

Probe-Klausur

Grundlagen der OOP

Datum: 15.01.2026

Zeitvorgabe: 90 Minuten

Teilnehmer:

Name: _____

Inhaltsverzeichnis

Teil 1: Klassendiagramm 35 Punkte.....	3
Teil 2: Anwendungsfalldiagramm 20 Punkte.....	6
Teil 3: Aktivitätsdiagramm 15 Punkte.....	8
Teil 4: Zustandsdiagramm 20 Punkte.....	10
Teil 5: Sequenzdiagramm 10 Punkte.....	12

Ergebnis:

Gesamtpunktzahl: _____

Erreichte Punkte: _____

Note: _____

Korrektur-Datum: _____

Dozent: _____

Teil 1: Klassendiagramm

35 Punkte**Aufgabe 1:****12 Punkte**

Ein Streaming-Anbieter will seinen Kunden eine neue Smartphone-App anbieten.

- a) Es soll zunächst die Klasse *Film* für einzelne Filmobjekte entwickelt werden.

8 Punkte

Die Klasse *Film* soll Folgendes beinhalten:

- Die nur innerhalb der Klasse sichtbaren Instanzvariablen *id*, *titel*, *genre*, *jahr* und *dauer*
- Einen öffentlichen Konstruktor zur Initialisierung aller Instanzvariablen
- Beispielhaft für die Instanzvariable *titel* je eine öffentliche *Get*- und *Set*-Methode

Erstellen Sie das UML-Klassendiagramm für die Klasse *Film* und geben Sie dabei jeweils sinnvolle Datentypen an.

- b) Implementieren Sie in Pseudocode die *Get*-Methode für *titel*.

2 Punkte

- c) Implementieren Sie in Pseudocode die *Set*-Methode für *titel*.

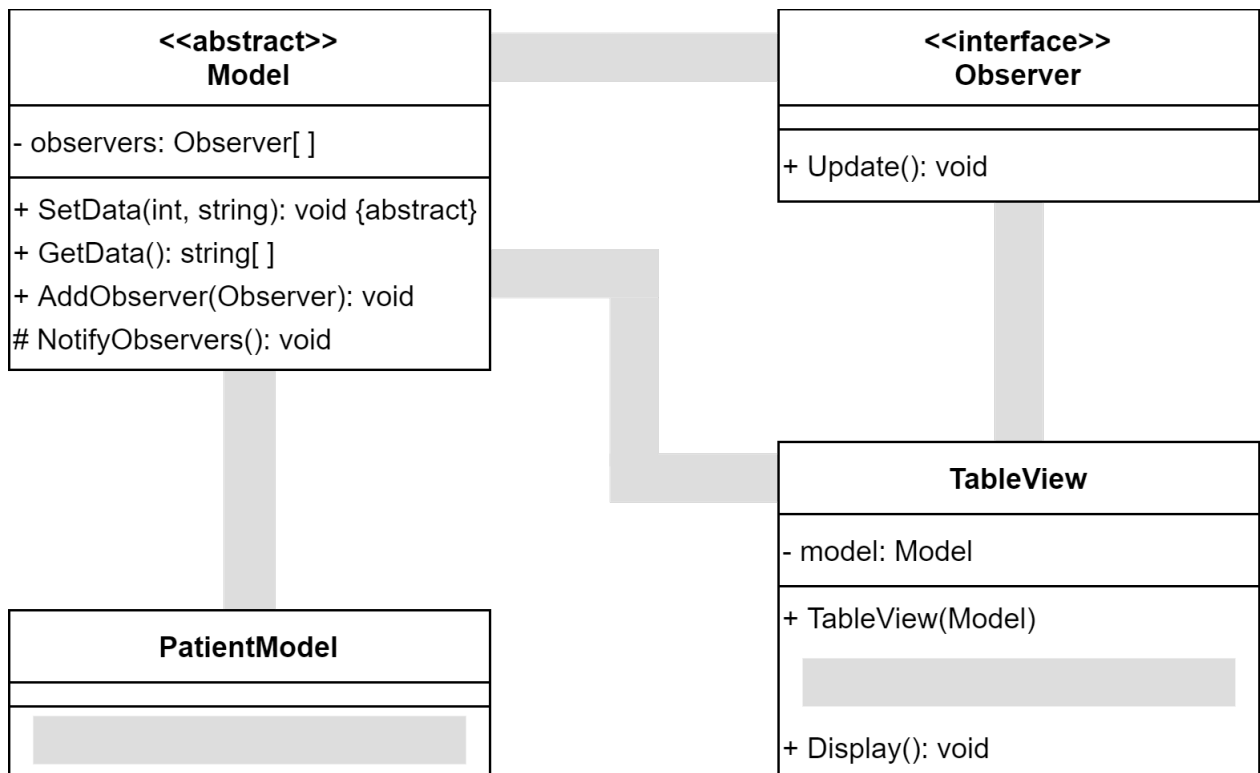
2 Punkte

Aufgabe 2:**13 Punkte**

Im Folgenden sehen Sie ein Beispiel für das Observer-Pattern. Es wird häufig eingesetzt, damit Anzeigen aktualisiert werden, sobald sich im Hintergrund die Daten geändert haben.

