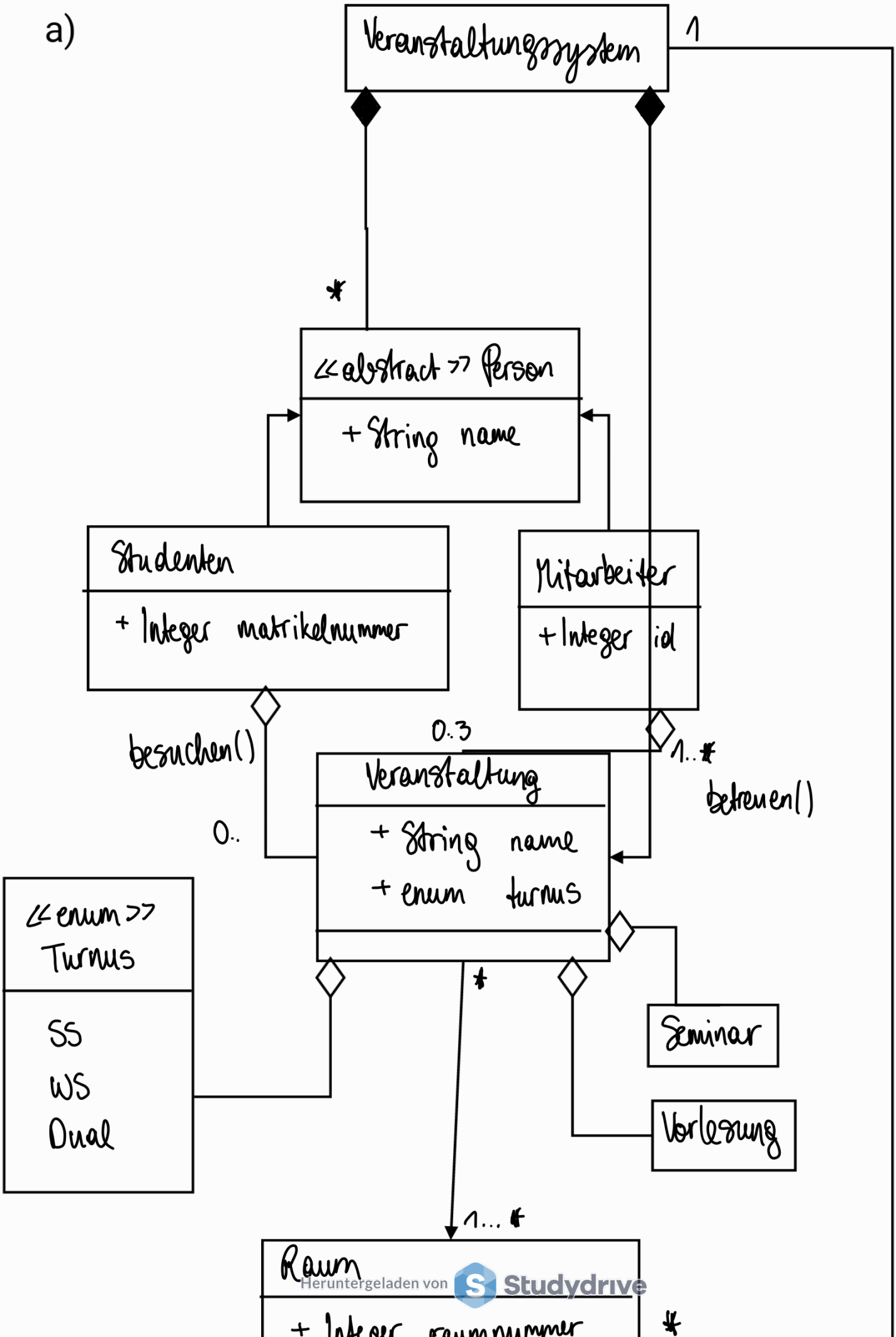
Aufgabe 3.1: Zustandsdiagramm



CD

Aufgabe 3.2

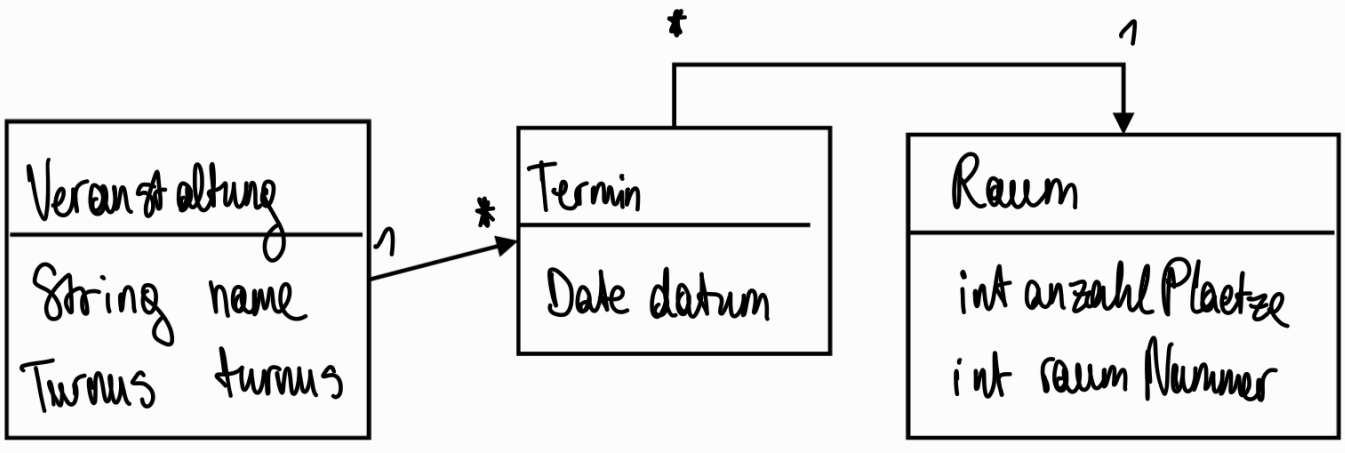
a)



+ Integer sitzplaetze

b) (i) Das Entwurfsmuster „Koordinator" kann das System entsprechend verbessern, weil dadurch Veranstaltung und Raum entkoppelt werden.

ii)



c) (i) Die Kollegin hat das Entwurfsmuster „Rolle" (nicht wechselnd) umgesetzt

(ii) Die Anforderung, die die Kollegin umgesetzt hat, ist, dass eine Person sowohl Student/in als auch Mitarbeiter/in sein kann

Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Dipl.-Inform. Deni Raco

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 4

Abgabe: 07.12.2021 10:30 Uhr

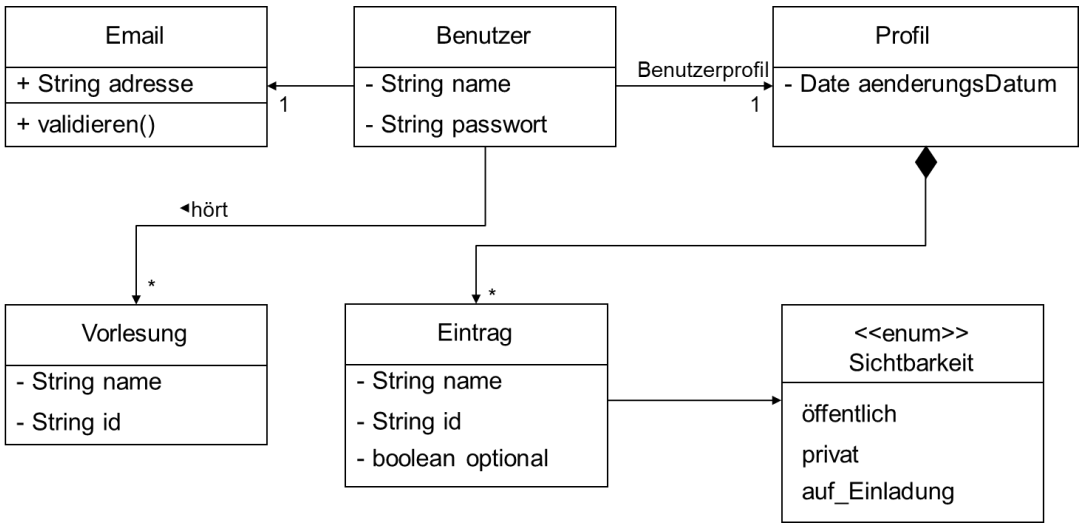
Organisatorisches

Die Übungsaufgaben müssen in Gruppen von drei bis vier Personen abgegeben werden. Die Abgabe ist über den RWTHmoodle Lernraum der Vorlesung einzureichen. Alle Gruppenmitglieder müssen auf der Abgabe vermerkt sein, inkl. Matrikelnummer. Doppelabgaben führen zu Nichtbewertung. Die Rückgabe der Ergebnisse erfolgt über den RWTHmoodle Lernraum.

Aufgabe 4.1 – Klassendiagramme im Entwurf

Diese Aufgabe ist **4 Punkte** wert.

Sie haben in der Vorlesung Verfeinerungen von Klassendiagrammen kennen gelernt, mit denen man grobe, unvollständige Klassendiagramme konkretisieren kann, um sie als Vorlagen für den Feinentwurf einsetzen zu können. Betrachten Sie nun das folgende Klassendiagramm aus der Analyse.



Sie wollen aus diesem Analysediagramm ein Entwurfsdiagramm ableiten. Führen Sie dafür die folgenden Verfeinerungen durch.

a) Ergänzen Sie das Klassendiagramm um folgende Sachverhalte auszudrücken. Wählen Sie jeweils ein geeignetes Analysemuster aus und begründen Sie kurz Ihre Wahl.

   Ein Benutzer kann als Hörer an einer Vorlesung teilnehmen. Hierbei werden das aktuelle Semester und der Studiengang des Benutzers gespeichert.

   Profileinträge können in Kategorien gruppiert werden (z.B. Kategorie „Kontaktdaten“, „Interessen“ etc.). Dabei kann eine Kategorie Unterkategorien beinhalten.

b) Anschließend sollten dedizierte Verwalterklassen eingepflegt werden. Ergänzen Sie das Diagramm um notwendige Verwalterklassen. Diese unterscheiden sich von fachlichen Klassen aus der Problemdomäne (Benutzer, Profil, …) dadurch, dass sie strukturelle und technische Aufgaben übernehmen. Folgendes soll im Anschluss in Ihrem Diagramm berücksichtigt sein.
   i. Erzeugen, Löschen, Laden, Speichern von Vorlesungen, Profilen und Benutzern mit individuellen Schnittstellen.
   ii. Möglichkeit alle Teilnehmer einer Vorlesung für ein bestimmtes Semester abzufragen.
   iii. Die Verwaltung von Benutzern und Profilen wird in derselben Klasse realisiert.

c) Qualifizieren Sie eine Assoziation auf sinnvolle Weise.

Aufgabe 4.2 – Entwurfsmuster

Für diese Aufgabe werden **4 Punkte** vergeben.

Folgende Beschreibung ist gegeben:

„Die API des aPhone bietet jeder Anwendung, die auf dem aPhone läuft, ein Objekt namens `NetworkStatus` *an, welches die Methode* `hasInternetConnection()` *anbietet. Alle Anwendungen können über diese Methode in Erfahrung bringen, ob gerade eine Verbindung zum Internet möglich ist, oder nicht. Diese Information ändert sich permanent.“*

Nennen Sie das Entwurfsmuster der Gang of Four, das für die Realisierung der Anforderung geeignet ist. Dabei sollen beliebig viele Anwendungen immer auf dem neuesten Stand dieser Information bleiben können. Zeichnen Sie im Anschluss ein Sequenzdiagramm, das alle zentralen Abläufe des Musters darstellt.

Hinweis: Zwei Objekte und vier Interaktionen reichen aus.

Aufgabe 4.3 – Sequenzdiagramme in der Analyse

Für diese Aufgabe werden **2 Punkte** vergeben.

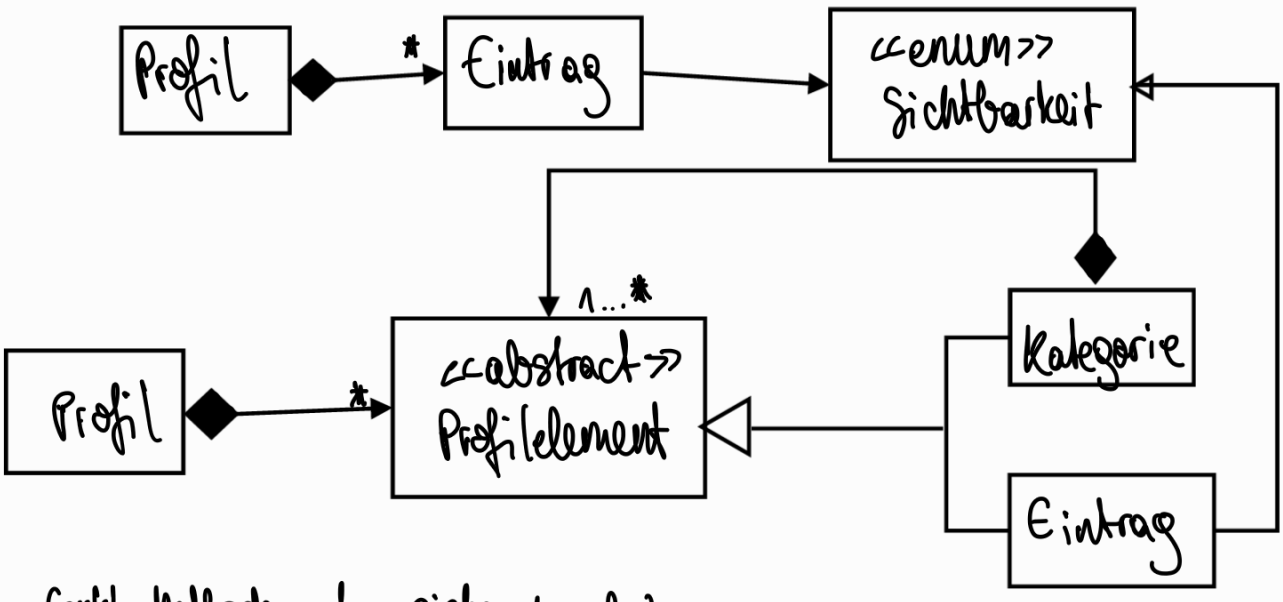
Modellieren Sie den folgenden Text als UML Sequenzdiagramm:

Ein Kunde geht zu einem Imbiss und bestellt bei einem Mitarbeiter einen Teller Pommes. Der Mitarbeiter fragt den Kunden, ob der Kunde Ketchup zu seinen Fritten möchte, was der Kun-

de bestätigt. Danach fragt der Mitarbeiter, ob noch etwas zu der Bestellung hinzukommt. Dies verneint der Kunde. Dann fängt der Mitarbeiter an, die Pommes vorzubereiten und nach einiger Zeit liefert er dem Kunden die Bestellung. Der Kunde isst die Pommes direkt im Imbiss.

