Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2016

46116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Softwaretechnologie/Datenbanksysteme**

Anzahl der gestellten Themen (Aufgaben): **2**

Anzahl der Druckseiten dieser Vorlage: **14**

Bitte wenden!

Thema Nr. 1

(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1

Aufgabe 1: Begriffe und Konzepte

a) Was ist der Unterschied zwischen einem Software-Fault und einem Software-Failure?

b) Was ist der Unterschied zwischen Testing und Model-Checking? Welchen Vorteil hat Testing gegenüber Model-Checking bzw. umgekehrt?

c) Eine Unternehmensberatung stellt fest: „Die Wartungskosten Ihrer Projekte machen bis zu 70% der gesamten Projektkosten aus!“ Diskutieren Sie, ob dieser hohe Wert grundsätzlich negativ zu bewerten ist.

d) Welche Phasen werden in jedem Prozess-Modell der Softwareentwicklung durchlaufen?

e) Nennen Sie zwei Vorteile und zwei Nachteile des Wasserfallmodells, jeweils mit Begründung.

f) Nennen Sie zwei Vorteile und zwei Nachteile der agilen Softwareentwicklung, jeweils mit Begründung.

g) Nennen Sie zwei Maßnahmen, mit denen agile Methoden versuchen, eine frühe Validierung zu erreichen. Erläutern Sie zusätzlich kurz, warum eine frühe Validierung von Vorteil ist.

h) Bewerten Sie folgende Aussage eines Projektmanagers: „Falls wir in Verzug geraten, dann können wir jederzeit neue Programmierer hinzuholen, und können die Deadline doch noch einhalten.“

Fortsetzung nächste Seite!

Aufgabe 2: Entwurfsmuster

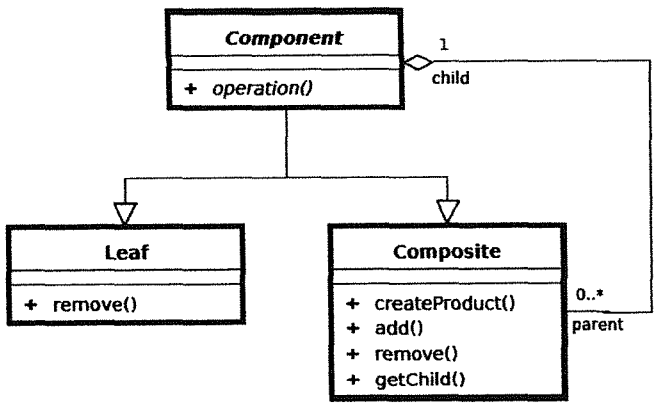
a) **Singleton:** Im Folgenden sehen Sie eine unvollständige Klassendefinition der Klasse Singleton des gleichnamigen Entwurfsmusters. Unter der Klasse befindet sich ein Aufruf, der der Variablen s die Instanz der Klasse zuweist.

Geben Sie den vervollständigten Quelltext der Klassendefinition und des Aufrufs an (nicht auf dem Aufgabenblatt vervollständigen).

```
public class Singleton {
  [      ]static final Singleton INSTANCE = new [      ];

  [      ]Singleton() {}

  [      ][      ]Singleton getInstance() {
      return [      ];
    }
}

Singleton s = [                    ];
```

b) **Fehlersuche und Identifikation:** Im Folgenden sehen Sie ein UML-Diagramm eines Entwurfsmusters mit Fehlern.

Geben Sie ein korrektes UML-Diagramm für das gemeinte Entwurfsmuster an und nennen Sie den Namen des nun korrekten Entwurfsmusters (nicht auf dem Aufgabenblatt berichtigen).



Fortsetzung nächste Seite!

