

Die Handlungsschritte 1 bis 5 beziehen sich auf die folgende Ausgangssituation:

Der gemeinnützige Verein Helferlein e. V. vermittelt Dienstleistungen von freiwilligen Helfern an hilfesuchende Bürger. Bislang wurden diese Dienstleistungen telefonisch kommuniziert. Nun möchte der Helferlein e. V. die Verwaltung der Dienstleistungen digitalisieren.

Sie sollen vier der folgenden fünf Aufgaben in diesem Projekt erledigen:

1. UML-Anwendungsfall-Diagramm entwickeln
2. Prozedur für Entgeltaufstellung schreiben
3. Objektorientiertes Modell für geschützten Methodenzugriff entwickeln
4. Relationales Datenbankmodell für Leistungserbringung anfertigen
5. SQL-Abfragen zu Mitgliedern, Angeboten und Bewertungen erstellen

Im Belegsatz finden Sie dazu:

- SQL
- Sequenzdiagramm
- Klassendiagramm
- USE-Case

1. Handlungsschritt (25 Punkte)

In der Webanwendung der Helferlein e. V. können Mitglieder ihre persönlichen Daten verwalten. Falls noch nicht erfolgt, ist ein Login erforderlich.

Jedes Mitglied kann Terminvorschläge für eine Dienstleistung abrufen. Es kann eine Dienstleistung buchen, muss dazu aber in jedem Fall Terminvorschläge für die Dienstleistung abrufen. Für die Buchung ist ebenfalls ein Login erforderlich.

Ein Administrator ist ein Mitglied, welches Dienstleistungsarten verwalten kann. Auch hier ist ein Login erforderlich, falls dieser noch nicht durchgeführt wurde.

Für einen Gast gibt es ebenfalls die Möglichkeit, Terminvorschläge für eine Dienstleistung abzurufen. Außerdem kann sich ein Guest als Mitglied registrieren.

Erstellen Sie auf der gegenüberliegenden Seite ein Anwendungsfalldiagramm, welches die beschriebenen Anwendungsfälle und die erforderlichen Akteure darstellt.

2. Handlungsschritt (25 Punkte)

Korrekturrand

Für den Helferlein e. V. soll eine Prozedur entwickelt werden, welche die erbrachten Leistungen und die aufsummierten Entgelte auflistet. Die Daten sind in einem Journal gespeichert.

Journal (Beispiel)

Datum	MitgliedID	LeistungID	AnzahlStunden
01.04.2021	100062	100076	2
11.04.2021	100062	100076	3
10.04.2021	100062	500123	1
13.04.2021	201235	200234	1
14.04.2021	201235	200234	1
07.04.2021	201235	200356	1

Das Journal ist nach MitgliedID und bei gleicher MitgliedID nach LeistungID sortiert.

Die Ausgabeliste soll wie folgt aufgebaut sein:

Nr	MitgliedID	Name	Vorname	LeistungID	Leistung	AnzahlStunden	Stundensatz	Gesamt
1	100062	Clausen	Jens	100076	Gießen	5	6,00	30,00
2	100062	Clausen	Jens	500123	Umgraben	1	6,00	6,00
						Summe	36,00	
1	201235	Rader	Sabrina	200234	Hausputz	2	6,00	12,00
2	201235	Rader	Sabrina	200356	Einkauf	1	6,00	6,00
						Summe	18,00	
						Gesamtsumme	54,00	

Folgende Funktionen sollen verwendet werden:

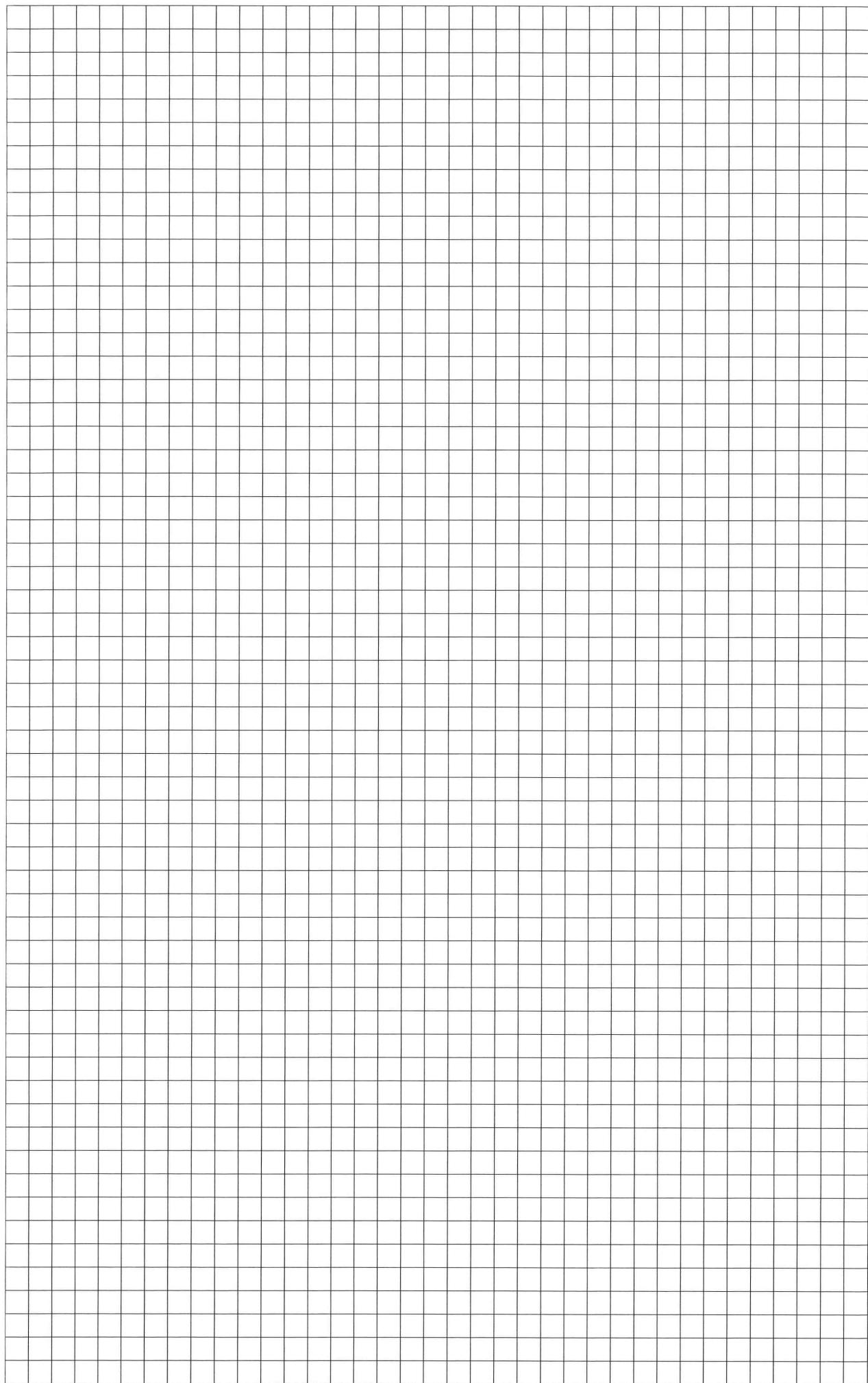
hole_satz() : String	Liest den nächsten Datensatz der Journal-Tabelle in eine Zeichenkette ein. Kann kein Satz mehr gelesen werden, liefert die Funktion den String "".
lese_m_id(satz : String) : Integer	Ermittelt die <i>MitgliedID</i> aus <i>satz</i> .
lese_l_id(satz : String) : Integer	Ermittelt die <i>LeistungID</i> aus <i>satz</i> .
lese_anz_std(satz : String) : Integer	Ermittelt die Anzahl der Stunden aus <i>satz</i> .
schreibe_kopf()	Schreibt die Kopfzeile der Positionen-Tabelle
schreibe_daten(nr : Integer, mitgliedid: Integer, leistungid : Integer, anzahlstunden : Integer, stundensatz: Double, summe: Double)	Schreibt eine Datenzeile in der geforderten Darstellung. <i>Name</i> , <i>Vorname</i> und <i>Leistung</i> werden automatisch aus <i>mitgliedid</i> und <i>leistungid</i> ermittelt.
schreibe_summe(summe : Double)	Schreibt die (berechnete) Summe für ein Mitglied
schreibe_gsumme(gsumme : Double)	Schreibt die (berechnete) Gesamtsumme des Journals

Entwickeln Sie auf der gegenüberliegenden Seite einen Algorithmus für die Prozedur *erstelle_liste(stundensatz: Double)*, der die Liste entsprechend der Vorgabe schreibt und dazu die entsprechenden Werte berechnet. Verwendete Variablen müssen nicht deklariert werden.

Stellen Sie den Algorithmus in Pseudocode, einem Struktogramm oder einem PAP dar.

```
erstelle_liste(stundensatz : Double)
```