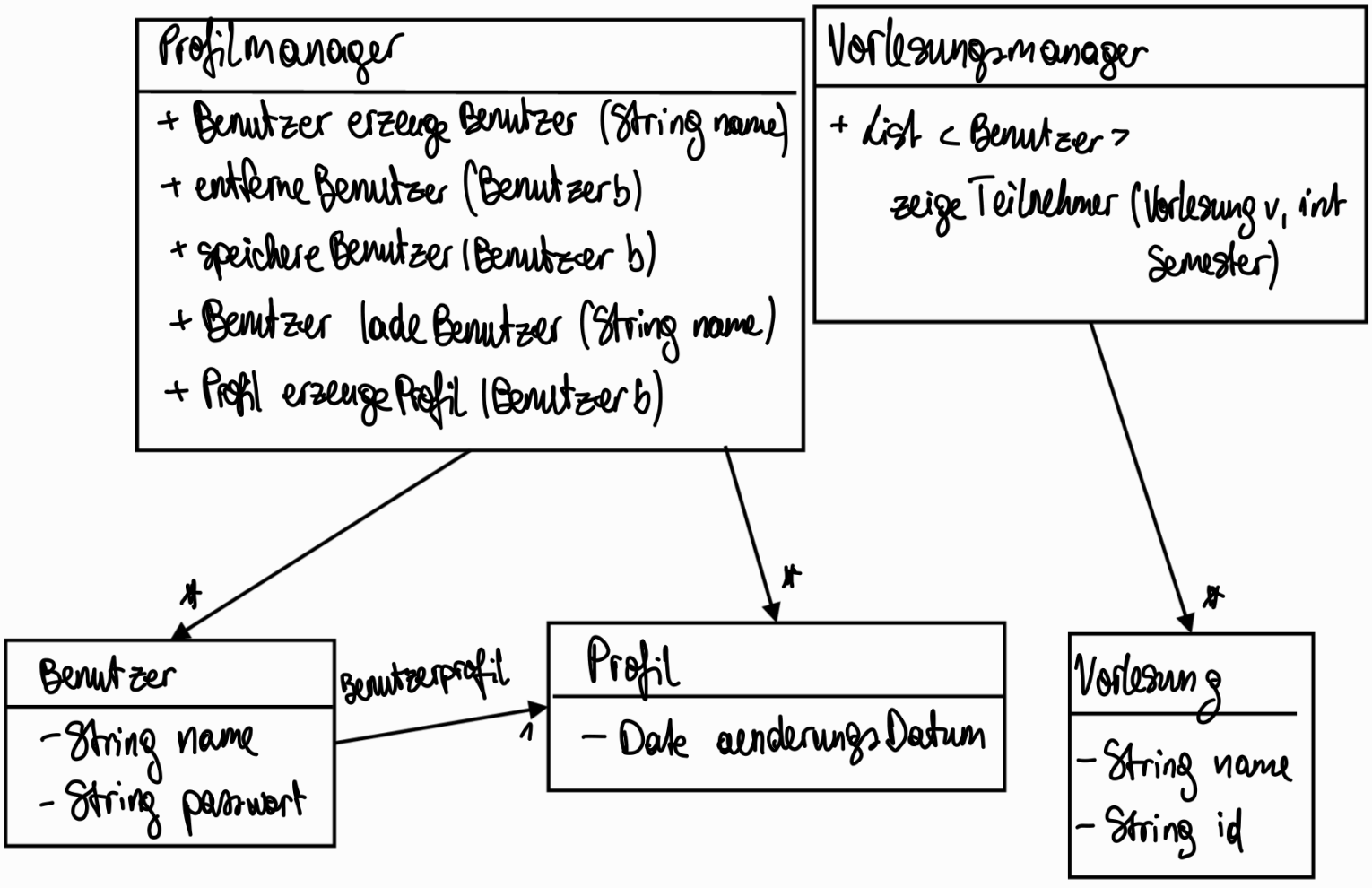
Aufgabe 4.1: Klassendiagramm im Entwurf

a) Kompositum: Profileinträge in Kategorien gruppieren

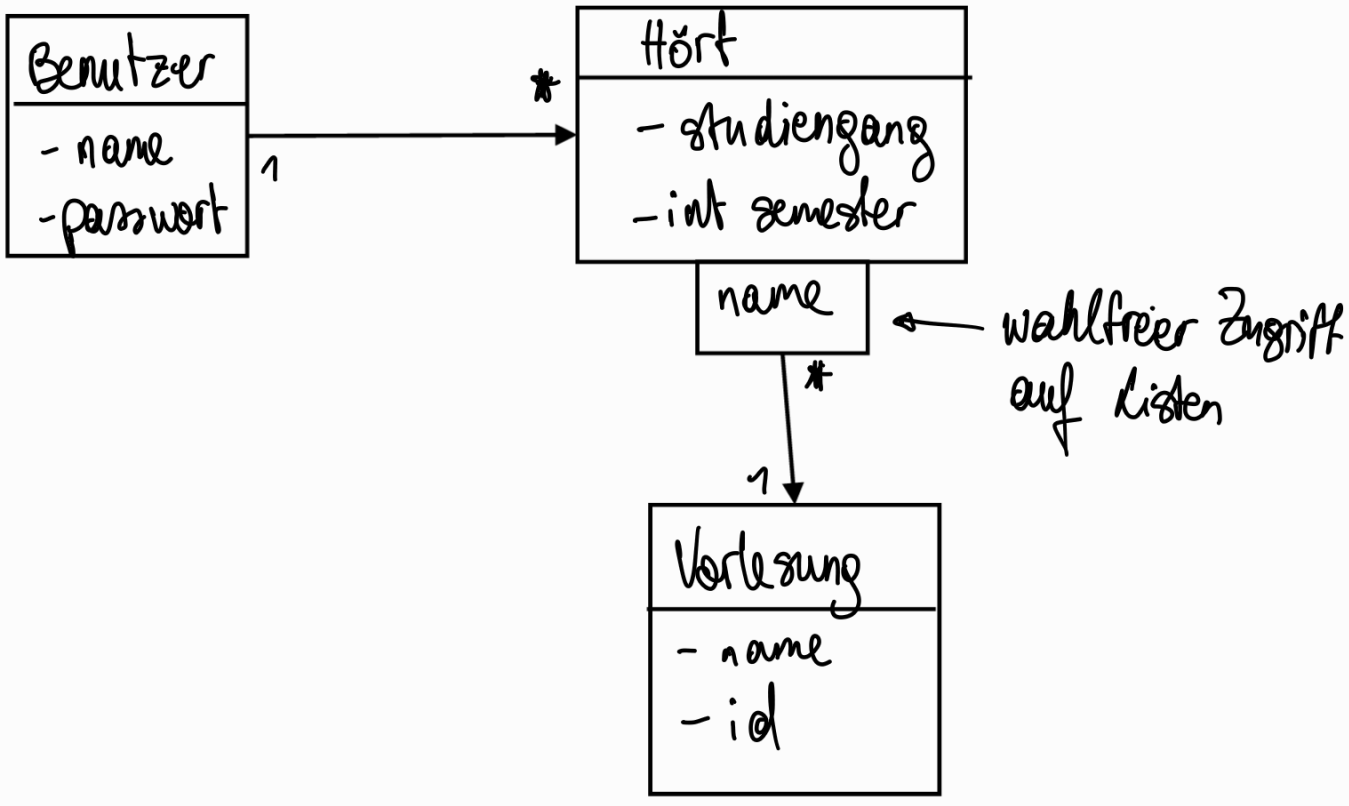


(exkl. Methoden etc., siehe Angabe)

b) Manager/Verwaltungsklasse



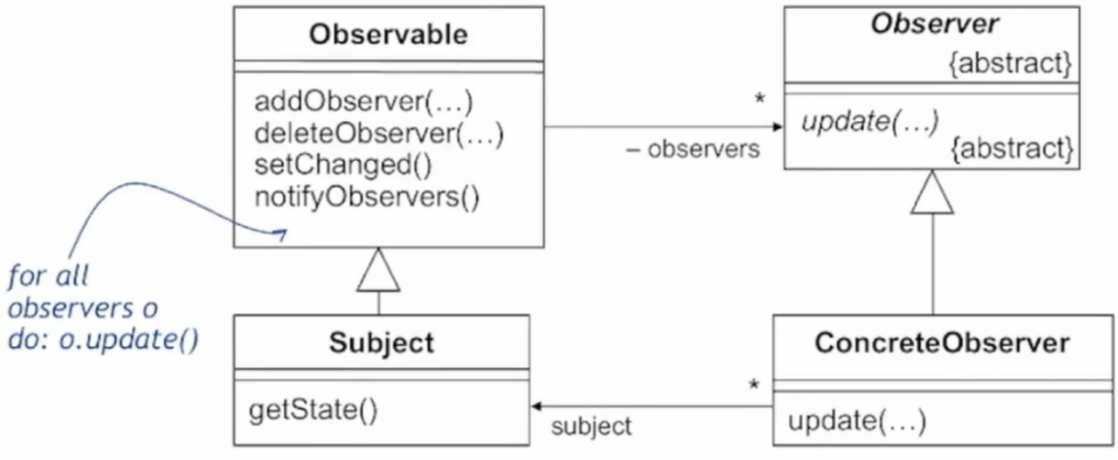
c) Assoziation qualifizieren



Aufgabe 4.2: Entwurfsmuster

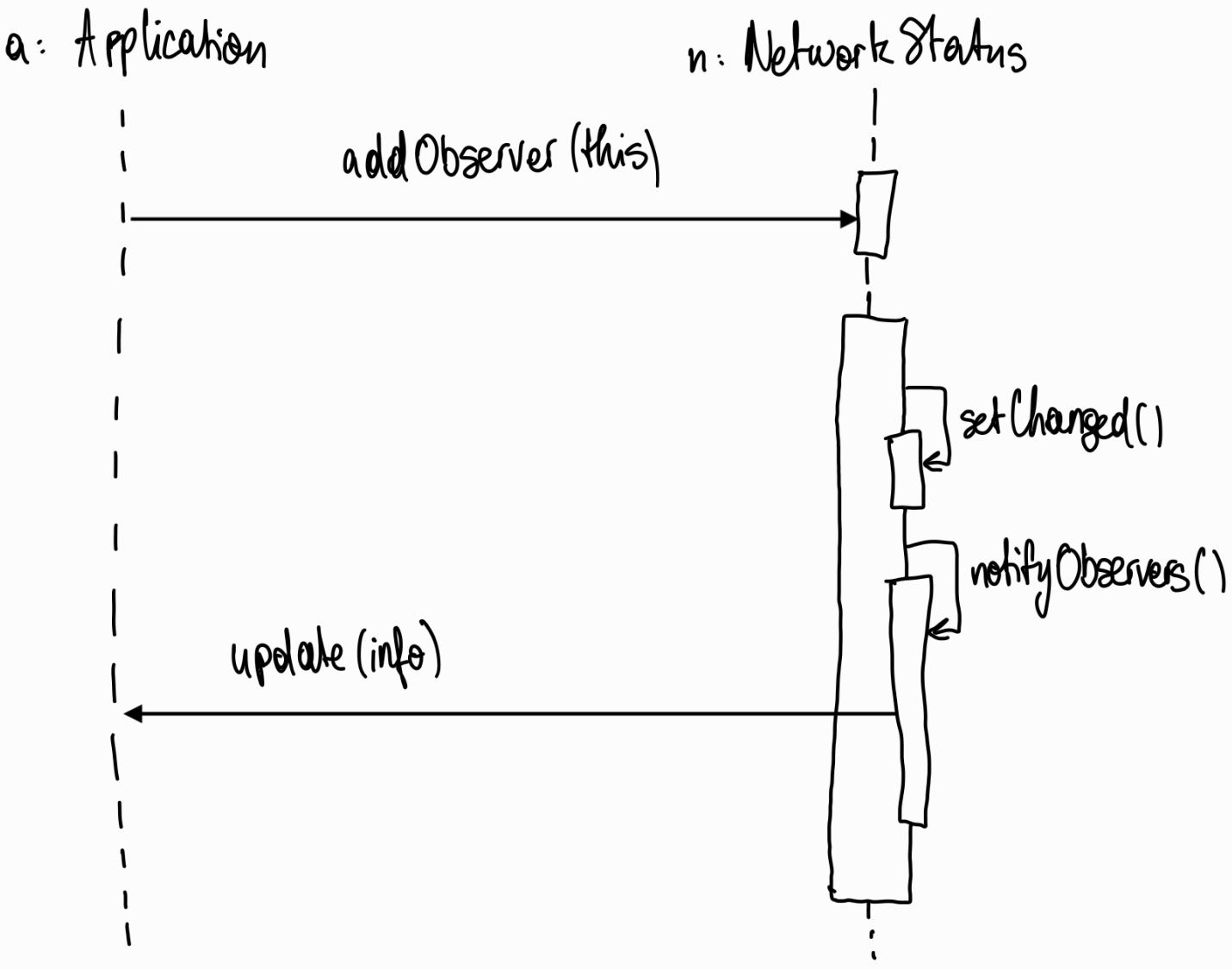
=> Observer Muster

- Name: **Observer** (dt.: Beobachter)
- *Problem*:
  - Mehrere Objekte sind interessiert an bestimmten Zustandsänderungen eines Objektes
- *Lösung*:

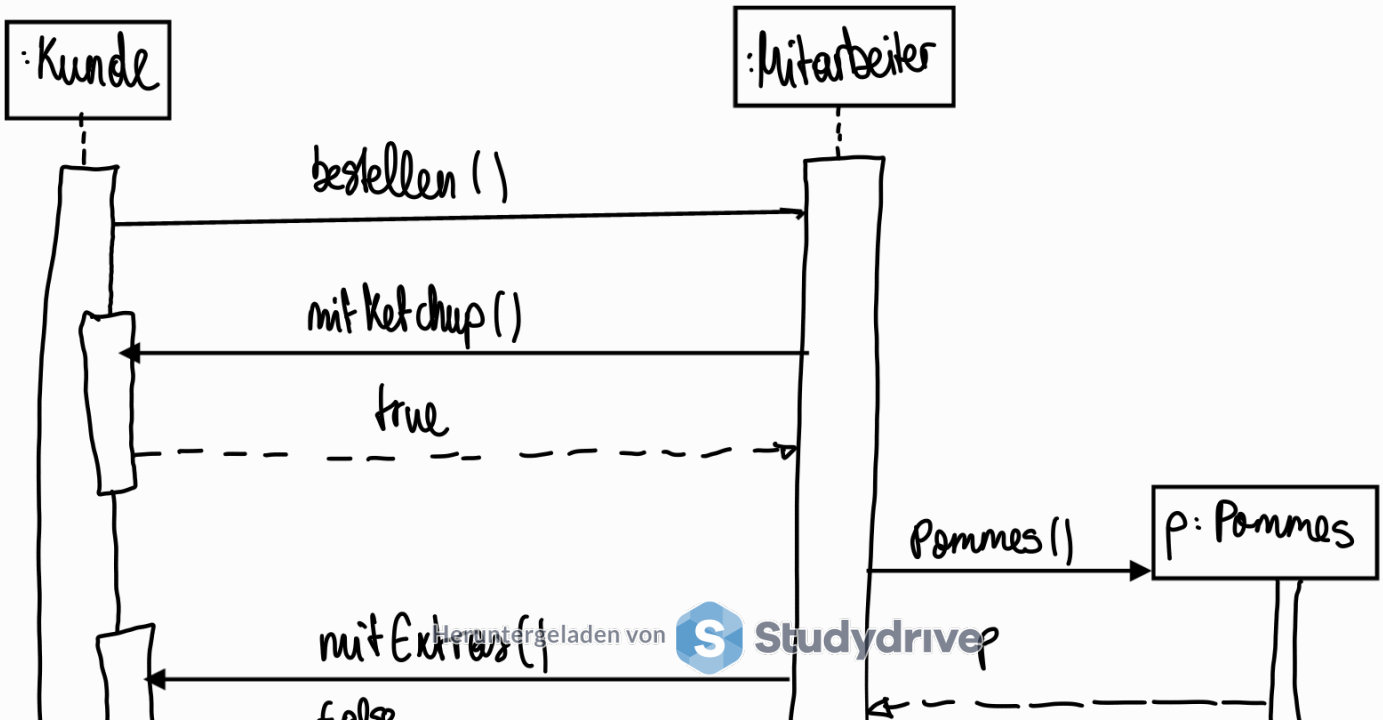


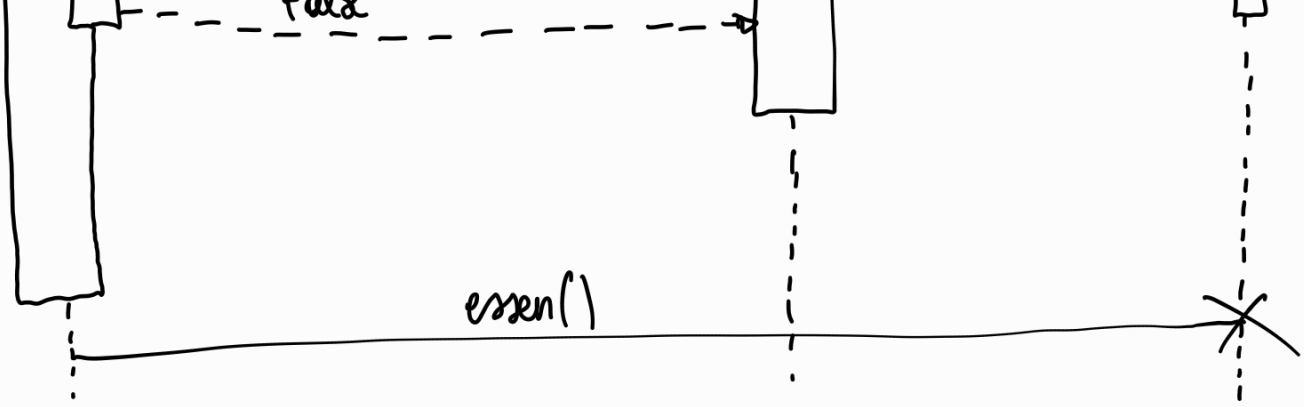
Konkrete Realisierungen weichen meist in Details ab (z.B. Interface Observer) !

Sequenzdiagramm:



Aufgabe 4.3: Sequenzdiagramme in der Analyse





Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Dipl.-Inform. Deni Raco

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 5

Abgabe: 14.12.2021 10:30 Uhr

Organisatorisches

Die Übungsaufgaben müssen in Gruppen von drei bis vier Personen abgegeben werden. Die Abgabe ist über den RWTHmoodle Lernraum der Vorlesung einzureichen. Alle Gruppenmitglieder müssen auf der Abgabe vermerkt sein, inkl. Matrikelnummer. Doppelabgaben führen zu Nichtbewertung. Die Rückgabe der Ergebnisse erfolgt über den RWTHmoodle Lernraum.

Aufgabe 5.1 (10 Punkte)

Sie befinden sich im Entwurf zweier separater Bibliothekssysteme. Dabei machen Sie sich genauere Gedanken über die Datenstruktur des Systems.

Teilaufgabe a) (5 Punkte)

Formalisieren Sie die folgende textuelle Beschreibung des Systems, indem Sie ein geeignetes Klassendiagramm erstellen.

Es werden zwei Arten von Bibliotheken separat voneinander entwickelt. Die erste Art von Bibliothek enthält Algorithmen für Probleme aus der Geometrie. Jede Bibliothek dieser Art und jeder Algorithmus haben dabei einen Namen. Des Weiteren enthält die Bibliothek geometrische Objekte. Diese sind Quadrate, Kreise und Geraden, es kann aber noch andere Objekte geben. Kreise bieten dabei jeweils eine Funktion, um ihren Mittelpunkt und ihren Radius als Gleitkommazahl auszugeben. Eine spezielle Art von Algorithmus ist dafür gedacht zu bestimmen, ob sich zwei Kreise schneiden.

Die zweite Art von Bibliothek ist eine für geometrische Objekte, die auf Benutzeroberflächen (GUI) angezeigt werden können. Eine GUI Bibliothek hat einen Namen enthält dabei GUI Elemente. Jedes GUI Element hat einen Namen und eine Methode, mit der man es auf eine Benutzeroberfläche zeichnen kann. Es gibt genau die GUI Elemente Quadrat, Kreis und Punkt. Ein Punkt besitzt zusätzlich seine x und y Koordinaten als Gleitkommazahlen. Ein Kreis besitzt genau einen Punkt, der seinen Mittelpunkt festlegt und einen Radius, der als Gleitkommazahl angegeben wird.

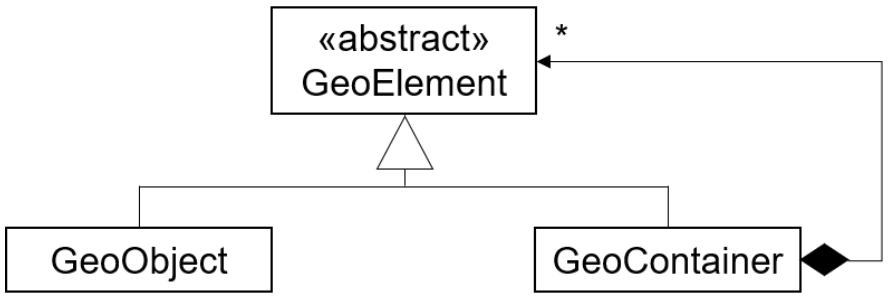
Teilaufgabe b) (3 Punkte)

Da die Algorithmen zu Beginn nicht auf den GUI Elementen ausgeführt werden können, soll es durch eine neue Anforderung an das System nun möglich sein, Algorithmen, die prüfen, ob sich zwei Kreise schneiden, für Kreise aus der GUI Bibliothek zu nutzen.

1) Welches Entwurfsmuster ist zur Umsetzung dieser Anforderung geeignet?
2) Erweitern Sie Ihr Klassendiagramm aus Teilaufgabe a) entsprechend um das Entwurfsmuster.

Teilaufgabe c) (2 Punkte)

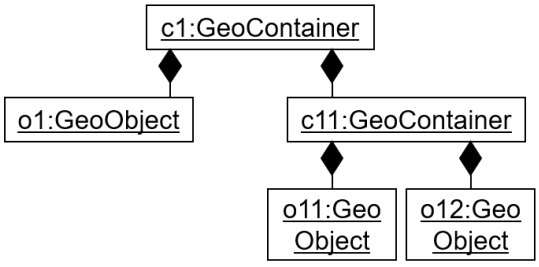
Gegeben ist folgendes Klassendiagramm:



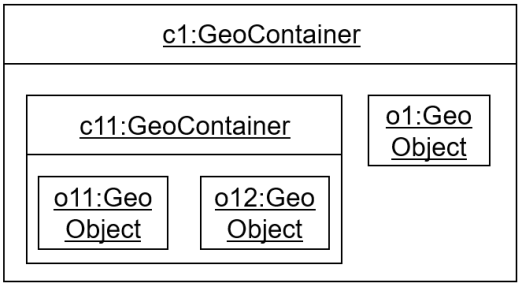
1) Um welches Entwurfsmuster handelt es sich?

Es sind folgende Objektdiagramme gegeben:

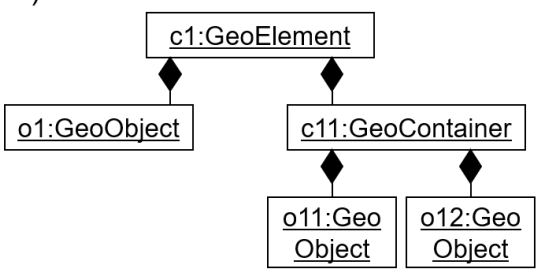
i)



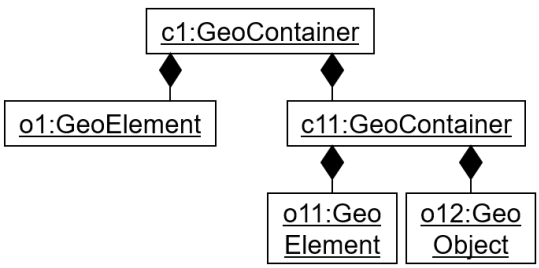
ii)



iii)



iv)



2) Geben Sie an, welche Objektdiagramme mögliche Instanziierungen des Klassendiagramms sind und welche nicht.

3) Welche Objektdiagramme sind äquivalent?

Aufgabe 5.2 (8 Punkte)

Die Firma Carmpere konzeptioniert die verschiedenen Use Cases zum Laden der Batterien, der von ihnen produzierten Autos mit Elektro- und Hybridmotoren. Es gibt Ladestationen und Schnellladestationen. Die Schnellladestationen sollen an stark frequentierten Tankstellen auf Autobahnen eingesetzt werden. Jede Schnellladestation soll alle Funktionalitäten bereitstellen können, die auch von einer Ladestation bereitgestellt werden. Fahrer interagieren mit Ladestationen. Ladestationen können zusätzlich auch mit Autos interagieren. Ladestationen sollen überprüfen können, ob ein Auto ordnungsgemäß mit der Ladestation über ein Ladekabel verbunden ist. Weiterhin sollen Ladestationen dazu in der Lage sein, die benötigte Lademenge für ein Auto zu berechnen. Ein Fahrer soll einen Ladewunsch an einer Ladestation äußern können. Das Äußern des Ladewunschs hat zur Folge, dass die Ladestation die Lademenge für das Auto berechnet. Wenn die Lademenge berechnet wird, soll auch überprüft werden, ob das Auto ordnungsgemäß mit der Ladestation verbunden ist. Ein Auto soll von einer Ladestation geladen werden können. Schnellladestationen können Autos zusätzlich schnellladen. Das Schnellladen eines Autos hat zur Folge, dass die Batterie des Autos schneller geladen wird als beim herkömmlichen Ladevorgang. Falls bei einer Ladestation ein Ladevorgang gestartet wird und die Ladestation eine Schnellladestation ist, dann kann auch der Schnellladevorgang ausgeführt werden. Fahrer sollen an Ladestationen bezahlen können. Es kann entweder per Carmpere App oder per Kreditkarte bezahlt werden.

Erstellen Sie ein Use Case Diagramm, das die softwarerelevanten Aspekte zum Laden der Batterien der Autos der Firma Carmpere darstellt.

Aufgabe 5.3 (10 Punkte)

Sie entwickeln eine Webanwendung für Diskussionsforen, die auf der Hauptseite eine Übersicht aller eingeloggten Benutzer anzeigen soll. Die Übersicht soll zudem anzeigen, in welchem Bereich der Seite sich ein eingeloggter Benutzer gerade befindet. Ausgeloggte Benutzer bleiben anonym.

Modellieren Sie den für eine solche Übersicht relevanten Status eines einzelnen Benutzers und die zugehörigen Statusänderungen als Zustandsdiagram. Folgende Ereignisse, Bedingungen und Aktionen stehen Ihnen dabei zur Verfügung. Verwenden Sie einen sinnvollen Oberzustand. Die Aktionen steuern hierbei die Anzeige des Benutzers in der Übersicht.

Ereignisse

- Benutzer logt sich ein (`Login`)
- Benutzer logt sich aus (`Logout`)
- Benutzer hat seit 5 Minuten keine Seite angefordert (`Timeout`)
- Benutzer ruft Forenübersicht auf (`RequestMainpage`)

- Benutzer ruft Diskussionsforum auf (`RequestBoard`)
- Benutzer ruft Diskussionsthema auf (`RequestTopic`)

Bedingungen

- Login war korrekt (`validLogin`)

Aktionen

- Ausgeloggten Benutzer aus der Übersicht entfernen (`removeFromList`)
- Eingeloggten Benutzer als Betrachter der Forenübersicht anzeigen (`onMainpage`)
- Eingeloggten Benutzer als Leser eines Diskussionsforums anzeigen (`readsBoard`)
- Eingeloggten Benutzer als Leser eines Diskussionsthemas anzeigen (`readsTopic`)

Zu Beginn ist ein Benutzer ausgeloggt. Er kann sich durch Eingabe eines korrekten Logins einloggen. Wenn er als eingeloggter Benutzer ein Diskussionsforum oder ein Diskussionsthema aufruft, wird das in der Übersicht entsprechend angezeigt. Er kann ein Diskussionsforum nur über die Forenübersicht aufrufen. Ebenso kann er ein Diskussionsthema nur aufrufen, wenn er ein Diskussionsforum liest. Er kann sowohl von einem Diskussionsforum als auch von einem Diskussionsthema auf die Forenübersicht zurückkehren.