Dabei erbt die konkrete Klasse „*PatientModel*“ von der abstrakten Klasse „*Model*“. Die Klasse „*TableView*“ implementiert das Interface „*Observer*“.

- a) Ergänzen Sie im vorliegenden UML-Klassendiagramm alle Beziehungen und die noch fehlenden Methoden in den grau markierten Bereichen. 10 Punkte



- b) Der Konstruktor der Klasse „*TableView*“ muss zwei Aufgaben übernehmen:

1. Er initialisiert die *Model*-Referenz im Objekt mit dem übergebenen *Model*-Parameter.
2. Er registriert sich selbst als Observer. Dazu ruft er die Methode „*AddObserver()*“ auf dem ihm bekannten *Model*-Objekt auf und übergibt dabei seine eigene Referenz.

Geben Sie den Konstruktor im Pseudocode an.

3 Punkte

Aufgabe 3:**10 Punkte**

Die UML definiert verschiedene Typen von Beziehungen zwischen Klassen. Geben Sie in den folgenden Beispielen an, um welchen Beziehungstyp es sich handelt und zeichnen Sie diesen im Klassendiagramm ein.

Beschreibung	Beziehungstyp	Klassendiagramm
Eine Vorlesung wird von mehreren Studenten gebildet.		<div>Vorlesung</div> <div>Student</div>
Sowohl ein Professor als auch ein Assistent ist ein Mitarbeiter der Universität.		<div>Mitarbeiter</div> <div>Professor</div> <div>Assistent</div>
Eine Universität besteht aus mehreren Fachbereichen.		<div>Universität</div> <div>Fachbereich</div>
Ein Professor nutzt eine Tafel während seiner Vorlesung.		<div>Professor</div> <div>Tafel</div>
Ein Professor hat eine Menge von Büchern.		<div>Professor</div> <div>Buch</div>

Teil 2:

Anwendungsfalldiagramm

20 Punkte**Aufgabe 1:****2 Punkte**

Was stellt ein Akteur dar?

- A) Ein internes Softwaremodul
- B) Eine Klasse
- C) Eine externe Rolle oder Person
- D) Einen Systemzustand

Aufgabe 2:**2 Punkte**

Wofür steht die Beziehung «include»?

- A) Optionaler Ablauf
- B) Vererbung
- C) Zwingend eingebundener Anwendungsfall
- D) Alternative Ausführung

Aufgabe 3:**2 Punkte**

Wofür steht die Beziehung «extend»?

- A) Pflichtfunktion
- B) Bedingte Erweiterung eines Anwendungsfalls
- C) Klassenvererbung
- D) Parallele Ausführung

Aufgabe 4:**2 Punkte**

Können Akteure untereinander in Beziehung stehen?

- A) Nein
- B) Nur mit «include»
- C) Ja, durch Generalisierung
- D) Nur über Sequenzen

Aufgabe 5:**12 Punkte**

Die Versicherungsgesellschaft Knall & Peng AG benötigt ein neues Programm zur Betreuung ihrer Kunden. In einem ersten Schritt wurde festgelegt, dass folgende Punkte umgesetzt werden sollen:

- Fahrzeughalter und Sachbearbeiter schließen gemeinsam Verträge ab.
- Fahrzeughalter zahlen Prämien.
- Sachbearbeiter können Konditionen ändern.
- Sowohl Fahrzeughalter als auch Sachbearbeiter können Versicherungen kündigen.
- Fahrzeughalter können Schäden melden. Dies schließt eine weitere Bearbeitung durch Sachbearbeiter ein.

Erstellen Sie zu oben genanntem Sachverhalt ein Anwendungsfalldiagramm.