Korrekturrand



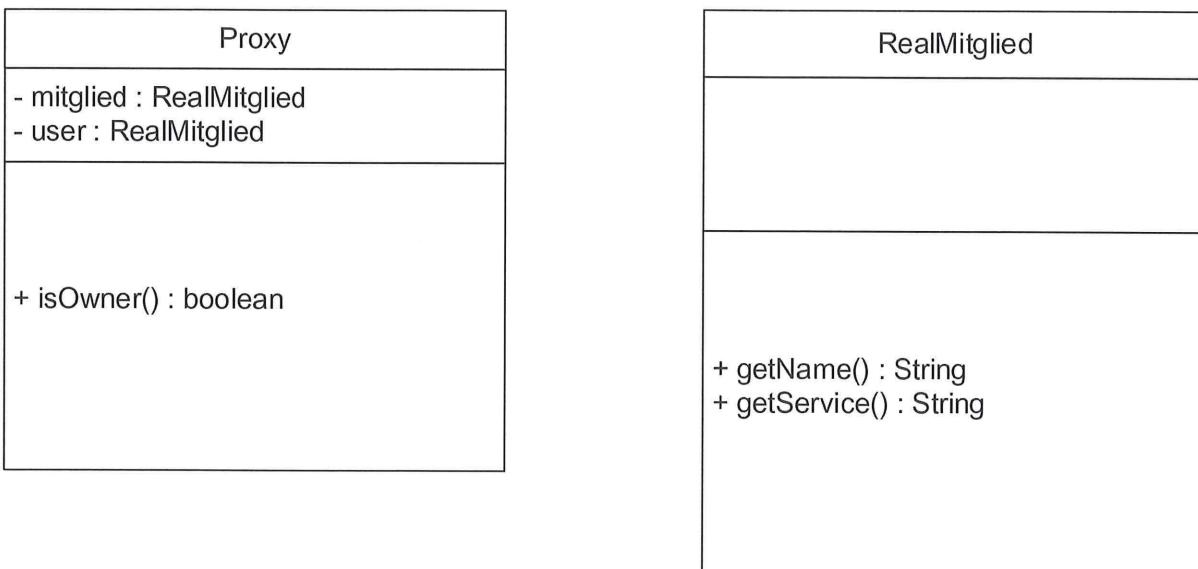
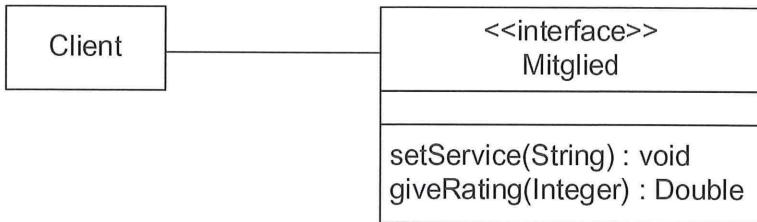
3. Handlungsschritt (25 Punkte)

Korrekturrand

Alle Mitglieder des Helferlein e. V. sollen über eine App nur ihre eigenen Serviceangebote editieren und nur für andere Mitglieder eine Bewertung von 1 bis 10 abgeben können. Um dies zu gewährleisten, wird der Zugriff auf die Methoden „setService“ und „giveRating“ über die Klasse Proxy gesteuert. Die Methode „isOwner“ vergleicht die Objekte „mitglied“ und „user“.

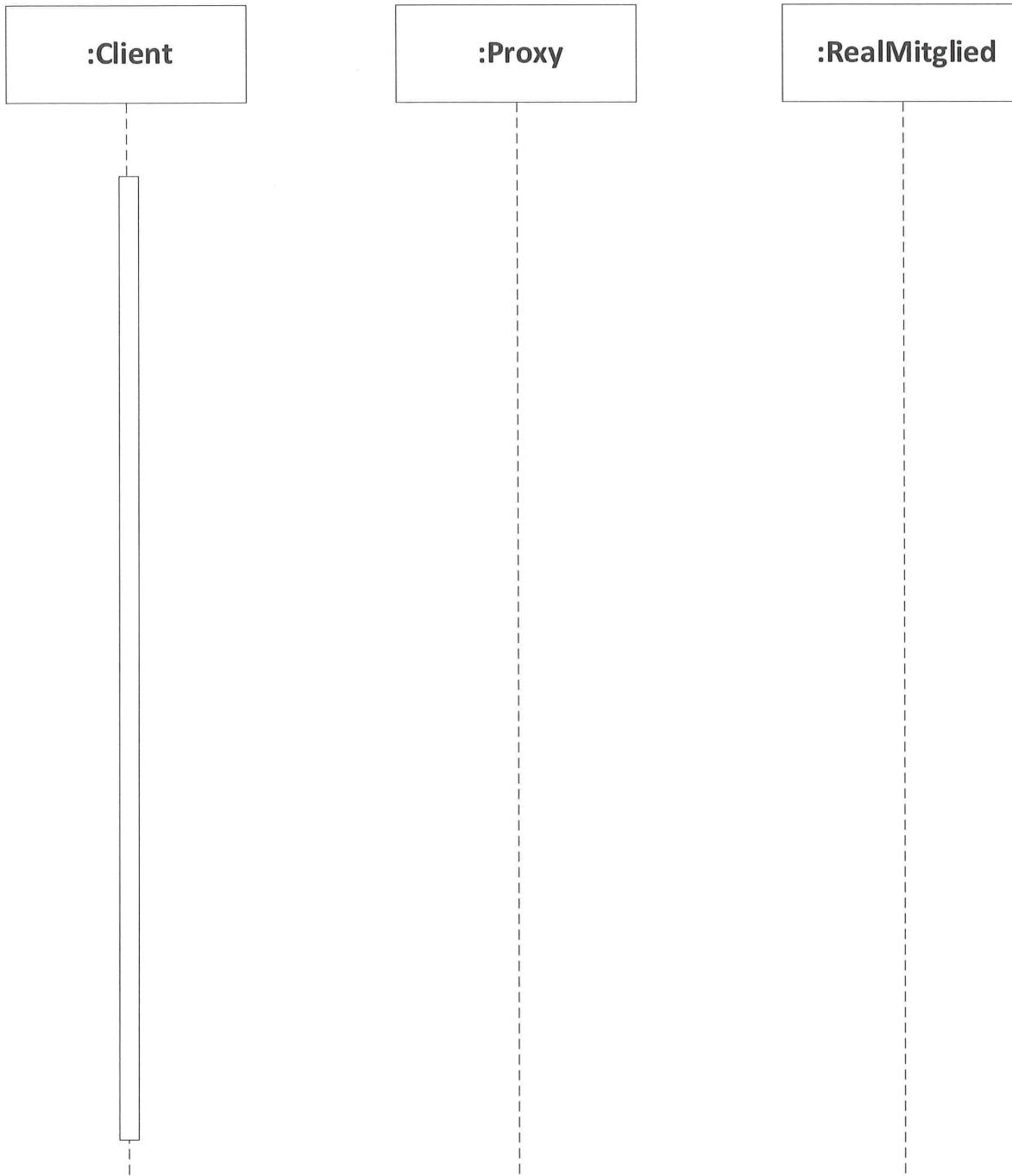
a) Ergänzen Sie das folgende unvollständige UML-Klassendiagramm nach den folgenden Vorgaben. 10 Punkte