Aufgabe 3: „OOA/OOD - UML“

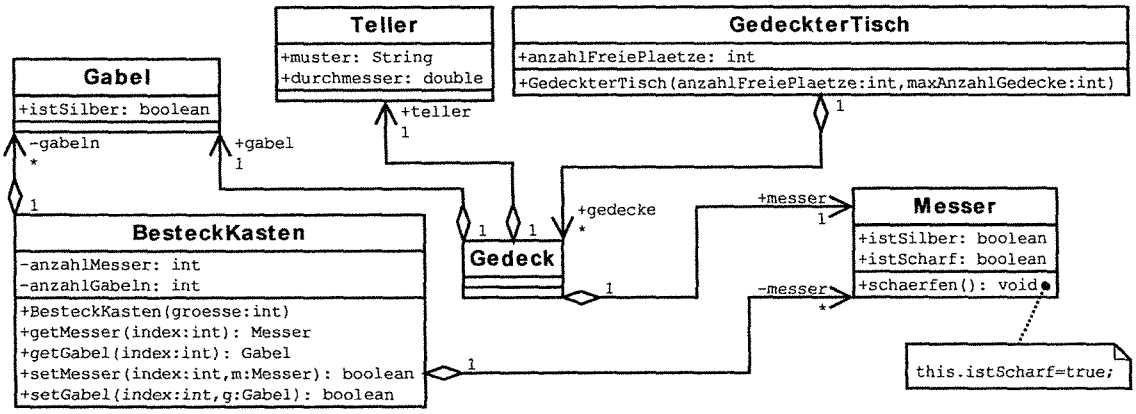
Gegeben sei folgende Beschreibung eines Informationssystems für Parkhäuser:

„Ein Parkhaus kann einen oder mehrere Ein- und Ausgänge haben. Jeder Ein- bzw. Ausgang hat einen entsprechenden Ticketautomaten und eine Schranke. Das Parkhaus hat mehrere Stockwerke. Jedes Stockwerk und das Parkhaus selbst haben eine bestimmte Kapazität sowie je eine Auslastung, die sich ändert, wenn ein Auto in das Parkhaus bzw. in das Stockwerk einfährt oder es wieder verlässt. Um ein solches Ereignis zu erkennen, sind an jedem Eingang, jedem Ausgang und in jedem Stockwerk entsprechende Sensoren angebracht, die dem Parkhaus bzw. dem Stockwerk einzelne ein- bzw. ausfahrende Autos signalisieren. An jedem Stockwerk und über jedem Eingang sind darüber hinaus Displays angebracht, die die Anzahl der freien Parkplätze im jeweiligen Stockwerk bzw. im gesamten Parkhaus anzeigen.“

a) Identifizieren Sie in der obigen Beschreibung alle möglichen Kandidaten für Klassen, Attribute und Methoden. Geben Sie jeweils an, welche davon Sie für geeignet halten und begründen Sie bei den anderen kurz, weshalb Sie sie eher verwerfen würden. Ordnen Sie die geeigneten Attribute und Methoden den jeweiligen Klassenkandidaten zu.

b) Erstellen Sie aus den von Ihnen vorangehend als geeignet identifizierten Kandidaten ein UML-Klassendiagramm einschließlich etwaiger Assoziationen zwischen den Klassen.

c) Implementieren Sie die im folgenden UML-Diagramm dargestellten Klassen in einer geeigneten objektorientierten Programmiersprache Ihrer Wahl:



Der Konstruktor von **`GedeckterTisch`** soll **`gedecke`** mit einem neuen Array der Größe **`maxAnzahlGedecke`** initialisieren. Der Konstruktor von **`BesteckKasten`** soll die beiden Array-Attribute mit entsprechenden Arrays der Größe **`groesse`** belegen.
Die **`get`**...-Methoden im **`BesteckKasten`** sollen den jeweiligen Array-Eintrag zurückgeben und stattdessen **`null`** eintragen. Die **`set`**...-Methoden im **`BesteckKasten`** sollen genau dann das übergebene Objekt an der genannten Stelle im Array eintragen und **`true`** zurückgeben, falls dort kein Objekt eingetragen ist - andernfalls sollen sie nichts tun und **`false`** zurückgeben.

Fortsetzung nächste Seite!

Aufgabe 4: „Zustandsautomaten"

Modellieren Sie das Einstellen einer Digitaluhr mit zwei Druckknöpfen mittels eines Zustandsautomaten, so dass Folgendes gilt:

- Nach Einlegen der Batterie befindet sich die Digitaluhr im Startzustand. Sie zeigt dabei zunächst die Uhrzeit „00:00:00" und läuft an (ein „interner Mechanismus" sendet dazu im Sekundentakt ein „tick"-Ereignis).
- Durch Betätigen des Druckknopfes A werden die drei Modi (Stunden stellen, Minuten stellen, Uhrzeitanzeige) zyklisch durchgeschaltet.
- Durch Betätigen des Druckknopfes B in einem der ersten beiden Modi wird die entsprechende Einheit (Stunden, Minuten) zyklisch erhöht – z.B. springt der Stundenzähler nach 21, 22, 23 wieder auf 0, 1, 2, ... Während der Einstellung einer Einheit steht die Uhr (d.h. „tick"s werden ignoriert). Die Betätigung des Druckknopfs B im Modus „Uhrzeitanzeige" ist wirkungslos.