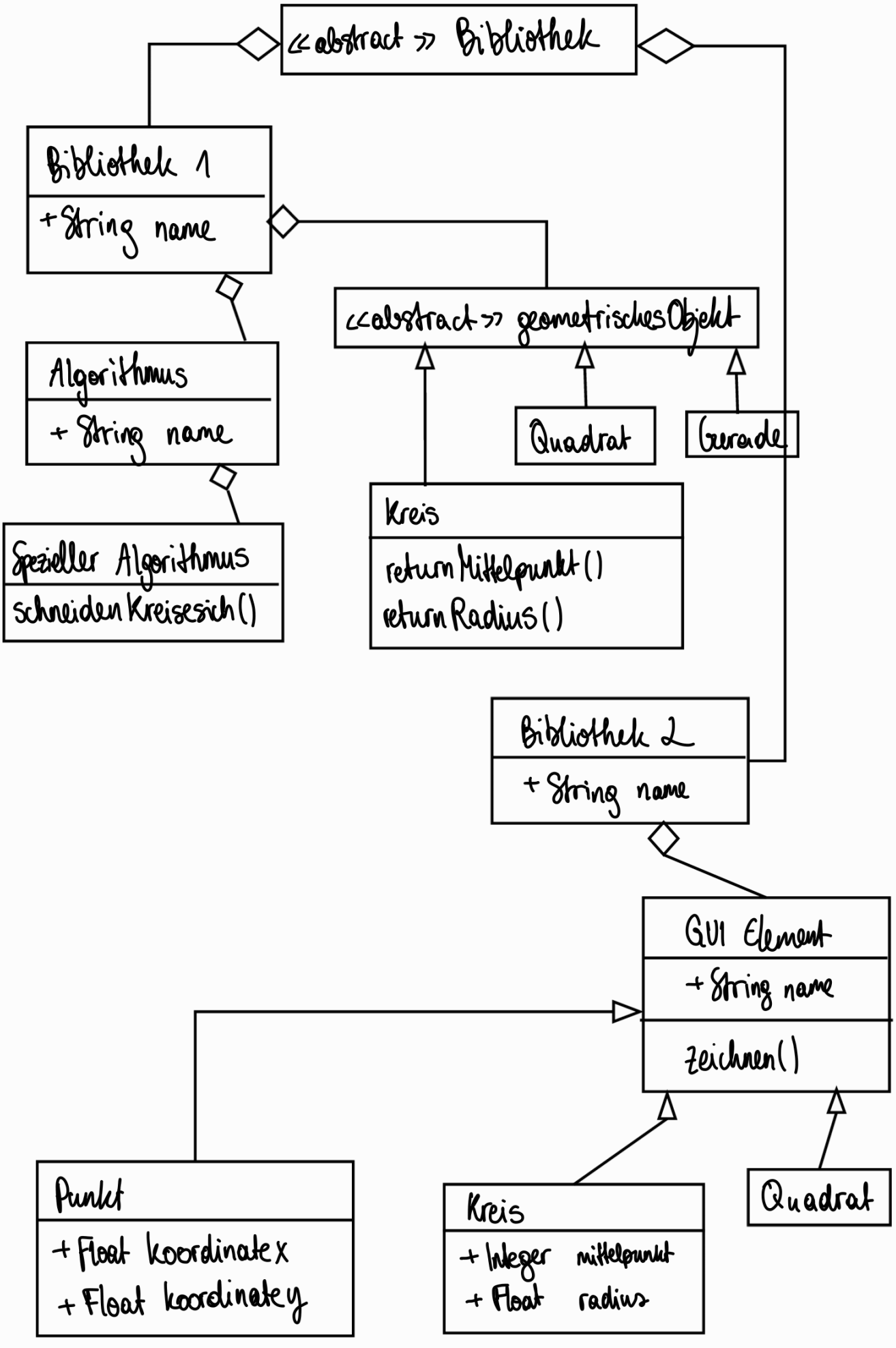
Der Benutzer kann sich jederzeit ausloggen. Wenn ein Benutzer länger als 5 Minuten keine neue Seite angefordert hat, wird er automatisch ausgeloggt. Ein ausgeloggter Benutzer wird aus der Übersicht entfernt.

Aufgabe 5.4 (4 Punkte)

Es ist folgende Beschreibung einer Bank gegeben. Erstellen Sie aus der gegebenen informellen Beschreibung ein passendes Klassendiagramm. Benutzen Sie mindestens einmal jeweils eine Komposition, eine Kardinalität, ein Attribut mit dem entsprechenden Typ, sowie eine Vererbungsbeziehung.

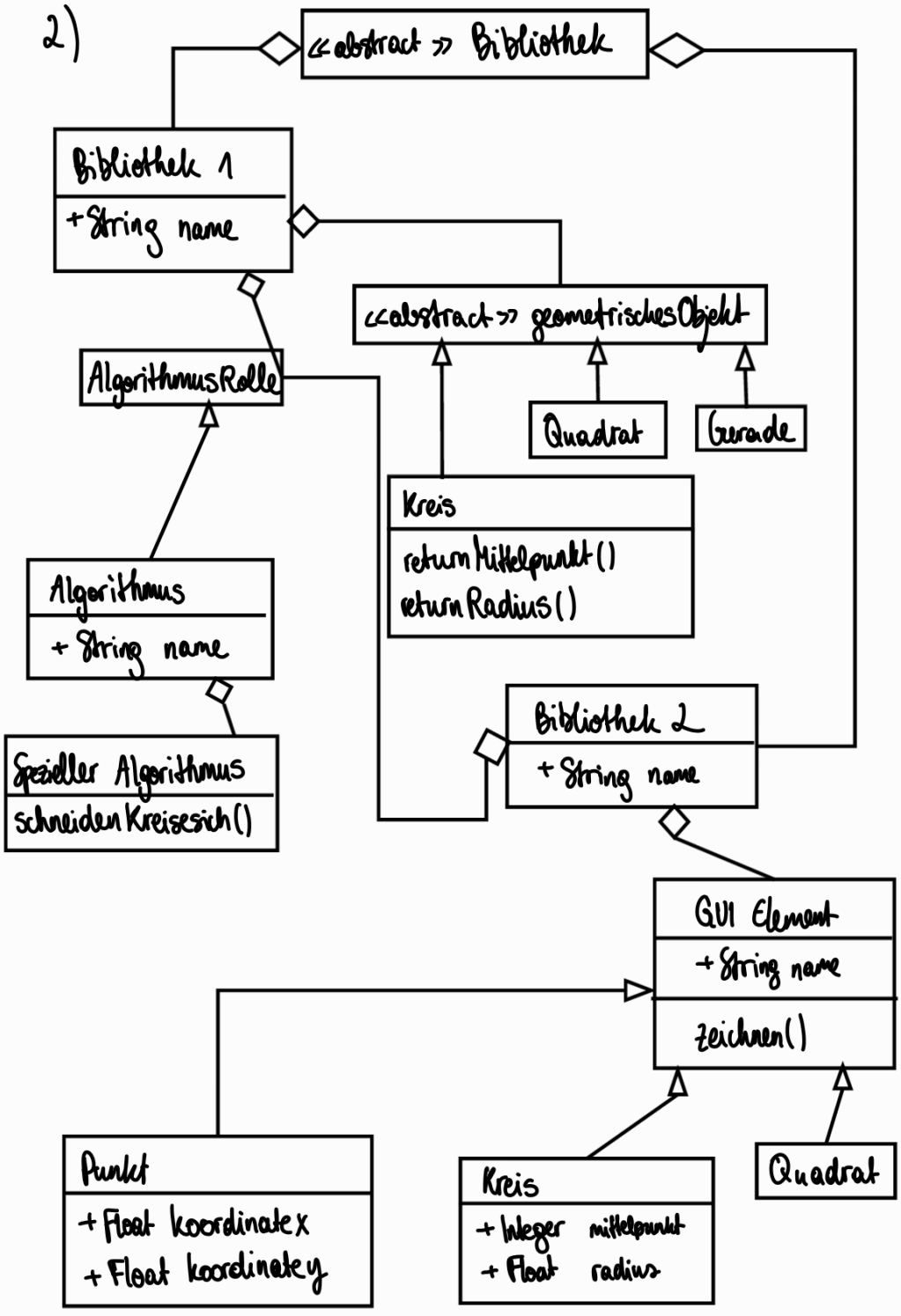
Eine Bank verwaltet beliebig viele ihrer Kunden. Für jeden Kunden sind Vorname und Name vom Typ String hinterlegt. Eine Bank besteht aus beliebig vielen Filialen. Ein Kunde hat mindestens ein und maximal fünf Konten, auf die er zugreifen kann. Jedes Konto gehört zu genau einem Kunden und ist exklusiv einer Bank zugeordnet. Es gibt insgesamt genau zwei Arten von Konten: Girokonten und Sparkonten. Ein Kunde der Bank muss genau ein Girokonto als Gehaltskonto führen. Jedes Konto verfügt über eine Kontonummer des Typen int und einen aktuellen Kontostand vom Typ double. Girokonten haben zusätzlich ein Überziehungslimit vom Typ double.

Aufgabe 5.1 a)



b) 1) Das Entwurfsmuster „Wechselnde Rolle" ist hierfür geeignet.

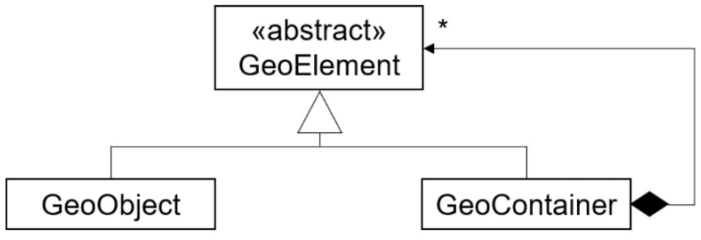
2)



c)

Teilaufgabe c) (2 Punkte)

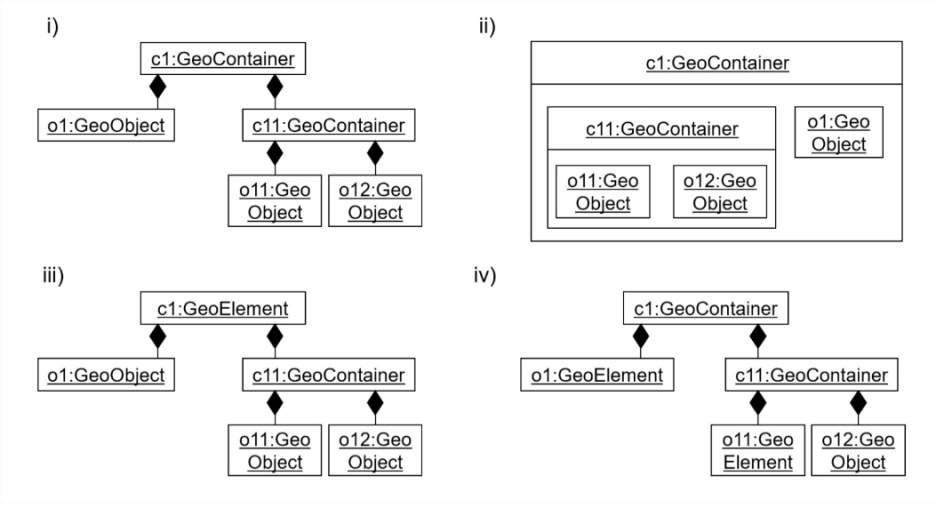
Gegeben ist folgendes Klassendiagramm:



1) Um welches Entwurfsmuster handelt es sich?

Brücke
Stellvertreter

Es handelt sich hierbei um das Entwurfsmuster „



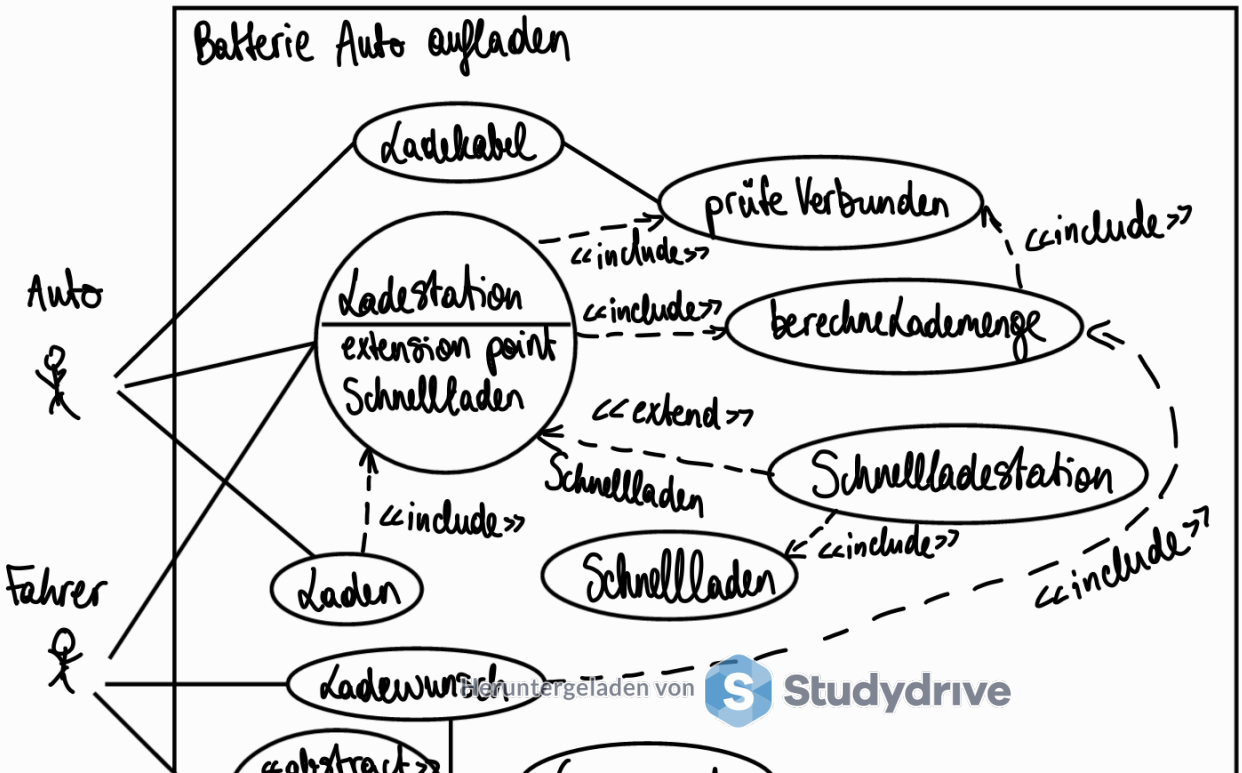
2) Geben Sie an, welche Objektdiagramme mögliche Instanziierungen des Klassendiagramms sind und welche nicht.

Mögliche Instanziierungen des Klassendiagramms sind ii) und iii). i) und iv) sind nicht möglich.

3) Welche Objektdiagramme sind äquivalent?

Äquivalent sind i) und ii).