- Die Klasse „RealMitglied“ soll die nur klassenintern sichtbaren Instanzvariablen „name“, „service“, „rating“, und „countRatings“ beinhalten.
- Der öffentliche Konstruktor der Klasse „RealMitglied“ soll zwei Übergabeparameter haben, mit denen „name“ und „service“ initialisiert werden.
- Der öffentliche Konstruktor der Klasse „Proxy“ soll zwei „RealMitglied“-Objekte übergeben bekommen und diese mit den Referenzen „mitglied“ und „user“ verknüpfen.
- Die Klassen „Proxy“ und „RealMitglied“ sollen beide das Interface Mitglied implementieren.
- Zeichnen Sie alle Klassenbeziehungen ein:



b) Die Funktionsweise des Zugriffproxies soll auf der Folgeseite mit einem UML-Sequenzdiagramm verdeutlicht werden: 10 Punkte

- Der Client ruft auf dem Proxyobjekt die Methode „giveRating“ auf.
- Die Methode „giveRating“ überprüft mittels der Methode „isOwner“, ob das bewertete RealMitglied „mitglied“ und der Bewerter „user“ ein und dieselbe Person sind.
- Im Ja-Fall gibt „giveRating“ den Wert -1.0 an den Client zurück.
- Im Nein-Fall wird die Methode „giveRating“ des RealMitglied-Objekts aufgerufen, welche die abgegebene Bewertung hinzufügt und die durchschnittliche Bewertung zurückgibt.
- Die durchschnittliche Bewertung wird an den Client weitergereicht.



- c) Der Client soll nicht mehr über den Konstruktor eine Instanz der Klasse Proxy erzeugen, sondern über eine statische Fabrikmethode.

Implementieren Sie in Pseudocode diese Methode der Klasse Proxy mit dem Namen „getInstance“.

5 Punkte

4. Handlungsschritt (25 Punkte)

Korrekturrand

Die Datenbank für die Leistungserbringung soll nach dem folgenden Pflichtenheft modelliert werden.

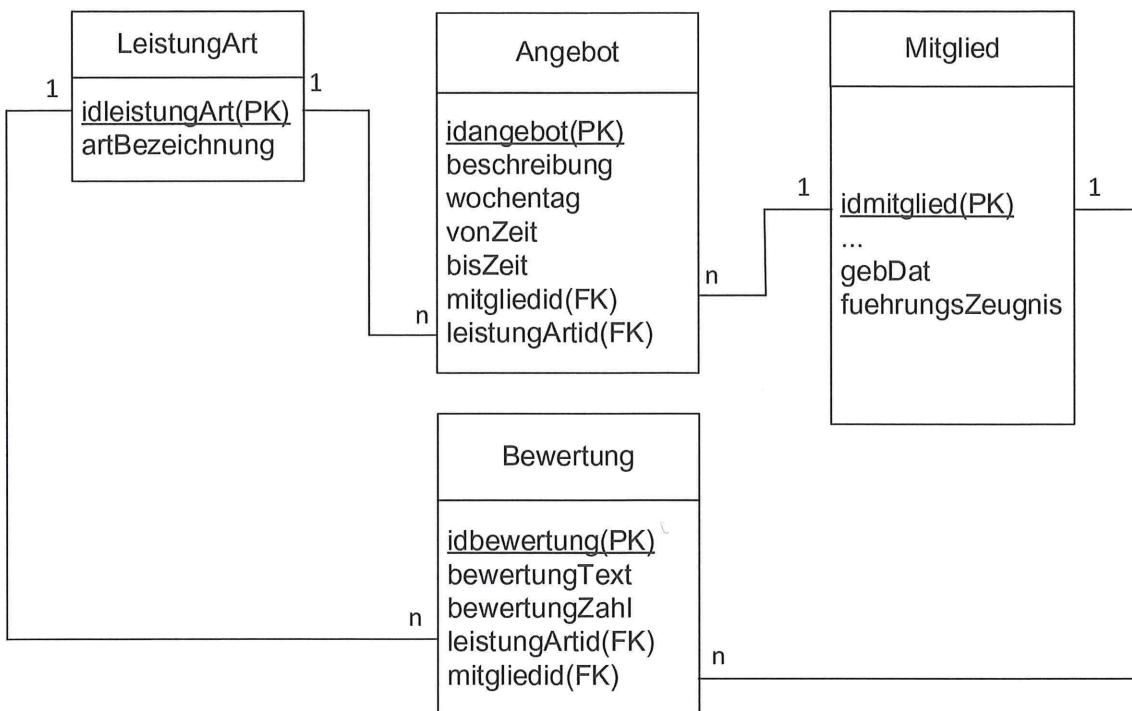
- Für die Mitglieder sollen folgende Attribute hinterlegt sein: vollständiger Name und vollständige Adresse, Telefonnummer, E-Mail-Adresse.
- Für jedes Mitglied sollen seine aufsummierten Einnahmen und Ausgaben in einer eigenen Tabelle verwaltet werden.
- Für die Leistung ist eine Beschreibung zu erfassen.
- Für die Leistung kann optional eine Ausrüstung erforderlich sein.
- Leistungsarten (z. B. Kinderbetreuung, Fahrdienst, Gartenarbeit) sowie Ausrüstung (z. B. Werkzeug, Rasenmäher) sollen jeweils in einer eigenen Tabelle hinterlegt sein.
- Jedes Mitglied kann beliebige Leistungen als Leistungsnehmer in Anspruch nehmen.
- Jedes Mitglied kann beliebige Leistungen als Leistungsgeber erbringen.
- Für jede erbrachte Leistung ist der Leistungsbereich und die Leistungsdauer in vollen Stunden zu erfassen.