Beschränken Sie sich dabei auf die folgenden drei Zustandsvariablen und stellen Sie die restlichen Zustände explizit graphisch dar:
Stunde: $h \in \{0, 1, 2, \ldots, 23\}$; Minute: $m \in \{0, 1, 2, \ldots, 59\}$; Sekunde: $s \in \{0, 1, 2, \ldots, 59\}$

Fortsetzung nächste Seite!

Teilaufgabe 2

1. ER-Modellierung

Gegeben seien folgende Informationen:

- Zirkusse geben Vorführungen. Zu jedem Zirkus ist dessen eindeutiger Name bekannt. Eine Vorführung wird durch ihr Datum gekennzeichnet, welches jedoch nur innerhalb eines Zirkus eindeutig ist.
- Es gibt Tickets für Vorführungen. Ein Ticket besitzt eine Kategorie und eine innerhalb einer Vorführung eindeutige Nummer.
- Für jeden Zirkus arbeiten Künstler. Für ein solches Beschäftigungsverhältnis ist die jeweilige Gage bekannt. Kein Künstler kann für mehrere Zirkusse arbeiten.
- Künstlern wird eine Personal-Nummer zugeordnet. Außerdem können Künstler in genau drei Kategorien eingeteilt werden: Akrobaten, Dresseure und Clowns. Manche Clowns sind auch Akrobaten.
- Es sind die Verantwortlichkeiten von Darbietungen innerhalb einer Vorführung folgendermaßen bekannt: Für Darbietungen in einer Vorführung ist immer nur ein Künstler verantwortlich. Für die gleiche Darbietung in verschiedenen Vorführungen können auch verschiedene Künstler verantwortlich sein. Zudem ist eine Darbietung über Ihren Namen eindeutig gekennzeichnet und besitzt zudem eine Dauer.
- Jeder Künstler erhält kostenlose Tickets.
- Manche Zirkusse drucken Werbeplakate. Ein Plakat besitzt eine ID und bewirbt höchstens einen Zirkus. Auf machen Plakaten sind ein oder mehrere Clowns abgebildet.

a) Erstellen Sie für das oben gegebene Szenario ein geeignetes ER-Diagramm. Verwenden Sie dabei – wenn angebracht – das Prinzip der Spezialisierung. Kennzeichnen Sie die Primärschlüssel der Entity-Typen, totale Teilnahmen und schwache Entity-Typen. Zeichnen Sie die Funktionalitäten der Relationship-Typen in das Diagramm ein.

b) Überführen Sie Ihr in Aufgabe a) erstelltes Modell in ein verfeinertes relationales Schema. Kennzeichnen Sie die Schlüssel durch Unterstreichen. Datentypen müssen nicht angegeben werden.

Fortsetzung nächste Seite!

2. SQL

Gegeben sei der folgende Ausschnitt des Schemas eines Flugverwaltungssystems:

```
Person : {[
    ID : INTEGER,
    Name : VARCHAR(255),
    Wohnort : VARCHAR(255)
]}

Airline : {[
    ID : CHAR(2),
    Name : VARCHAR(100),
    Sitz : VRACHAR(100)
]}

Flug : {[
    Airline : CHAR(2),
    Nummer : INTEGER,
    Datum : DATE,
    Start : VARCHAR(100),
    Ziel : VARCHAR(100),
    Entfernung: INTEGER,
    Flugzeit: REAL
]}

Ticket : {[
    PassagierID : INTEGER,
    Airline : CHAR(2),
    Nummer : INTEGER,
    Datum : DATE,
    Preis : REAL,
    Sitz : CHAR(3)
]}

Pilot : {[
    PilotID : INTEGER,
    Airline : CHAR(2),
    Nummer : INTEGER,
    Datum : DATE
]}
```