Teil 3:

Aktivitätsdiagramm

15 Punkte**Aufgabe 1:****15 Punkte**

Der Vorgang „Geld abheben an einem Geldautomaten“ wird wie folgt beschrieben:

- Der Kunde gibt seine EC-Karte ein.
- Der Geldautomat überprüft die EC-Karte.
 - Wenn die EC-Karte nicht gültig ist, wird sie einbehalten und der Vorgang abgebrochen.
 - Wenn die EC-Karte gültig ist, muss der Kunde seine PIN eingeben.
- Der Geldautomat überprüft die PIN.
 - Wenn die PIN nicht gültig ist, wird die Karte einbehalten und der Vorgang abgebrochen. Ein mehrfaches Eingeben der PIN ist nicht möglich!
 - Wenn die PIN gültig ist, gibt der Kunde den gewünschten Geldbetrag ein.
- Der Geldautomat aktualisiert das Konto und gibt die Karte wieder aus.
- Der Kunde entnimmt die EC-Karte.
- Anschließend stellt der Geldautomat das Geld bereit und der Kunde entnimmt es.

Stellen Sie den beschriebenen Vorgang auf der gegenüberliegenden Seite in einem UML-Aktivitätsdiagramm dar.

Kunde	Geldautomat

Teil 4:

Zustandsdiagramm

20 Punkte**Aufgabe 1:****5 Punkte**

Bewerten Sie die folgenden allgemeinen Aussagen zum Zustandsdiagramm:

Aussage**richtig falsch**

Das Zustandsdiagramm gehört zu den Verhaltensdiagrammen der UML.

☐ ☐

Der Übergang von einem Zustand zu einem anderen wird in der Regel durch ein Ereignis ausgelöst.

☐ ☐

Der Übergang von einem Zustand in einen anderen muss immer an eine Bedingung geknüpft sein.

☐ ☐

Ein Objekt kann sich in mehreren Zuständen gleichzeitig befinden.

☐ ☐

Zustände können optional ein internes Verhalten definieren.

☐ ☐**Aufgabe 2:****15 Punkte**

Der Prozess des Tankens an einer Zapfsäule wird wie folgt beschrieben:

Wenn das Auto an der Zapfsäule ankommt, ist diese im Zustand *freigegeben*, zeigt aber noch die Werte des letzten Tankvorgangs an. Sobald der Zapfhahn ausgehängt wird, wechselt die Zapfsäule in den Zustand *Tanken möglich*, dabei werden die Werte der Anzeige zurückgesetzt. Wenn der Hebel des Zapfhahns betätigt wird, fließt Kraftstoff, die Zapfsäule wechselt in den Zustand *Tanken*. Während die Zapfsäule sich in diesem Zustand befindet, wird die Anzeige laufend aktualisiert. Wird der Zapfhahn wieder eingehängt ist das Tanken beendet und die Zapfsäule ist im Zustand *gesperrt*. Wenn der Kunde an der Kasse bezahlt, gibt der Tankwart dabei die Zapfsäule wieder für den nächsten Tankvorgang frei. Am Ende des Arbeitstages, wenn alle Tankvorgänge beendet und bezahlt wurden, wird die Zapfsäule abgeschaltet.

Erstellen Sie anhand dieser Informationen auf der gegenüberliegenden Seite ein UML-Zustandsdiagramm für eine Zapfsäule.

Teil 5: Sequenzdiagramm

10 Punkte**Aufgabe 1:****10 Punkte**

In einem Buchhaltungs-Programm ist bereits die Klasse *Buchungsverwaltung* erstellt worden. Diese ermöglicht unter anderem die Durchführung von Stornierungen für einzelne Buchungen.

Klasse *Buchungsverwaltung*:

Buchungsverwaltung
+ stornieren(buchungsNr: String): Boolean + getBuchung(buchungsNr: String): Buchung + ermittleStornogebuehr(buchung: Buchung): Double + erstelleRechnung(buchung: Buchung, stornogebuehr: Double): void + loescheBuchung(buchungsNr: String): void

Der Vorgang der Stornierung wird wie folgt beschrieben:

- Der Vorgang wird durch den Aufruf der Methode *stornieren* eingeleitet. Dabei wird eine Buchungs-Nr. übergeben.
- Dann wird mit der Methode *getBuchung* überprüft, ob eine Buchung zu dieser Buchungs-Nr. vorliegt.
- Wenn die Methode *getBuchung* eine passende Buchung geliefert hat, werden mit der Methode *ermittleStornogebuehr* die Stornogebühren zu dieser Buchung ermittelt.
- Danach wird mit der Methode *erstelleRechnung* die Rechnung erstellt.
- Anschließend wird die Buchung mit der Methode *loescheBuchung* gelöscht und die Methode *stornieren* liefert als Ergebnis *true* an den Client zurück.
- Wenn die Methode *getBuchung* keine passende Buchung geliefert hat, liefert die Methode *stornieren* als Ergebnis *false* an den Client zurück.

Ergänzen Sie auf der nächsten Seite das vorgegebene Sequenzdiagramm passend zu diesem Ablauf. Beachten Sie dabei die ggf. erforderlichen Methoden-Parameter und auch mögliche Rückgabewerte.