Erstellen Sie auf der gegenüberliegenden Seite ein relationales Datenbankmodell in der dritten Normalform. Kennzeichnen Sie Schlüsselattribute mit PK für Primärschlüssel und FK für Fremdschlüssel. Geben Sie alle Beziehungen mit ihren Kardinalitäten an. Geben Sie den Tabellen und Attributen selbsterklärende Namen. Adressdaten dürfen redundant sein.

5. Handlungsschritt (25 Punkte)

Korrekturrand

Informationen zu Mitgliedern, ihren jeweiligen Angeboten und Bewertungen, stehen in folgender Datenbank zur Verfügung:



Formulieren Sie SQL-Abfragen für die nachstehenden Aufgabenstellungen.

- a) Geben Sie alle Attribute des jüngsten Mitglieds aus. 4 Punkte

- b) Ermitteln Sie eine Mitgliederliste aufsteigend sortiert nach der durchschnittlichen Bewertung für die Leistungsart „Kinderbetreuung“. 6 Punkte

idmitglied	mitgliedName	Durchschnitt
3	Müller	2.1
2	Maier	3.0
25	Spielmann	4.5
...		

c) Erstellen Sie eine Angebotsliste, die alle Mitglieder und die entsprechende Leistungsart des Angebots ausgibt, welche donnerstags von 14:00 Uhr bis 16:00 Uhr zur Verfügung stehen.

7 Punkte

Korrekturrand

Beispielausgabe:

idMitglied	mitgliedName	artBezeichnung	wochentag	vonZeit	bisZeit
4	Hauser	Gartenarbeit	Donnerstag	14:00	16:00
4	Hauser	Hausarbeit	Donnerstag	14:00	16:00
2	Maier	Kinderbetreuung	Donnerstag	14:00	16:00
...					

d) Erstellen Sie eine neue Tabelle MitgliedArchiv. Transferieren Sie alle Mitglieder, die kein Angebot eingestellt haben, in diese Tabelle. Löschen Sie diese inaktiven Mitglieder aus der Tabelle Mitglied.

8 Punkte

PRÜFUNGSZEIT – NICHT BESTANDTEIL DER PRÜFUNG!

Wie beurteilen Sie nach der Bearbeitung der Aufgaben die zur Verfügung stehende Prüfungszeit?

Sie hätte kürzer sein können.

Sie war angemessen.

Sie hätte länger sein müssen.