Die Tabelle *Person* enthält Informationen über Passagiere und Piloten. *Airline* beschreibt Airlines, die Flüge anbieten, mit ihrer Kurzbezeichnung, ihrem Namen und ihrem (Firmen-)Sitz. In *Flug* wird abgelegt, welche Flüge von welcher Airline an welchem Tag durchgeführt werden. *Ticket* beschreibt, welche Passagiere auf welchem Flug gebucht sind. *Pilot* beinhaltet die Piloten der Flüge.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Ticket* mit allen Constraints (einschließlich Fremdschlüsselconstraints) anlegt. Stellen Sie dabei sicher, dass negative Preise ausgeschlossen sind. Erklären Sie kurz, warum *Sitz* Teil des Primärschlüssels ist.

b) Schreiben Sie eine SQL-Anweisung, die für jede Airline die Flugnummer bestimmt, mit der im Jahr 2014 die größten Einnahmen erzielt wurden. *Hinweis: Verwenden Sie die Funktion EXTRACT(YEAR FROM Datum), um das Jahr aus Attribut Datum zu extrahieren.*

c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Passagiere bestimmt, die nur mit Airlines geflogen sind, deren Sitz an ihrem Wohnort ist.

d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Passagiere bestimmt, die immer mit dem gleichen Kapitän geflogen sind.

e) Schreiben Sie eine SQL-Anweisung, die alle Städte bestimmt, die mit maximal einem Zwischenstopp von 'München' aus erreichbar sind.

Beachten Sie bei der Formulierung der SQL-Anweisungen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

Fortsetzung nächste Seite!

3. Entwurfstheorie

Gegeben sei folgendes relationale Schema R in erster Normalform:

$$R : \{[A,B,C,D]\}$$

und die Zerlegung $\rho = \{R_1, R_2\}$ von R mit $R_1 = \{A, D\}$ und $R_2 = \{B, C, D\}$.
Für R gelte folgende Menge FD funktionaler Abhängigkeiten:
$FD = \{$

$$\begin{aligned} AB &\rightarrow CD, \\ B &\rightarrow C, \\ D &\rightarrow A, \end{aligned}$$

$\}$

a) Bestimmen Sie alle Kandidatenschlüssel von R mit FD.
Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.

b) Prüfen Sie, ob R mit FD in 2NF bzw. 3NF ist.

c) Zeigen oder widerlegen Sie, dass die Zerlegung ρ von R abhängigkeitserhaltend bzgl. FD *ist.*

d) Zeigen oder widerlegen Sie, dass die Zerlegung ρ von R verlustfrei bzgl. FD ist.

e) Bestimmen Sie eine verlustfreie Zerlegung von R in BCNF-Relationen.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1

Aufgabe 1: Modellierung von statischen Strukturen durch Klassendiagramme

Entwerfen Sie ein UML-Klassendiagramm mit den Klassen

> Schule, Lehrerzimmer, Direktor, Klassenzimmer, Schüler, Lehrer,
> Bibliothek, Raum, Mitarbeiter, Buch, Person

und mit geeigneten Assoziationen und Vererbungsbeziehungen. Assoziationen sind mit einem Namen zu versehen und an allen Assoziationsenden sind Multiplizitäten anzubringen. Verwenden Sie, wo möglich, abstrakte Klassen und kennzeichnen Sie diese. Es brauchen keine Attribute und Operationen angegeben zu werden.

Fortsetzung nächste Seite!

Aufgabe 2: Modellierung von Interaktionen durch Sequenzdiagramme

Die Fahrtrichtung der ersten elektrischen Spielzeugeisenbahnen wurde häufig durch Stromunterbrechung gesteuert. Dazu betrachten wir das Anwendungsfall-Diagramm in Abb. 1.

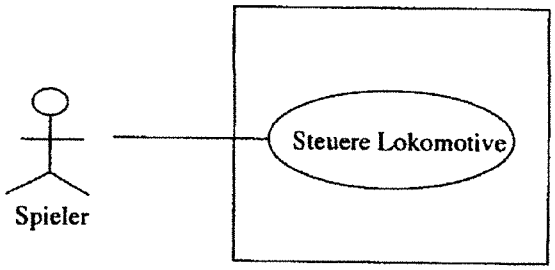


Abbildung 1: Anwendungsfall-Diagramm

An dem Anwendungsfall „Steuere Lokomotive" sind ein Spieler, als Aktor außerhalb des Systems, und jeweils ein Objekt der Klassen Stromschalter, Lokomotive, Scheinwerfer und Räder, als Objekte innerhalb des Systems, beteiligt. Zur Vereinfachung werden alle vier Räder durch ein einziges Objekt der Klasse Räder modelliert. Der Anwendungsfall zum Steuern einer Lokomotive wird durch folgendes Szenario beschrieben.