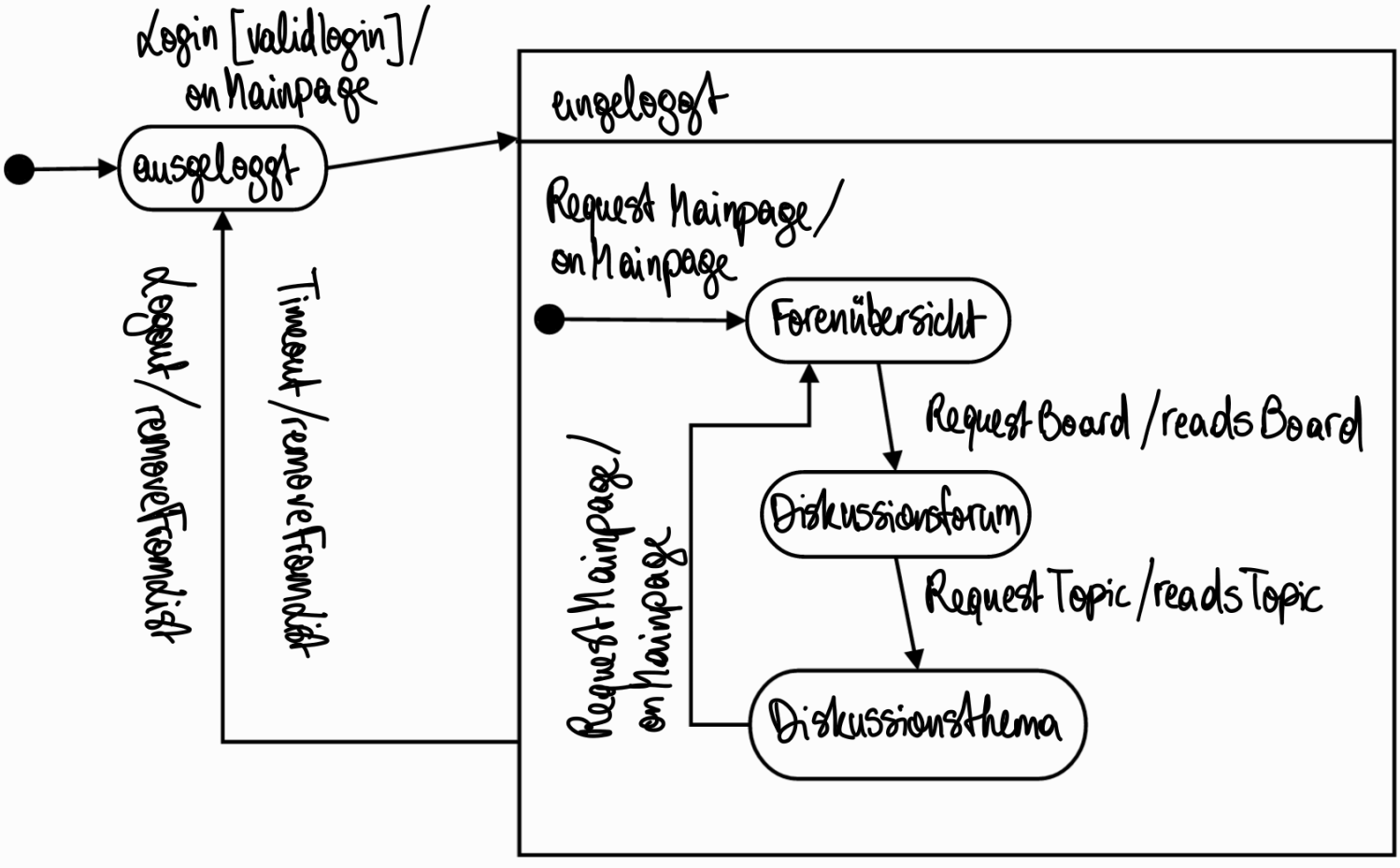
Aufgabe 5.2



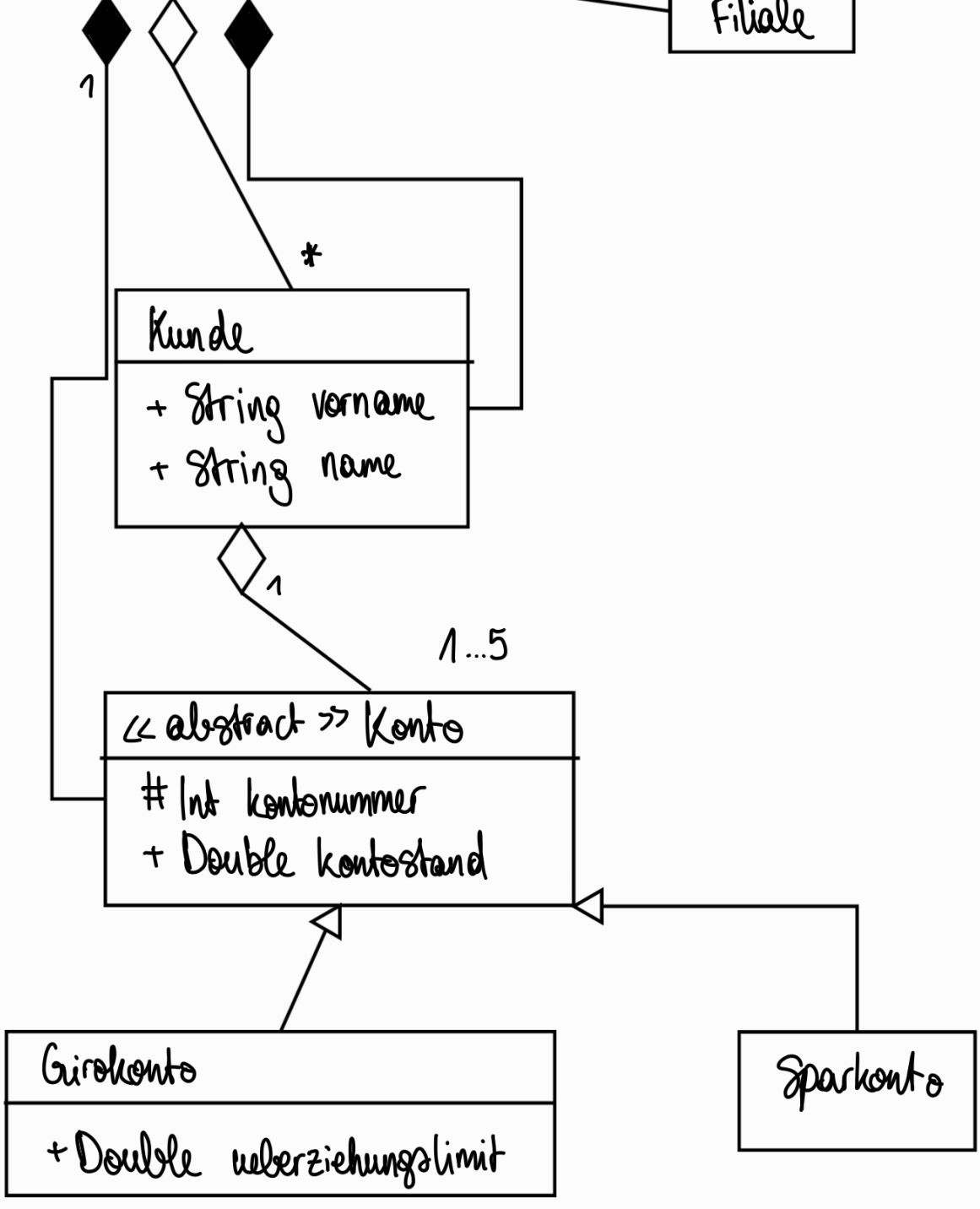
⇒ Ladestation & Schnellladestation müssten eigene Akteure sein

⇒ Schnellladen Syntax falsch

Aufgabe 5.3



Aufgabe 5.4



Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Dipl.-Inform. Deni Raco

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 6

Abgabe: 21.12.2021 10:30 Uhr

Aufgabe 6.1 (6 Punkte)

Man muss dazu in der Lage sein, sich zielgerichtet und eigenständig in eine neue Programmiersprache einzuarbeiten. In dieser Aufgabe werden Sie Programme zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen in sechs verschiedenen Programmiersprachen implementieren.

Der größte gemeinsame Teiler $ggT(a, b)$ zweier ganzer Zahlen a und b ist definiert als die ganze Zahl n mit der Eigenschaft, dass sie Teiler von a sowie von b ist und dass jede ganze Zahl, die ebenfalls ein Teiler von a sowie von b ist, auch ein Teiler von n ist.

Betrachten wir zum Beispiel die Zahlen 12 und 18.

- Die Zahl 12 hat die Teiler 1, 2, 3, 4, 6, 12.
- Die Zahl 18 hat die Teiler 1, 2, 3, 6, 9, 18.
- Die gemeinsamen Teiler von 12 und 18 sind 1, 2, 3, 6.
- Der größte gemeinsame Teiler von 12 und 18 ist also $ggT(12,18) = 6$.

Der größte gemeinsame Teiler zweier Zahlen kann leicht mit dem euklidischen Algorithmus berechnet werden. Die folgende Abbildung zeigt diesen Algorithmus in Pseudocode:

```
01  In:  Two integers a and b
02  Out: An integer
03
04  if a = 0 then
05    return abs(b)
06  endif
07  if b = 0 then
08    return abs(a)
09  endif
10  while(b ≠ 0) do
11    h := a % b
12    a := b
13    b := h
14  endwhile
15  return abs(a)
```

- Die Funktion `abs` ist eine Hilfsfunktion, die den Betrag einer ganzen Zahl zurückgibt. Beispielsweise gibt `abs(-3)` die Zahl 3, `abs(5)` die Zahl 5 und `abs(0)` die Zahl 0 zurück.
- Der Operator % ist der Modulo Operator und gibt den Rest der Ganzzahldivision zweier ganzer Zahlen zurück. Beispielsweise gibt 14 % 3 die Zahl 2 zurück.

Der Algorithmus gibt bei der Eingabe folgender Zahlen korrekte Resultate zurück:

- In: 12, 18 Out: 6
- In: 16, 20 Out: 4
- In: 120, 900 Out: 60
- In: 105, 26 Out: 1

Hinweise:

- Sie können die Lösungen der folgenden Aufgabenteile in Ihre Abgabe integrieren und müssen nicht zusätzliche Dateien einreichen. Erstellen Sie Ihre Lösungen bitte so, dass sie leicht kopierbar sind, falls möglich, um den Korrekturaufwand zu verringern.
- Falls Sie Probleme bei der Installation oder Anwendung eines Compilers oder Interpreters haben, dann versuchen Sie Ihre Probleme mithilfe einer Internetsuchmaschine zu beheben.
- Ihre Lösungen müssen jeweils den gesamten Quellcode beinhalten.
- Der Quellcode muss jeweils kompilieren.
- Sie können beliebig viele zusätzliche Methoden, Funktionen, Klassen, etc. implementieren und beliebige Standardbibliotheken nutzen.

Teilaufgabe a) (1 Punkt)

- Implementieren Sie den euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen in der Programmiersprache **Java**. Implementieren Sie zusätzlich die `main` Methode zum simplen Testen Ihrer Implementierung. Lassen Sie sich zum Testen die berechneten Werte für die obigen vier Eingabepaare auf der Konsole ausgeben.
- Sie können einen Java Compiler auf der folgenden Webseite downloaden: https://www.oracle.com/de/java/technologies/javase-jdk15-downloads.html

Benutzen Sie folgendes Gerüst für Ihren Quellcode:

```
01  public class GcdCalculator {
02    public static void main(String[] args) {
03      // Your code for testing ...
04    }
05
06    public int gcd(int a, int b) {
07      // Your code ...
08    }
09  }
```

Teilaufgabe b) (1 Punkt)

Implementieren Sie den euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen in der Programmiersprache **C**. Implementieren Sie zusätzlich die `main` Methode zum simplen Testen Ihrer Implementierung. Lassen Sie sich zum Testen die berechneten Werte für die obigen vier Eingabepaare auf der Konsole ausgeben.

- Die folgende Webseite enthält Instruktionen zur Installation eines C Compilers: http://hades.mech.northwestern.edu/index.php/Installing_a_C_Compiler_and_IDE

Benutzen Sie folgendes Gerüst für Ihren Quellcode:

```c
#include <stdio.h>
#include <stdlib.h>

// Your code for declarations if necessary ...

int main() {
  // Your code for testing ...
}

int gcd(int a, int b) {
  // Your code ...
}
```

Teilaufgabe c) (1 Punkt)

Implementieren Sie den euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen in der Programmiersprache **Python**.

- Sie können einen Python Compiler auf der folgenden Webseite downloaden: https://www.python.org/downloads/

Benutzen Sie folgendes Gerüst für Ihren Quellcode:

```python
def gcd(a, b):
    // Your code for gcd here ...

print(gcd(12,18))    # 6
print(gcd(16,20))    # 4
print(gcd(120,900))  # 60
print(gcd(105,26))   # 1
```

Teilaufgabe d) (1 Punkt)

Implementieren Sie den euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen in der Programmiersprache **JavaScript**.

- Sie können eine Anleitung zum Download eines und für die ersten Schritte mit einem JavaScript Interpreter auf der folgenden Webseite finden: https://gridscale.io/community/tutorials/node-js-beginner-guide/

Benutzen Sie folgendes Gerüst für Ihren Quellcode:

```
01  function gcd(a, b) {
02    // Your code ...
03  }
04
05  console.log(gcd(12,18))    // 6
06  console.log(gcd(16,20))    // 4
07  console.log(gcd(120,900))  // 60
08  console.log(gcd(105,26))   // 1
```

Teilaufgabe e) (1 Punkt)

Implementieren Sie den euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen in der Programmiersprache **Go**. Implementieren Sie zusätzlich die `main` Methode zum simplen Testen Ihrer Implementierung. Lassen Sie sich zum Testen die berechneten Werte für die obigen vier Eingabepaare auf der Konsole ausgeben.

- Sie können einen Go Interpreter auf der folgenden Webseite downloaden: https://golang.org/doc/install

Benutzen Sie folgendes Gerüst für Ihren Quellcode:

```
01  package main
02
03  import "fmt"
04
05  func main() {
06    // Your code for testing ...
07  }
08
09  func gcd(a, b int) int {
10    // Your code ...
11  }
```

Teilaufgabe f) (1 Punkt)

Implementieren Sie den euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen in der Programmiersprache **Kotlin**. Implementieren Sie zusätzlich die `main` Methode zum Testen Ihrer Implementierung. Lassen Sie sich zum Testen die berechneten Werte für die obigen vier Eingabepaare auf der Konsole ausgeben.

- Instruktionen zum Download und zur Nutzung eines Kotlin Compilers können auf der folgenden Webseite gefunden werden: https://kotlinlang.org/docs/tutorials/command-line.html

Benutzen Sie folgendes Gerüst für Ihren Quellcode:

```
01  fun main() {
02    // Your code for testing ...
03  }
04
05  fun gcd(a: Int, b: Int): Int {
06    // Your code ...
07  }
```

Aufgabe 6.2 (4 Punkte)

Teilaufgabe a) (2 Punkte)

Welche Aspekte/Eigenschaften eines Systems lassen sich durch UML Zustandsdiagramme (Statecharts) und UML Sequenzdiagramme modellieren? Wo liegt der wesentliche semantische Unterschied zwischen den beiden Diagrammtypen?

Teilaufgabe b) (2 Punkte)

Lassen sich UML Aktivitätsdiagramme und Zustandsdiagramme zur Modellierung eines Softwaresystems kombinieren? Wenn ja, welche Modellierungselemente eines Zustandsdiagramms könnten durch Aktivitätsdiagramme detailliert werden?

Aufgabe 6.2 (4 Punkte)

Teilaufgabe a) (2 Punkte)

Welche Aspekte/Eigenschaften eines Systems lassen sich durch UML Zustandsdiagramme (Statecharts) und UML Sequenzdiagramme modellieren? Wo liegt der wesentliche semantische Unterschied zwischen den beiden Diagrammtypen?

Teilaufgabe b) (2 Punkte)

Lassen sich UML Aktivitätsdiagramme und Zustandsdiagramme zur Modellierung eines Softwaresystems kombinieren? Wenn ja, welche Modellierungselemente eines Zustandsdiagramms könnten durch Aktivitätsdiagramme detailliert werden?

UML Aktivitätsdiagramme können Verhalten sehr gut modellieren, indem parallele, sequentiell abhängige und alternative Prozesse und Aktivitäten über verschiedene Symbole dargestellt werden. Alternativ kann über Swimlanes noch deutlicher herausgestellt werden, welcher Akteur für welchen Prozess verantwortlich ist.