Belegsatz

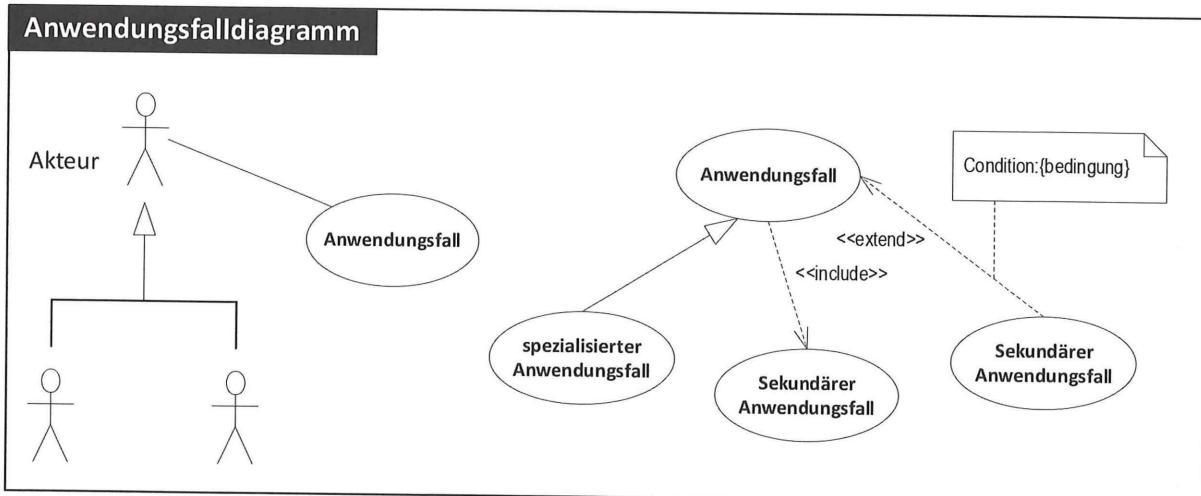
Fachinformatiker Anwendungsentwicklung
Fachinformatikerin Anwendungsentwicklung
1196

1 Ganzheitliche Aufgabe I Fachqualifikationen

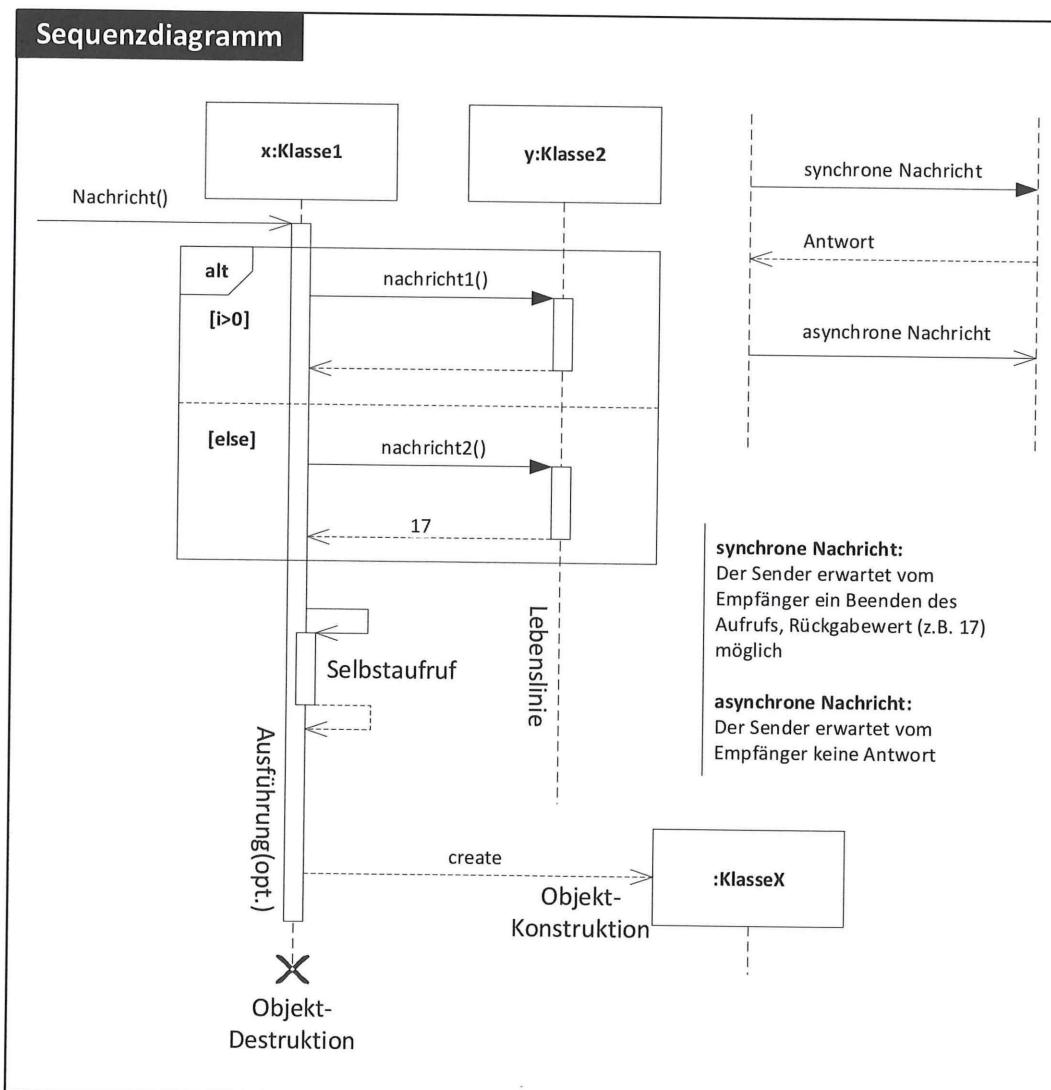
Inhalt

UML-Anwendungsfalldiagramm	Seite 2
UML-Sequenzdiagramm	Seite 2
UML-Klassendiagramm	Seite 3
SQL-Syntax (Auszug)	Seite 4/5