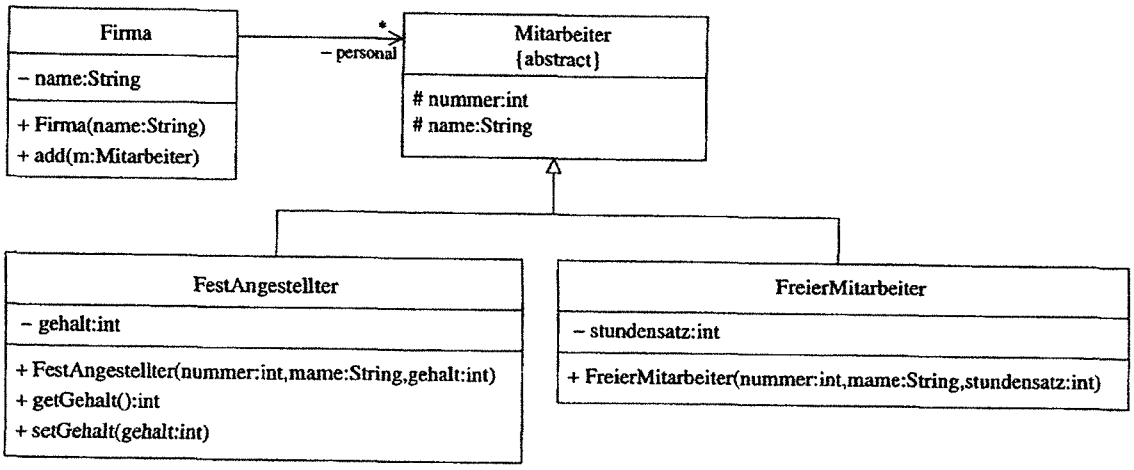
1. Der Spieler schaltet den Stromschalter ein, woraufhin der Schalter der Lokomotive *Strom zuführt*.
2. Die Lokomotive schickt nun den Rädern ein Signal um *vorwärts* zu fahren.
3. Dann schaltet der Spieler den Stromschalter aus, woraufhin der Schalter die Stromzufuhr bei der Lokomotive *abstellt*.
4. Daraufhin schickt die Lokomotive das Signal *stop* an die Räder.
5. Der Spieler schaltet jetzt den Stromschalter wieder ein, woraufhin der Schalter der Lokomotive *Strom zuführt*.
6. Die Lokomotive schickt den Rädern ein Signal um *rückwärts* zu fahren.
7. Nun schaltet der Spieler den Stromschalter wieder aus, woraufhin der Schalter die Stromzufuhr bei die Lokomotive *abstellt*.
8. Daraufhin schickt die Lokomotive wieder das Signal *stop* an die Räder.

Geben Sie ein Sequenzdiagramm an, das die oben beschriebenen Interaktionen zwischen Spieler, Stromschalter, Lokomotive und Rädern beschreibt.

Fortsetzung nächste Seite!

Aufgabe 3: Implementierung

Gegeben sei das folgende UML Klassendiagramm. Der Konstruktor der Klasse `Firma` soll den Namen der Firma initialisieren. Ein neu erzeugtes Firmenobjekt soll keine Mitarbeiter haben. Die Operation namens `add` der Klasse `Firma` soll einen Mitarbeiter zu dem Personal einer Firma hinzufügen. Die Konstruktoren der Klassen `FestAngestellter` und `FreierMitarbeiter` sollen die für die Objekte der jeweiligen Klasse gültigen Attribute initialisieren. Die Operationen namens `getGehalt` und `setGehalt` sollen das Gehalt eines fest angestellten Mitarbeiters liefern bzw. neu setzen.



a) Implementieren Sie das angegebene Klassendiagramm in einer objektorientierten Programmiersprache Ihrer Wahl. Die verwendete Sprache ist zu nennen. Beachten Sie dabei auch die im Klassendiagramm angegebenen Sichtbarkeiten. Insbesondere spezifiziert das Symbol # eine „protected" Sichtbarkeit. Für die Realisierung des Rollennamens `personal` und der Operation `add` der Klasse `Firma` kann ein geeigneter Kollektionstyp namens `HashSet` vorausgesetzt werden, der eine Methode namens `add` anbietet, die ein Objekt zu einer Kollektion hinzufügt, und einen Standardkonstruktor `HashSet()` besitzt, der ein Objekt erzeugt, das eine leere Menge repräsentiert. Alternativ dazu können Sie auch partielle Arrays zur Implementierung verwenden.