Zustandsdiagramme beschreiben das Verhalten von einzelnen Objekten durch endliche Zustandsübergangsdiagramme, die ähnlich wie endliche Automaten funktionieren. Der Fokus liegt hier auf möglichen Abläufen und der Verbindung dieser Abläufe zum aktuellen Zustand des Objekts (man kann bspw. Keine Tür schließen, die bereits geschlossen ist).

UML Sequenzdiagramme können die Interaktion zwischen Objekten sehr gut modellieren. Hierbei werden jedem Akteur verschiedene Aktivitäten zugeordnet, die eventuell voneinander abhängig sein können (zB Postbote kann Brief erst an Haushalt A verteilen, sobald er den Brief aus dem Verteilerzentrum erhalten hat), was durch diverse Pfeile gekennzeichnet wird. Der Fokus liegt auf dem Informationsfluss zwischen Objekten, die Zeit fließt nach unten.

Der wesentliche semantische **Unterschied** liegt in der Lese- und Schreibrichtung: UML Sequenzdiagramme werden von oben nach unten und von links nach rechts gelesen, UML Aktivitätsdiagramme und Aktivitätsdiagramme von links nach rechts. Jedes Diagramm hat leicht andere Symbole und beschreibt andere Teilaspekte eines Programms, der Fokus liegt auf anderen Aspekten.

Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Dipl.-Inform. Deni Raco

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 7

Abgabe: 11.01.2022 10:30 Uhr

Aufgabe 7.1 (10 Punkte)

Gegeben ist die Methode `closure(Boolean[][] m)`, welche die transitive Hülle eines Graphen berechnet. Der Eingabegraph ist mittels einer Adjazenzmatrix definiert, die durch das Array m kodiert ist.

```
    public static Optional<Boolean[][]> closure(Boolean[][] m) {
01    for(int i = 0; i < m.length; i++) {
02      if(m.length != m[i].length) {
03        return Optional.empty();
        }
      }
04    int length = m.length;
05    for(int k = 0; k < length; k++) {
06      for(int i = 0; i < length; i++) {
07        if(m[i][k] != null && m[i][k]) {
08          for(int j = 0; j < length; j++) {
09            if(m[k][j] != null && m[k][j]) {
10              m[i][j] = true;
              }
            }
          }
        }
      }
11    return Optional.of(m);
    }
```

Teilaufgabe a) Kontrollflussgraph erstellen (2,5 Punkte)

Konstruieren Sie einen Kontrollflussgraphen für die Methode `closure(Boolean[][] m)`. Benutzen Sie die links vom Methodenrumpf angegebenen Nummern zur Beschriftung der zugehörigen Knoten im Kontrollflussgraphen.

Teilaufgabe b) Anweisungsüberdeckungstest (3 Punkte)

Nennen Sie eine repräsentative Eingabemenge für einen Anweisungsüberdeckungstest der Methode `closure(Boolean[][] m)`. Für die Angabe der Eingabewerte können Sie die übliche Matrixschreibweise nutzen. Beispielsweise repräsentiert die Matrix $\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$ ein Array A mit `A[0][0]=1`, `A[1][0]=2`, `A[0][1]=3` und `A[1][1]=4`. Ihre Eingabewerte dürfen maximal 3 x 3 Matrizen sein.

Teilaufgabe c) Erreichbarkeit von Anweisungen (2 Punkte)

Gibt es eine Methode, deren Kontrollflussgraph nur Knoten enthält, die vom Startknoten aus erreichbar sind, obwohl es keine Menge von Eingaben gibt, sodass jede Anweisung der Methode mindestens bei einer Eingabe ausgeführt wird?

Falls Sie diese Frage negativ beantworten, dann begründen Sie die Antwort. Falls Sie die Frage positiv beantworten, dann geben Sie eine Methode an (z.B. in Java), die obiges erfüllt.

Teilaufgabe d) Dominanzrelationen in Kontrollflussgraphen (1,5 + 1 Punkte)

Seien u und v zwei Knoten eines Kontrollflussgraphen G. Der Knoten u dominiert den Knoten v in G genau dann, wenn jeder Pfad in G, der im Startknoten von G beginnt und den Knoten v besucht auch den Knoten u besucht.

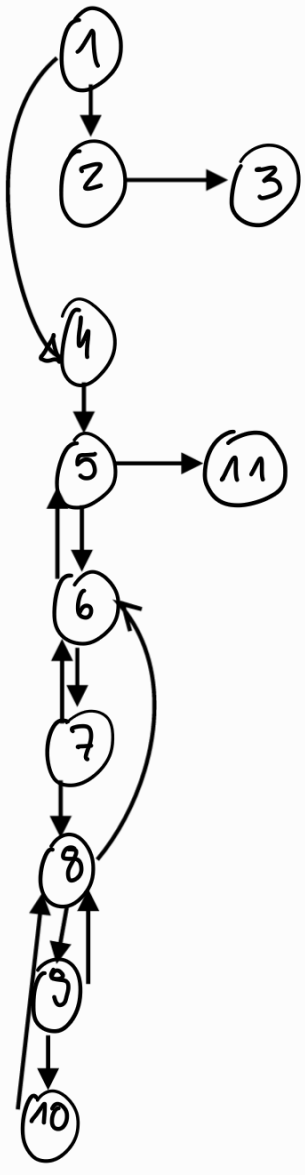
Beurteilen Sie für jede der folgenden Aussagen, ob sie wahr ist und begründen Sie die jeweiligen Antworten mit maximal drei Sätzen oder durch ein Gegenbeispiel.

1) Jeder Knoten eines Kontrollflussgraphen G dominiert sich selbst in G.
2) Wenn u und v zwei Knoten eines Kontrollflussgraphen G sind und der Knoten u den Knoten v in G dominiert, dann dominiert der Knoten v den Knoten u in G.
3) Wenn u, v und w drei Knoten eines Kontrollflussgraphen G sind, der Knoten u den Knoten v in G dominiert und der Knoten v den Knoten w in G dominiert, dann dominiert der Knoten u den Knoten w in G.

Wozu und wie kann eine vollständig bekannte Dominanzrelation für einen Kontrollflussgraphen bei einer Anweisungsüberdeckungstesterstellung ausgenutzt werden?

Aufgabe 7.1

a) Kontrollflussgraph



b) Anweisungsüberdeckungstest

(true
false)

m mit m.length > 0 und m[i][k] == m[k][j] == true und i, j, k >= 0

m[0][0] == true erfüllt Bedingung

wähle m = (true)

c) Erreichbarkeit von Anweisungen

⇒ fa: Methode in Java:

```
public boolean strange() {
    if (true) {
        return false;
    }
    return true;
}
```

d) Dominanzrelationen

1) Jeder Knoten von G dominiert sich selbst in G.

⇒ wahr

2) Wenn u und v 2 Knoten in G und u v dominiert, dann dominiert v u.

⇒ falsch

3) Wenn u, v und w 3 Knoten in G, u v in G dominiert, v w in G dominiert, dann dominiert u w in G.

⇒ wahr

Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Dipl.-Inform. Deni Raco

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 8

Abgabe: 18.01.2022 10:30 Uhr

Aufgabe 8.1 (5 Punkte)

Gegeben ist die folgende Methode zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen:

```
    public int gcd(int a, int b) {
01    if (a == 0) {
02      return Math.abs(b);
      }

03    while (b != 0) {
04      int h = a % b;
        a = b;
        b = h;
      }
05    return Math.abs(a);
    }
```

Teilaufgabe a) Kontrollflussgraph erstellen (1 Punkt)

Konstruieren Sie einen Kontrollflussgraphen für die Methode `gcd(int a, int b)`. Benutzen Sie die links vom Methodenrumpf angegebenen Nummern zur Beschriftung der zugehörigen Knoten im Kontrollflussgraphen.

Teilaufgabe b) Anweisungsüberdeckungstest erstellen (1 Punkt)

Nennen Sie eine repräsentative Eingabemenge für einen Anweisungsüberdeckungstest der Methode `gcd(int a, int b)` und geben Sie für jede Eingabe die Reihenfolge der besuchten Knoten im Kontrollflussgraphen an.

Teilaufgabe c) Zweigüberdeckungstest (1 Punkt)

Ist jede repräsentative Eingabemenge für einen Anweisungsüberdeckungstest der Methode `gcd(int a, int b)` auch eine repräsentative Eingabemenge für einen Zweigüberdeckungstest der Methode? Begründen Sie Ihre Antwort.

Teilaufgabe d) Anweisungs- vs. Zweig- Überdeckungstests (1 Punkt)

Warum sind Zweigüberdeckungstests für Kontrollflussgraphen, in denen alle Knoten vom Startknoten aus erreichbar sind, mindestens genau so stark wie Anweisungsüberdeckungstests für diese Kontrollflussgraphen?

Teilaufgabe e) Zweig- vs. Pfad- Überdeckungstests (1 Punkt)

In welchen Fällen ist ein Pfadüberdeckungstest stärker als ein Zweigüberdeckungstest?

Aufgabe 8.2 (5 Punkte)

Die Methode `solveKnapsack(int[] weights, int[] values, int bound)` löst ein Knappsackproblem mit den Gewichten `weights`, Werten `values` und Rucksackkapazität `bound`. Der Wert `weights[i]` repräsentiert das Gewicht des Gegenstands `i`. Analog repräsentiert der Wert `values[i]` den Wert des Gegenstands `i`.

```
   Optional<Integer> solveKnapsack(int[] weights, int[] values, int bound) {
01   if(weights.length != values.length || bound <= 0) {
02     return Optional.empty();
     }
03   int objects = weights.length;
     int[][] r = new int[objects + 1][bound + 1];
04   for(int i = objects - 1; i >= 0; i--) {
05     for(int j = 1; j <= bound; j++) {
06       if(weights[i] <= j) {
07         int valWithI = values[i] + r[i+1][j-weights[i]];
           int valWithoutI = r[i+1][j];
08         if(valWithI > valWithoutI) {
09           r[i][j] = valWithI;
           } else {
10           r[i][j] = valWithoutI;
           }
         } else {
11         r[i][j] = r[i+1][j];
         }
       }
     }
12   return Optional.of(r[0][bound]);
   }
```

Teilaufgabe a) Kontrollflussgraph erstellen (3 Punkte)

Konstruieren Sie einen Kontrollflussgraphen für die Methode `solveKnapsack`. Benutzen Sie die links vom Methodenrumpf angegebenen Nummern zur Beschriftung der zugehörigen Knoten im Kontrollflussgraphen. Nutzen Sie zur Konstruktion des Kontrollflussgraphen die nächste Seite.

Teilaufgabe b) Anweisungsüberdeckungstest (2 Punkte)

Nennen Sie eine repräsentative Eingabemenge mit höchstens drei verschiedenen Eingaben für einen Anweisungsüberdeckungstest der Methode `solveKnapsack` und geben Sie für

jede Eingabe die Reihenfolge der besuchten Knoten im Kontrollflussgraphen an. Für die Angabe der Array-Eingabewerte können sie die übliche Tupelschreibweise nutzen. Beispielsweise repräsentiert das Tupel (1, 2, 3) ein Array A mit `A[0]=1`, `A[1]=2`, `A[2]=3`. Das Tupel () repräsentiert ein leeres Array. Die angegebenen Arrays dürfen jeweils nicht mehr als zwei Elemente enthalten. Sie können zur Angabe der Eingaben folgendes Schema verwenden:

1. Eingabe

   weights =

   values =

   bound =

   Reihenfolge der besuchten Knoten =

2. Eingabe

   weights =

   values =

   bound =

   Reihenfolge der besuchten Knoten =

3. Eingabe

   weights =

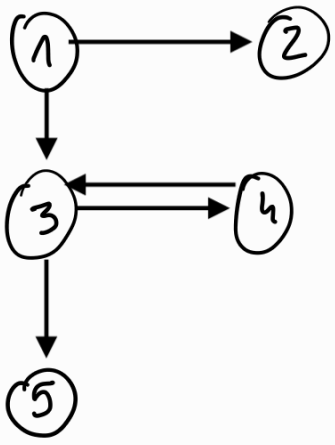
   values =

   bound =

   Reihenfolge der besuchten Knoten =

Aufgabe 8.1

a) Kontrollflussgraph



b) Anweisungsüberdeckungtest

Eingabe 1: a=0, b=1

Reihenfolge: 1, 2

Eingabe 2: a=1, b=1

Reihenfolge: 1, 3, 4, 3, 5

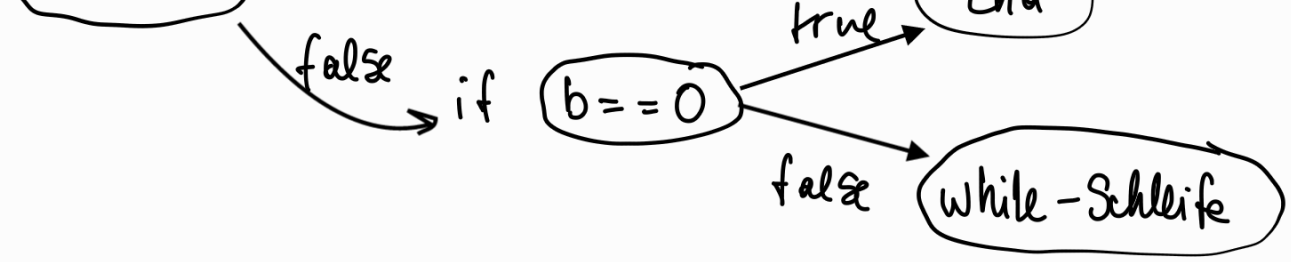
Eingabe 3: a=1 b=0

Reihenfolge: 1, 3, 5

c) Zweigüberdeckungstest

Ja, weil bei jeder Fallunterscheidung beide Zweige ausgeführt werden (Bedingung = true und Bedingung = false), d.h.:

Die Testfälle in b) überdecken alle Zweige.



d)

d) Warum sind Zweigüberdeckungstests für Kontrollflussgraphen, in denen alle Knoten vom Startknoten aus erreichbar sind, mindestens genau so stark wie Anweisungsüberdeckungstests für diese Kontrollflussgraphen?

- Wenn eine Eingabemenge dazu führt, dass jede Kante in einem solchen Kontrollflussgraphen mindestens einmal besucht wird, dann führt diese Eingabemenge auch dazu, dass jeder Knoten in diesem Kontrollflussgraphen mindestens einmal besucht wird.
- Alternative Antwort: Wenn für eine Eingabemenge bei einem solchen Kontrollflussgraphen bei jeder Fallunterscheidung beide Zweige (Bedingung = wahr und Bedingung = falsch) mindestens einmal ausgeführt werden, dann werden insbesondere auch alle Anweisungen im Programm mindestens einmal ausgeführt.