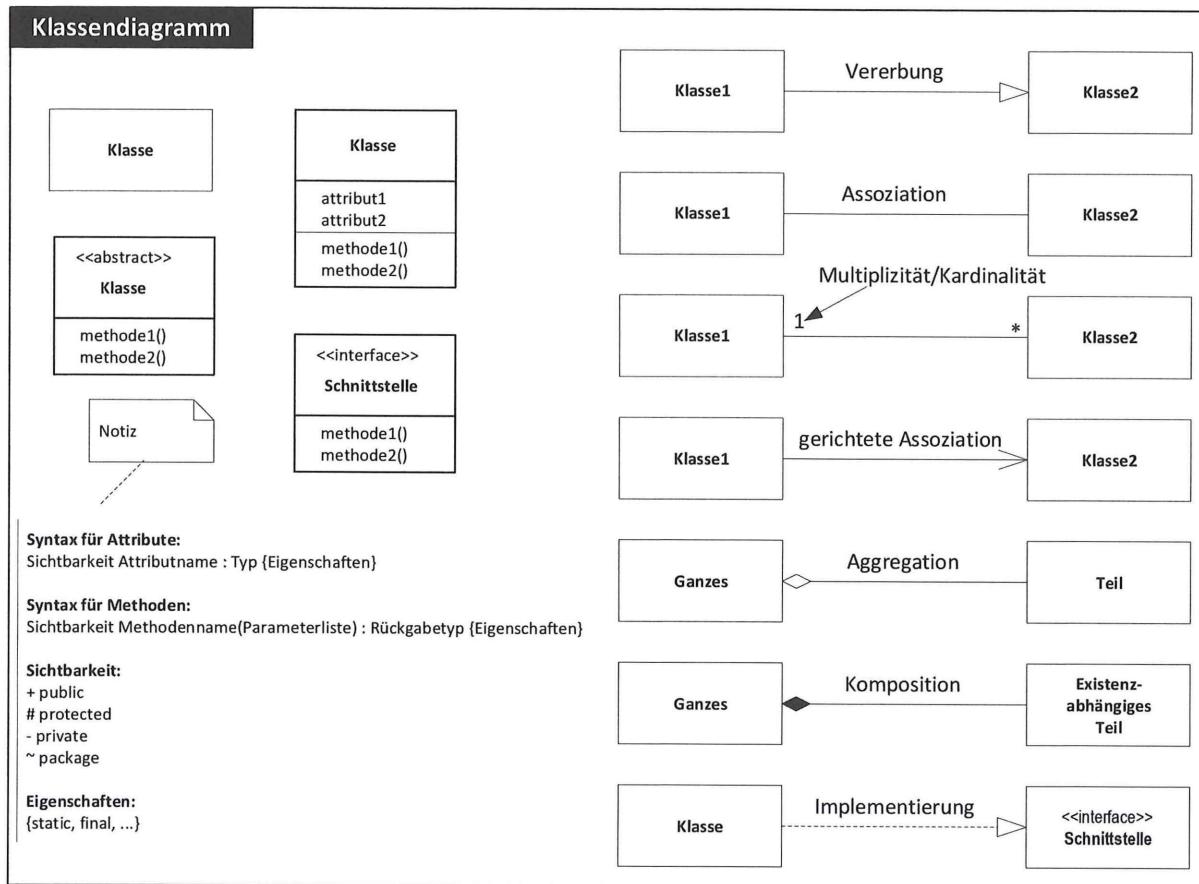
UML-Anwendungsfalldiagramm



UML-Sequenzdiagramm



UML-Klassendiagramm



SQL-Syntax (Auszug)