b) Schreiben Sie eine main-Methode mit Kopf `public static void main(String [ ] args)`, so dass nach Ausführung der Methode eine Firma namens „Magnifico" existiert, die eine fest Angestellte namens „Martha Huber" mit Nummer 1 und einem Gehalt von 4000 Euro sowie einen freien Mitarbeiter namens „Hugo Frost" mit Nummer 2 und einem Stundensatz von 100 Euro hat.

c) Erläutern Sie den Begriff *Polymorphismus* anhand eines Beispiels aus der obigen Anwendung.

Fortsetzung nächste Seite!

Aufgabe 4: Formale Verifikation mit vollständiger Induktion

Gegeben sei folgende rekursive Methodendeklaration in der Sprache Java. Es wird als Vorbedingung vorausgesetzt, dass die Methode `cn` nur für Werte `n` ≥ 0 aufgerufen wird.

```
int cn(int n) {
if (n == 0)
return 1;
else
return (4*(n-1) + 2) * cn(n-1) / (n+1);
}
```

Sie können im Folgenden vereinfachend annehmen, dass es keinen Überlauf in der Berechnung gibt, d.h. dass der Datentyp `int` für die Berechnung des Ergebnisses stets ausreicht.

a) Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `cn(n)` für jedes `n` ≥ 0 die n-te Catalan-Zahl C_n berechnet, wobei

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

Beim Induktionsschritt können Sie die beiden folgenden Gleichungen verwenden:

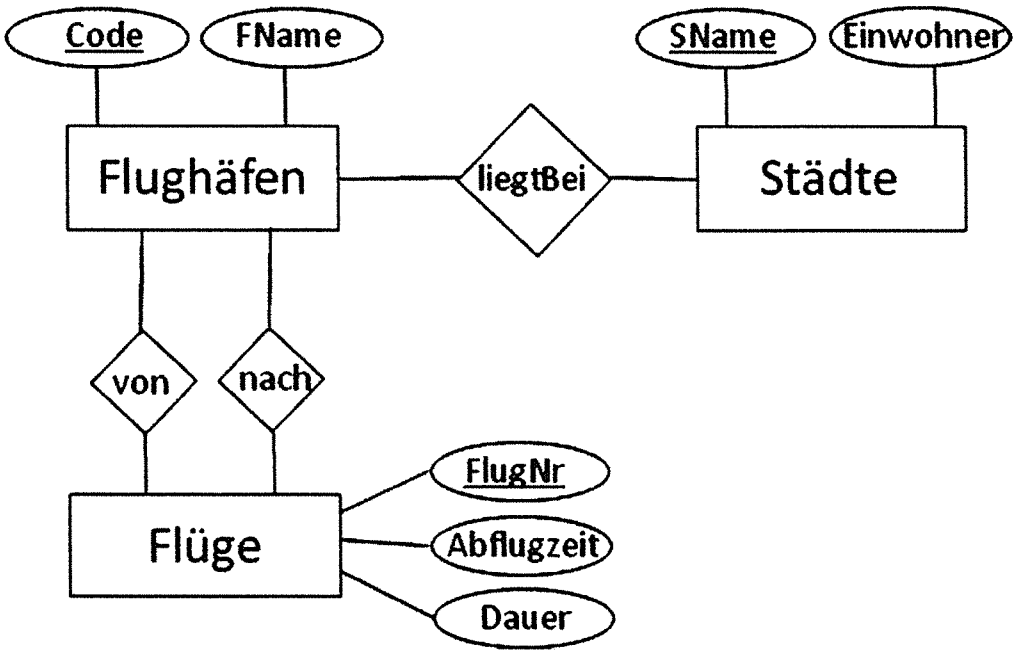
(i) $(2(n+1))! = (4n+2) \cdot (n+1) \cdot (2n)!$
(ii) $(n+2)! \cdot (n+1)! = (n+2) \cdot (n+1) \cdot (n+1)! \cdot n!$

b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie, warum der Methodenaufruf `cn(n)` für jedes `n` ≥ 0 terminiert.

Fortsetzung nächste Seite!