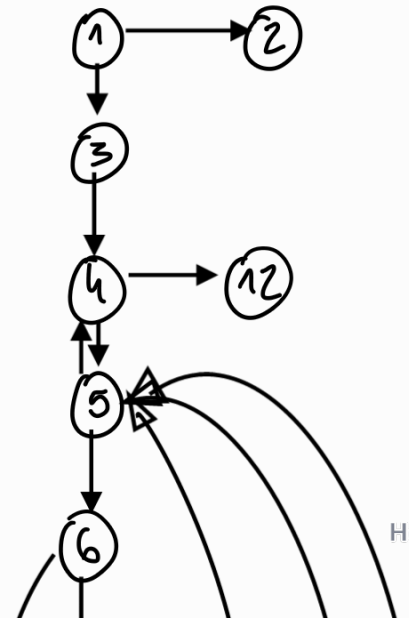
e)

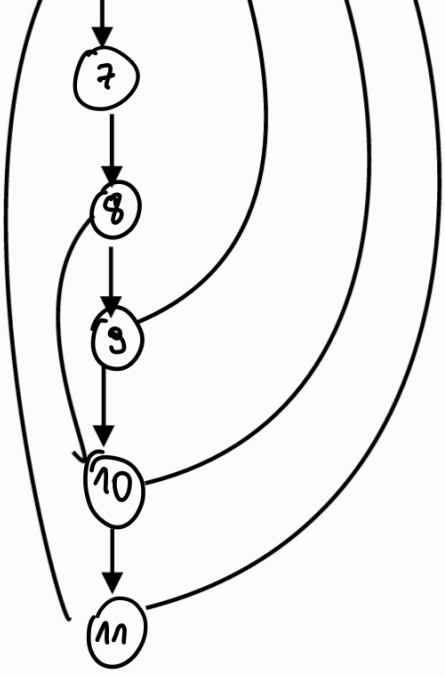
e) In welchen Fällen ist ein Pfadüberdeckungstest stärker als ein Zweigüberdeckungstest?

- Wenn das Programm komplexe Verzweigungsbedingungen enthält oder wenn Schleifen durch den Zweigüberdeckungstest nur unzureichend getestet werden.

Aufgabe 8.2

a) Kontrollflussgraph





b) Anweisungsüberdeckungstest

1) weights = ( )
values = ( )
bound = −1

Reihenfolge: 1, 2

2) weights = (2)
values = (1)
bound = 2

Reihenfolge: 1, 3, 4, 5, 6, 11, 5, 6, 7, 8, 9, 5, 4, 12

3) weights = (1)
values = (0)
bound = 1

Reihenfolge: 1, 3, 4, 5, 6, 7, 8, 10, 5, 4, 12

Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Dipl.-Inform. Deni Raco

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 9

Abgabe: 25.01.2022 10:30 Uhr

Aufgabe 9.1 (3 Punkte)

In einem Java Projekt zur Entwicklung von Software für elektrische Autos wird Maven als Build Tool genutzt. Die Entwicklungsartefakte werden über ein Maven Repository verwaltet. Es sei folgenden Maven *pom* gegeben:

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <project xmlns="http://maven.apache.org/POM/4.0.0"
03          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
05   <modelVersion>4.0.0</modelVersion>
06
07   <groupId>org.example</groupId>
08   <artifactId>elons-electric-cars</artifactId>
09   <packaging>pom</packaging>
10   <version>1.0-SNAPSHOT</version>
11
12   <properties>
13     <maven.compiler.source>8</maven.compiler.source>
14     <maven.compiler.target>8</maven.compiler.target>
15     <guava.version>25.1-jre</guava.version>
16     <junit.version>4.13.1</junit.version>
17   </properties>
18
19   <dependencies>
20     <dependency>
21       <groupId>com.google.guava</groupId>
22       <artifactId>guava</artifactId>
23       <version>${guava.version}</version>
24       <scope>compile</scope>
25     </dependency>
26     <dependency>
27       <groupId>junit</groupId>
28       <artifactId>junit</artifactId>
29       <version>${junit.version}</version>
30       <scope>test</scope>
31     </dependency>
32     <dependency>
33       <groupId>org.apache.commons</groupId>
34       <artifactId>commons-lang3</artifactId>
35       <version>3.11</version>
36     </dependency>
37   </dependencies>
38 </project>
```

Erstellen Sie eine äquivalente *build.gradle* Datei und beachten Sie dabei folgende Struktur:

```
01  plugins {
02  // ...
03  }
04
05  ext {
06  // ...
07  }
08
09  dependencies {
10  // ...
11  }
12
13  group = // ...
14  version = // ...
15
16  java.sourceCompatibility = JavaVersion.VERSION_1_8
```

Hinweis: Angaben zu repositories oder publishing müssen in der build.gradle nicht gemacht werden, sondern nur die durch „// …“ markierten Bereiche bzw. Werte müssen befüllt werden.

Aufgabe 9.2 (5 Punkte)

Die Firma MelonSoft bietet konfigurierbare Tabletcomputer, welche PadMy genannt werden, zum Verkauf an. Jeder PadMy besteht aus einem Display, einer Speichereinheit und einem Prozessor. Ein PadMy Display hat entweder eine Größe von 10,2”, eine Größe von 11” oder eine Größe von 12,9”. Für jedes PadMy stehen die Prozessoren P100 oder P200 zur Auswahl. Ein PadMy kann mit WiFi ausgestattet werden. Falls ein PadMy mit WiFi ausgestattet ist, dann kann es auch mit Mobilfunk ausgestattet werden. Der Speicher hat entweder eine Größe von 64GB, von 128GB oder von 256GB. Ein Speicher von 64GB wird ausschließlich mit PadMy Varianten angeboten, die einen P100 Prozessor und kein 12,9” großes Display verbaut haben. Ein Speicher von 256GB wird ausschließlich in PadMy Varianten verbaut, die eine Displaygröße von 12,9” haben.

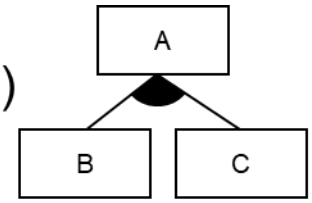
Teilaufgabe a) Feature Diagramm aus Spezifikation erstellen (3,5 Punkte)

Modellieren Sie die PadMy Konfigurationen mithilfe eines Feature Diagramms.

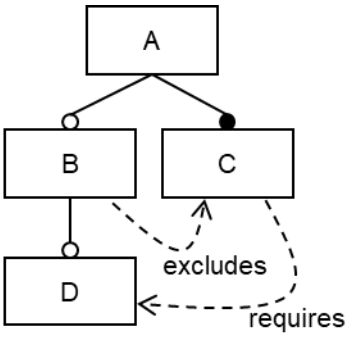
Teilaufgabe b) Konfigurationsanzahl (1,5 Punkte)

Wie viele verschiedene gültige Konfigurationen haben die folgenden Feature Diagramme? Berücksichtigen Sie auch die leere Konfiguration.

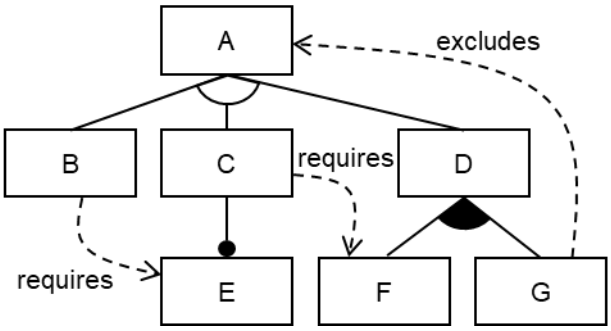
(1)



(2)



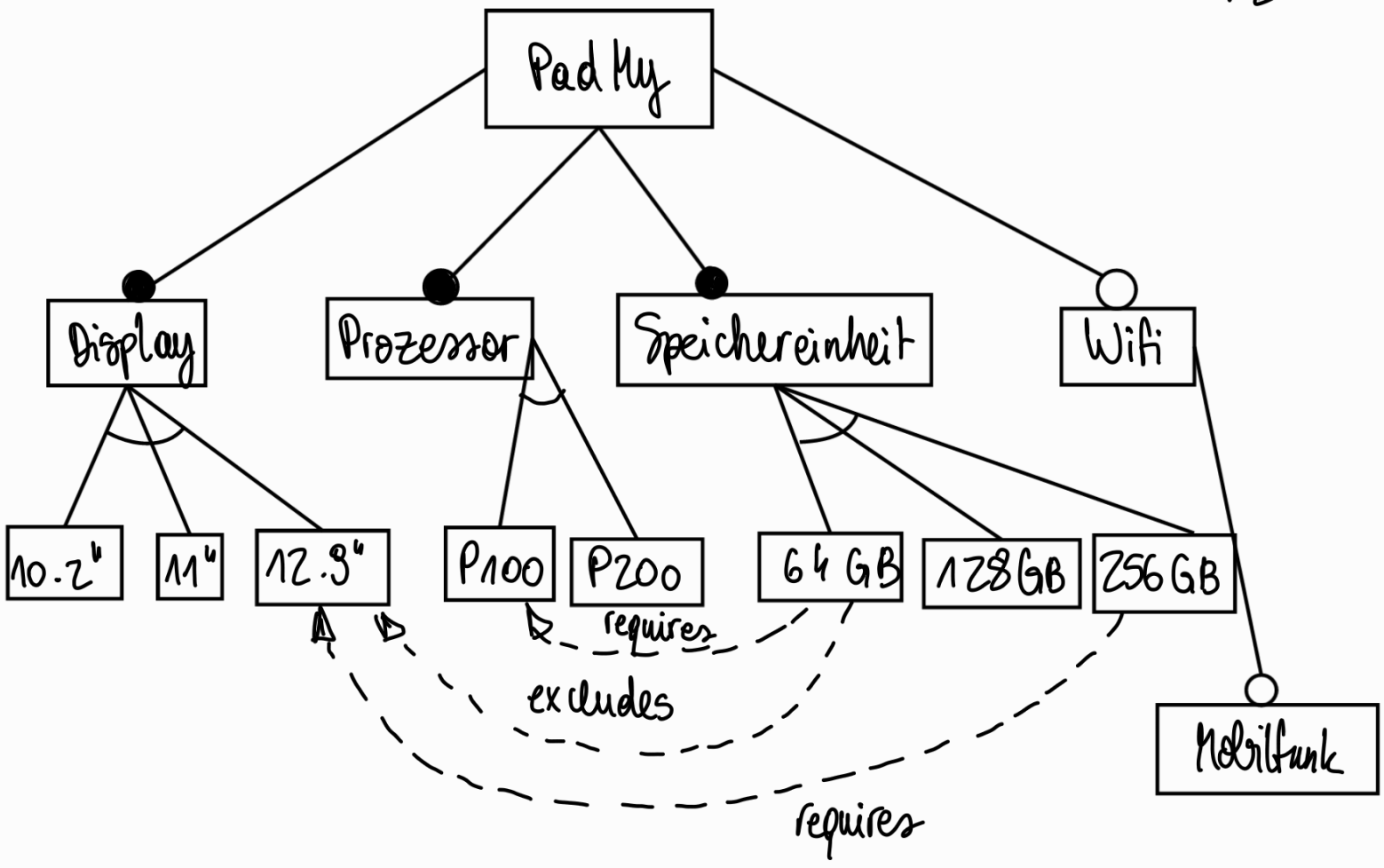
(3)



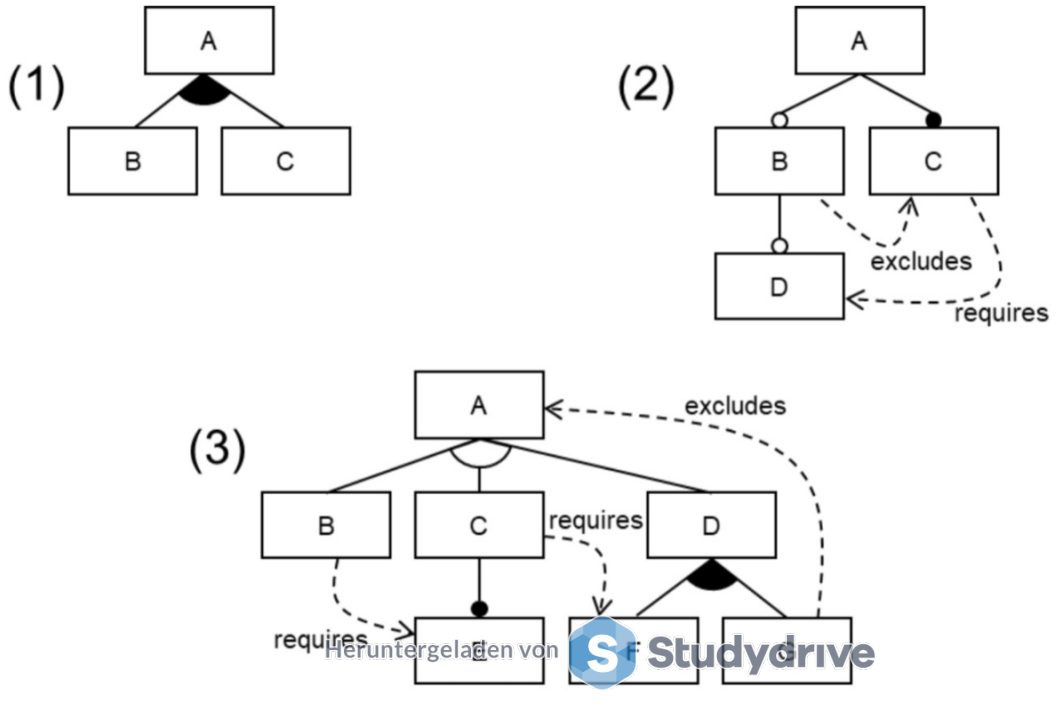
Aufgabe 9.2

a) Feature-Diagramm

FD



b) Konfigurationszahl



① $\emptyset$

$\{A, B\}$

$\{A, C\}$

$\{A, B, C\}$

4 Konfigurationen

② $\emptyset$

1 Konfig.

③ $\emptyset$

$\{A, D, F\}$

2 Konfigurationen

Lehrstuhl für Software Engineering
RWTH Aachen University
Prof. Bernhard Rumpe
Mathias Pfeiffer, M. Sc.
Hendrik Kausch, M. Sc.
Dipl.-Inform. Deni Raco

Softwaretechnik
Übung
WS 2021/22

Aufgabenblatt 10

Abgabe: 27.01.2022 10:30 Uhr (Donnerstag!)

Globalübung findet statt: 01.02.2022 10:30 Uhr (Dienstag)

Hinweis

Dieses Aufgabenblatt gibt ausschließlich Bonuspunkte. Zur Errechnung der Bonuspunkteanzahl für die Klausur werden als Basis 120 Punkte zugrunde gelegt. 120 Punkte sind durch die Bearbeitung der Aufgabenblätter 1-9 erreichbar. Die Punkte, die Sie mit diesem Aufgabenblatt erreichen werden Ihnen zusätzlich angerechnet. Sollten Sie also über alle Aufgabenblätter (1-10) hinweg mindestens 108 Punkte (90% von 120 Punkten) erreicht haben, dann wird Ihnen die volle Bonuspunkteanzahl angerechnet, falls Sie ohne die Berücksichtigung von Bonuspunkten mindestens die Note 4,0 erreichen.

Aufgabe 10.1 (10 Punkte)

Die Firma Carmpere ist neuer Mitbewerber auf dem Automobilmarkt und entwirft die Varianten für ihr erstes Auto. Jedes Auto enthält genau einen Motor. Carmpere bietet die Motorentypen Benzinmotor, Elektromotor und Hybridmotor an. Gegen einen Aufpreis können Autos der Firma Carmpere mit den folgenden speziellen Schließsystemen ausgestattet werden. Ein schlüsselloses Schließsystem bietet die Möglichkeit das Auto aufzuschließen, ohne den Schlüssel in das physikalische Schlüsselloch einzuführen. Ein weiteres Schließsystem bietet die Möglichkeit die Autos per Mobiltelefon aufzuschließen. Aus technischen Gründen kann ein schlüsselloses Schließsystem nicht verbaut werden, wenn ein Schließsystem verbaut ist, das es ermöglicht, das Auto per Mobiltelefon aufzuschließen. Eine weitere Art Schließsystem ermöglicht das Aufschließen des Autos per Fingerabdruckerkennung. Da die Fingerabdrücke auf Mobiltelefonen registriert werden, kann ein Schließsystem, das auf Fingerabdruckerkennung basiert, nur verbaut werden, wenn das Schließsystem verbaut ist, das es ermöglicht, das Auto per Mobiltelefon aufzuschließen.

Teilaufgabe a) Feature Diagramm erstellen (4 Punkte)

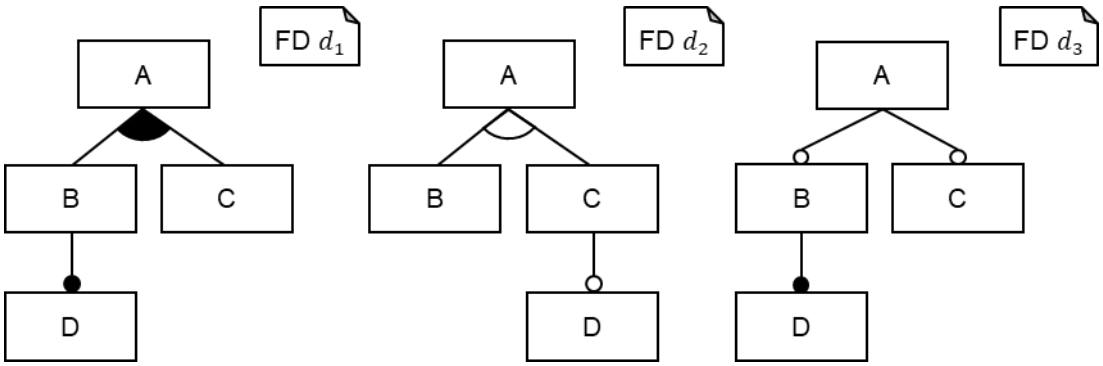
Modellieren Sie alle Konfigurationen des ersten Autos der Firma Carmpere mithilfe eines Feature Diagramms.

Teilaufgabe b) Semantische Differenz (6 Punkte)

Die *semantische Differenz* von einem Feature Diagramm d zu einem Feature Diagramm d' ist definiert als die Menge aller gültigen Konfigurationen von d, die keine gültigen Konfigurationen von d' sind. Ein Feature Diagramm d ist genau dann eine *Verfeinerung* eines Feature Diagramms d', wenn die semantische Differenz von d zu d' die leere Menge ist.

Verwenden Sie im Folgenden für Konfigurationen eine Mengenschreibweise. Beispielsweise repräsentiert die Menge {A, B, C} die Konfiguration, die exakt die Features A, B und C enthält. Die Menge {{A, B}, {B, C}} ist die Menge der zwei Konfigurationen {A, B} und {B, C}.

Im Folgenden sind die drei Feature Diagramme d_1, d_2 und d_3 dargestellt.



1) Berechnen Sie die semantische Differenz von d_1 zu d_2 (0,5 Punkte).
2) Berechnen Sie die semantische Differenz von d_2 zu d_3 (0,5 Punkte).
3) Berechnen Sie die semantische Differenz von d_1 zu d_3 (0,5 Punkte).

Beurteilen Sie für jede der folgenden Aussagen, ob sie wahr ist und begründen Sie Ihre Antwort in minimal einem und maximal drei Sätzen oder durch ein Gegenbeispiel.

4) Jedes Feature Diagramm ist eine Verfeinerung von sich selbst (0,5 Punkte).
5) Wenn ein Feature Diagramm d eine Verfeinerung von einem Feature Diagramm d' ist und ein Feature Diagramm d'' eine Verfeinerung von dem Feature Diagramm d' ist, dann ist d eine Verfeinerung von d'' (0,5 Punkte).
6) Der Schnitt der semantischen Differenz von einem Feature Diagramm d zu einem Feature Diagramm d' mit der semantischen Differenz von d' zu d ist leer (0,5 Punkte).

7) Wenn ein Feature Diagramm d eine Verfeinerung von einem Feature Diagramm d' ist, das Feature Diagramm d' eine Verfeinerung von einem Feature Diagramm d'' ist und d'' eine Verfeinerung von d ist, dann sind die Mengen der gültigen Konfigurationen von d und d'' identisch (1 Punkt).

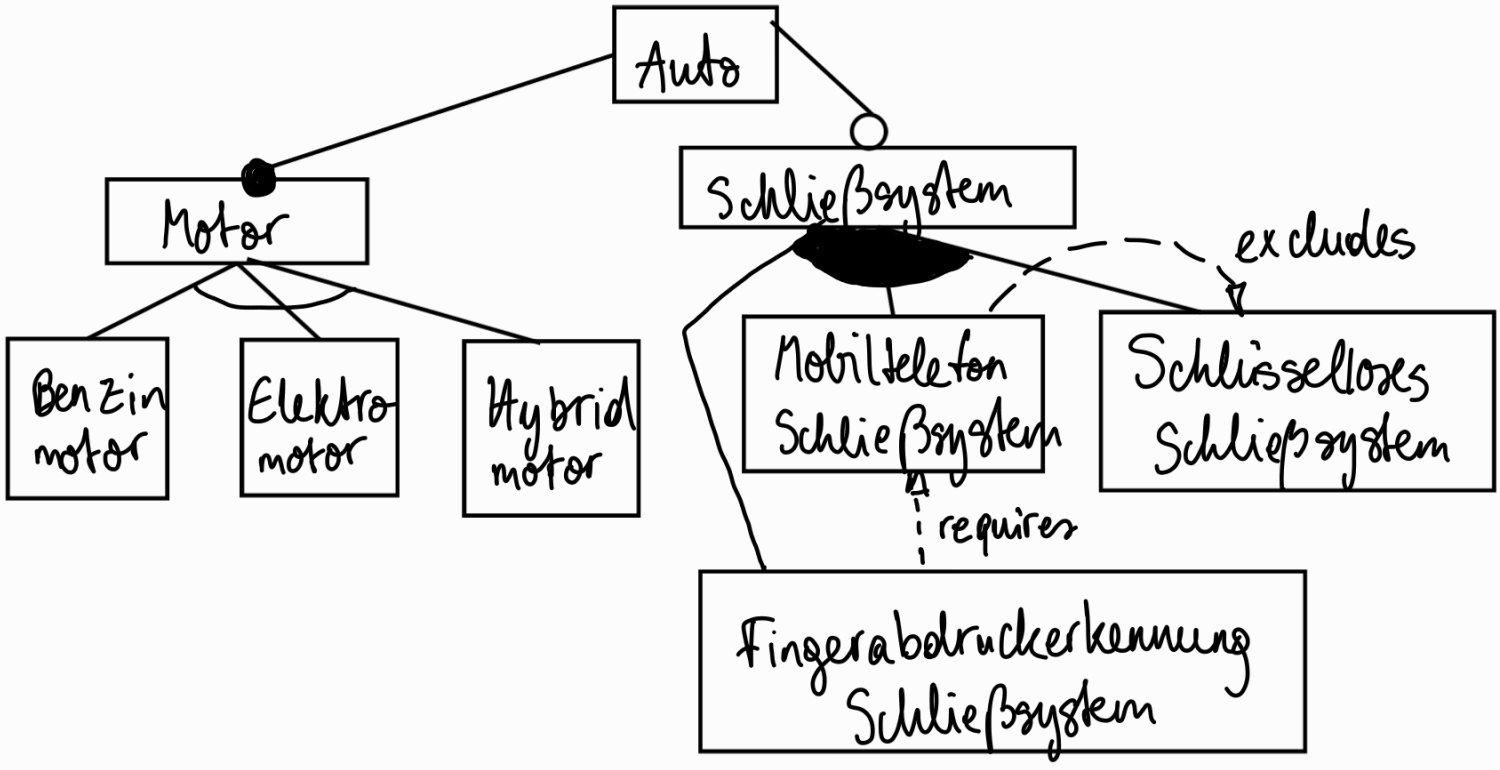
In späten Entwicklungsphasen sollte aus jeder gültigen Konfiguration eines Feature Diagramms ein Produkt hergeleitet werden können. Feature Diagramme entwickeln sich oft auch in späten Entwicklungsphasen weiter. In späten Entwicklungsphasen sollten die Produkte aller gültigen Konfiguration immer korrekt sein. Das heißt, keine Konfiguration sollte zu einem inkorrekten Produkt führen, z.B. zu nicht kompilierbarerem Code.

Angenommen, d und d' sind zwei Feature Diagramme und die semantischen Differenzen von d zu d' und von d' zu d sind bekannt. Das Feature Diagramm d' ist die Nachfolgeversion des Feature Diagramms d. Beurteilen Sie im Kontext der folgenden Szenarios inwiefern die bekannten semantischen Differenzen Entwicklern beim Überprüfen der Produkte der gültigen Konfigurationen von d' auf Korrektheit helfen können. Begründen Sie die Antworten.

8) Das Produkt jeder gültigen Konfiguration vom Feature Diagramm d ist korrekt und das Feature Diagramm d' ist eine Verfeinerung von d (1 Punkt).
9) Das Produkt jeder gültigen Konfiguration vom Feature Diagramm d ist korrekt und das Feature Diagramm d ist eine Verfeinerung von d' (1 Punkt).

Aufgabe 10.1

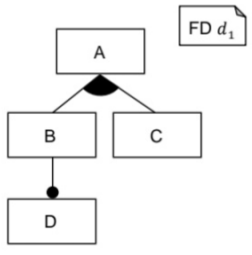
a) Feature Diagramm

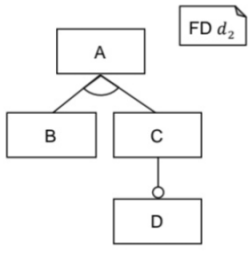


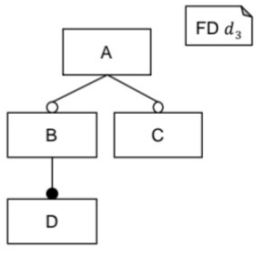
b) Semantische Differenz

- Im Folgenden sind die drei Feature Diagramme d_1, d_2 und d_3 dargestellt.

1. Berechnen Sie die semantische Differenz von d_1 zu d_2.
2. Berechnen Sie die semantische Differenz von d_2 zu d_3.
3. Berechnen Sie die semantische Differenz von d_1 zu d_3.







① $semDiff(d_1, d_2) = \{\{A, B, D\}, \{A, B, C, D\}\}$

② $semDiff(d_2, d_3) = \{\{A, B\}, \{A, C, D\}\}$

③ $semDiff(d_1, d_3) = \emptyset$

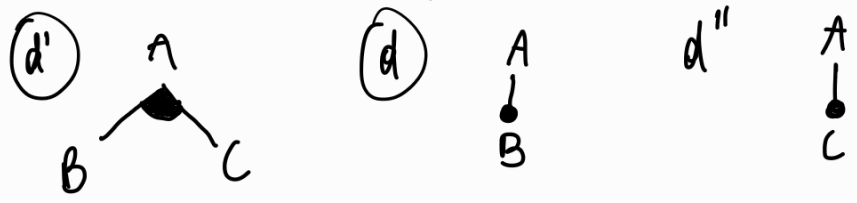
Beurteilen Sie für jede der folgenden Aussagen, ob sie wahr ist und begründen Sie Ihre Antwort in minimal einem und maximal drei Sätzen oder durch ein Gegenbeispiel.

4) Jedes Feature Diagramm ist eine Verfeinerung von sich selbst (0,5 Punkte).

⇒ wahr, weil die semantische Differenz von d zu sich selbst immer die leere Menge ist.

5) Wenn ein Feature Diagramm d eine Verfeinerung von einem Feature Diagramm d' ist und ein Feature Diagramm d'' eine Verfeinerung von dem Feature Diagramm d' ist, dann ist d eine Verfeinerung von d'' (0,5 Punkte).

⇒ falsch, Gegenbeispiel:



6) Der Schnitt der semantischen Differenz von einem Feature Diagramm d zu einem Feature Diagramm d' mit der semantischen Differenz von d' zu d ist leer (0,5 Punkte).

⇒ wahr: die semantische Differenz von d zu d': $d \setminus d'$

von d' zu d: $d' \setminus d$

$\Rightarrow (d \setminus d') \cap (d' \setminus d) = \emptyset$ ∎

7) Wenn ein Feature Diagramm d eine Verfeinerung von einem Feature Diagramm d' ist, das Feature Diagramm d' eine Verfeinerung von einem Feature Diagramm d'' ist und d'' eine Verfeinerung von d ist, dann sind die Mengen der gültigen Konfigurationen von d und d'' identisch (1 Punkt).

⇒ wahr:

d Verfeinerung von d': $d \subseteq d'$
d' Verfeinerung von d'': $d' \subseteq d''$
d'' Verfeinerung von d: $d'' \subseteq d$

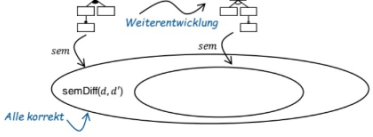
$\Rightarrow d \subseteq d''$ und $d = d'$ ∎

8) Das Produkt jeder gültigen Konfiguration vom Feature Diagramm d ist korrekt und das Feature Diagramm d' ist eine Verfeinerung von d (1 Punkt).

Der Entwickler kann direkt davon ausgehen, dass alle Produkte aller Konfigurationen von d' korrekt sind, da laut Annahme alle Produkte aller Konfigurationen von d korrekt sind und jede gültige Konfiguration von d' auch eine gültige Konfiguration von d ist.

- Annahme: semDiff$(d', d) = \emptyset$ und $\forall c \in sem(d): correct(c)$
  - $sem(d') \subseteq sem(d)$ und $\forall c \in sem(d): correct(c)$
  - $\forall c \in sem(d'): correct(c)$

FD d FD d'



9) Das Produkt jeder gültigen Konfiguration vom Feature Diagramm d ist korrekt und das Feature Diagramm d ist eine Verfeinerung von d' (1 Punkt).

Es genügt die Produkte der Konfigurationen in der semantischen Differenz von d' zu d auf Korrektheit zu überprüfen, da die Produkte der Konfigurationen von d', die auch Konfigurationen von d sind, laut Annahme korrekt sind.

- Annahme: semDiff$(d, d') = \emptyset$ und $\forall c \in sem(d): correct(c)$
  - $sem(d) \subseteq sem(d')$ und $\forall c \in sem(d): correct(c)$
  - $\forall c \in sem(d') \cap sem(d): correct(c)$