Syntax	Beschreibung
Tabelle	
CREATE TABLE Tabellenname(Spaltenname < DATENTYP >, Primärschlüssel, Fremdschlüssel)	Erzeugt eine neue leere Tabelle mit der beschriebenen Struktur
ALTER TABLE Tabellenname ADD COLUMN Spaltenname Datentyp DROP COLUMN Spaltenname Datentyp ADD FOREIGN KEY (Spaltenname) REFERENCES Tabellenname(Primärschlüsselspaltenname)	Änderungen an einer Tabelle: Hinzufügen einer Spalte Entfernen einer Spalte Definiert eine Spalte als Fremdschlüssel
CHARACTER	Textdatentyp
DECIMAL	Numerischer Datentyp (Festkommazahl)
DOUBLE	Numerischer Datentyp (Doppelte Präzision)
INTEGER	Numerischer Datentyp (Ganzzahl)
DATE	Datum (Format DD.MM.YYYY)
PRIMARY KEY (Spaltenname)	Erstellung eines Primärschlüssels
FOREIGN KEY (Spaltenname) REFERENCES Tabellenname(Primärschlüsselspaltenname)	Erstellung einer Fremdschlüssel-Beziehung
DROP TABLE Tabellenname	Löscht eine Tabelle
Befehle, Klauseln, Attribute	
SELECT * Spaltenname1 [, Spaltenname2, ...]	Wählt die Spalten einer oder mehrerer Tabellen, deren Inhalte in die Liste aufgenommen werden sollen; alle Spalten (*) oder die namentlich aufgeführten
FROM	Name der Tabelle oder Namen der Tabellen, aus denen die Daten der Ausgabe stammen sollen
SELECT ... (SELECT ... FROM ... WHERE ...) AS xyz FROM ... WHERE ...	Unterabfrage, die in eine äußere SELECT-Anweisung geschachtelt ist. Das Ergebnis der Unterabfrage wird im Spaltenausdruck (z. B. hier: xyz) ausgegeben.
SELECT DISTINCT	Eliminiert Redundanzen, die in einer Tabellen auftreten können, Werte werden jeweils nur einmal angezeigt.
INNER JOIN	Liefert nur die Datensätze zweier Tabellen, die gleiche Datenwerte enthalten
LEFT JOIN / LEFT OUTER JOIN	Liefert von der erstgenannten (linken) Tabelle alle Datensätze und von der zweiten Tabelle jene, deren Datenwerte mit denen der ersten Tabelle übereinstimmen
RIGHT JOIN / RIGHT OUTER JOIN	Liefert von der zweiten (rechten) Tabelle alle Datensätze und von der ersten Tabelle jene, deren Datenwerte mit denen der zweiten Tabelle übereinstimmen
FULL JOIN	Liefert aus beiden Tabellen jeweils alle Datensätze
WHERE	Bedingung, nach der Datensätze ausgewählt werden sollen
WHERE EXISTS (subquery) WHERE NOT EXISTS (subquery)	Die Bedingungen EXISTS prüft, ob die Suchbedingung einer Unterabfrage mindestens eine Zeile zurückliefert. NOT EXIST negiert die Bedingung.
GROUP BY Spaltenname1 [,Spaltenname2, ...]	Gruppierung (Aggregation) nach Inhalt des genannten Feldes
ORDER BY Spaltenname1 [,Spaltenname2, ...] ASC DESC	Sortierung nach Inhalt des genannten Feldes oder der genannten Felder ASC: aufsteigend; DESC: absteigend
Syntax	Beschreibung
Datenmanipulation	
DELETE FROM Tabellenname	Löschen von Datensätzen in der genannten Tabelle
UPDATE Tabellenname SET	Aktualisiert Daten in Feldern einer Tabelle
INSERT INTO Tabellenname VALUES (Wert für Spalte 1 [, Wert für Spalte 2, ...]) oder SELECT ... FROM ... WHERE	Fügt Datensätze in die genannte Tabelle, die entweder mit festen Werten belegt oder Ergebnis eines SELECT-Befehls sind

Aggregatfunktionen	
AVG(Spaltenname)	Ermittelt das arithmetische Mittel aller Werte im angegebenen Feld
COUNT(Spaltenname *)	Ermittelt die Anzahl der Datensätze mit Nicht-NULL-Werten im angegebenen Feld oder alle Datensätze der Tabelle (dann mit Operator *)
SUM(Spaltenname Formel)	Ermittelt die Summe aller Werte im angegebenen Feld oder der Formelergebnisse
MIN(Spaltenname Formel)	Ermittelt den kleinsten aller Werte im angegebenen Feld
MAX (Spaltenname Formel)	Ermittelt den größten aller Werte im angegebenen Feld
Funktionen	
LEFT(Zeichenkette, Anzahlzeichen)	Liefert Anzahlzeichen der Zeichenkette von links.
RIGHT(Zeichenkette, Anzahlzeichen)	Liefert Anzahlzeichen der Zeichenkette von rechts.
CURRENT	Liefert das aktuelle Datum mit der aktuellen Uhrzeit
CONVERT(time,[DatumZeit])	Liefert die Uhrzeit aus einer DatumZeit-Angabe
DATE(Wert)	Wandelt einen Wert in ein Datum um
DAY(Datum)	Liefert den Tag des Monats aus dem angegebenen Datum
MONTH(Datum)	Liefert den Monat aus dem angegebenen Datum
TODAY	Liefert das aktuelle Datum
WEEKDAY(Datum)	Liefert den Tag der Woche aus dem angegebenen Datum
YEAR(Datum)	Liefert das Jahr aus dem angegebenen Datum
DATEADD(Datumsteil, Intervall, Datum)	Fügt einem Datum ein Intervall (ausgedrückt in den unter Datumsteil angegebenen Einheiten) hinzu
DATEDIFF(Datumsteil, Anfangsdatum, Enddatum) Datumsteile: DAY, MONTH, YEAR	Liefert Enddatum-Startdatum (ausgedrückt in den unter Datumsteil angegebenen Einheiten)
Operatoren	
AND	Logisches UND
LIKE	Überprüfung von Textattributen auf Gleichheit, Verwendung von Platzhaltern möglich.
NOT	Logische Negation
OR	Logisches ODER
=	Test auf Gleichheit
>, >=, <, <=, < >	Test auf Ungleichheit
*	Multiplikation
/	Division
+	Addition, positives Vorzeichen
-	Subtraktion, negatives Vorzeichen

Stand 2018-03-29