Teilaufgabe 2

1. Sie sollen für die Fluggesellschaft „FanHansa“ ein Fluginformationssystem erstellen. Als Grundlage des Entwurfs dient das folgende rudimentäre Entity-Relationship-Diagramm:



Zeichnen Sie das ER-Diagramm ab und vervollständigen Sie es mit

- Funktionalitäten (1:1, 1:N, N:1, N:M) und
- (min, max) – Angaben.

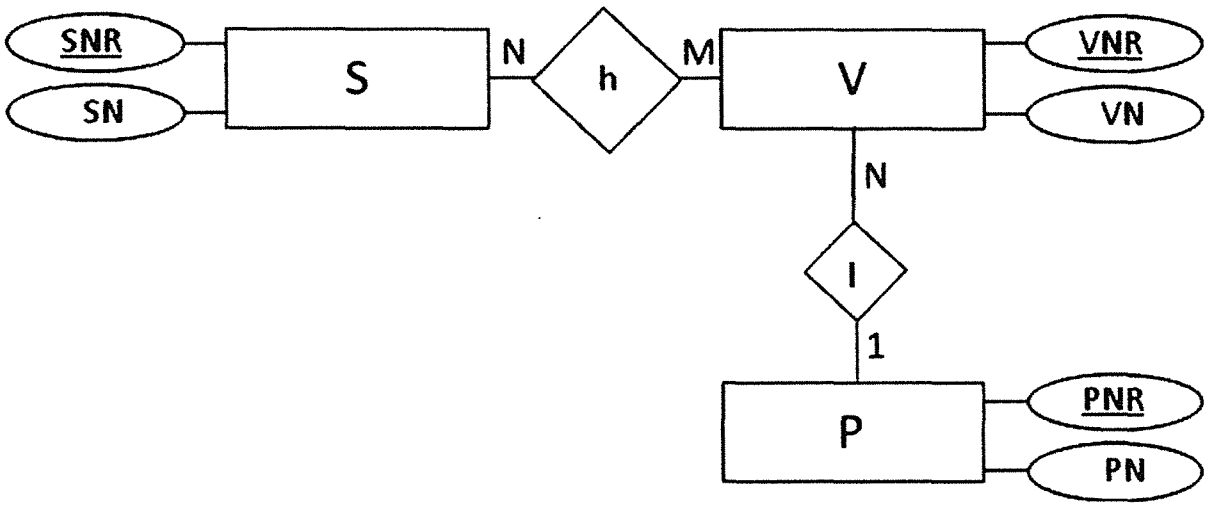
2. Setzen Sie das ER-Diagramm in ein relationales Schema um. Beachten Sie dabei:
   - Fassen Sie, wenn möglich, Relationen zusammen.
   - Markieren Sie Primärschlüssel.
   - Markieren Sie die Fremdschlüssel.

3. Geben Sie die Statements in SQL an, welche die Relationen aus 2. (inklusive Primär- und Fremdschlüsseldefinitionen) in eine relationale Datenbank implementieren.

4. Formulieren Sie in SQL folgende Anfragen auf Basis Ihrer oben erstellten Fluginformationssystem-Datenbank.
   a. Welche Städte kann man von München aus direkt erreichen?
   b. Welche Städte kann man von München aus mit maximal einmaligem Umsteigen erreichen?

Fortsetzung nächste Seite!

5. Gegeben sei folgendes ER-Diagramm:



Ein nicht-versierter Datenbank-Laie hat daraus das folgende relationale Schema generiert:

AllesZusammen: {[SNR, SN, VNR, VN, PNR, PN]}

a. Bestimmen Sie alle nicht-trivialen funktionalen Abhängigkeiten der Relation *AllesZusammen.*

b. Bestimmen Sie alle Kandidatenschlüssel der Relation *AllesZusammen.*

c. In welcher höchsten Normalform befindet sich das Schema?

d. Nomalisieren Sie die Relation. Sie dürfen wahlweise den Synthese- oder den Dekompositionsalgorithmus anwenden.

e. In welcher höchsten Normalform befinden sich Ihre generierten Schemata? Begründen Sie Ihre Antwort mithilfe der zugeordneten funktionalen Abhängigkeiten.

6. Wofür stehen die folgenden Abkürzungen?

   - ACID
   - WAL-Prinzip
   - LSN

Erläutern Sie jeweils die Bedeutung kurz. Erläutern Sie auch kurz das Konzept der Serialisierbarkeit.