Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 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: **12**

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1: Softwaretechnologie

Aufgabe 1:

a) Erklären Sie die Begriffe *abstrakte Klasse* und *Interface*, wie sie in UML (und Java) gebräuchlich sind.

b) Was versteht man unter einer *abstrakten Methode*?

c) Erläutern Sie kurz den Begriff einer *Schablonenmethode* (engl. template method).

d) Erläutern Sie kurz, inwiefern die objektorientierte Softwareentwicklung die Wiederverwendung von Software unterstützt.

Aufgabe 2:

Gegeben sei ein Softwaresystem mit drei Klassen *MainClass*, *FirstClass*, *SecondClass*, so dass die Klasse *MainClass* die *main*-Methode enthält. Zur Laufzeit sollen die im Sequenzdiagramm in Abb. 1 dargestellten Interaktionen stattfinden.

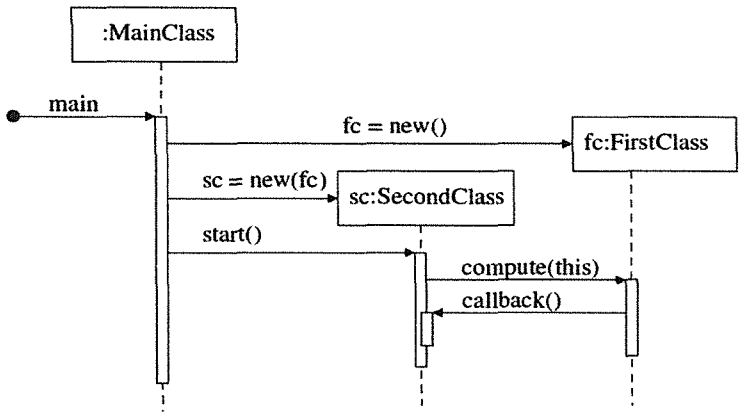


Abbildung 1: Interaktionen

Fortsetzung nächste Seite!

a) Geben Sie ein Klassendiagramm an, das die statische Struktur des Systems modelliert. Das Klassendiagramm soll alle für die obigen Interaktionen benötigten Modellelemente zeigen, einschließlich Assoziationen (mit Rollennamen und Multiplizitäten an gerichteten Assoziationsenden), sowie Konstruktoren und Operationen (jeweils ggf. mit formalen Parametern und deren Typen).

b) Implementieren Sie das System in einer objektorientierten Programmiersprache Ihrer Wahl. Für benutzerdefinierte Konstruktoren und Operationen (Methoden) sind die Rümpfe anzugeben, so dass zur Laufzeit die in Abb. 1 dargestellten Interaktionen stattfinden. Die Methode *callback* braucht nicht implementiert zu werden.

Aufgabe 3:

Es wird das Verhalten einer automatischen Straßenbahntüre betrachtet, die durch einen Verriegelungsschalter des Fahrers und durch eine Öffnungstaste des Fahrgasts gesteuert werden kann. Der Verriegelungsschalter kann vor- und zurückgestellt werden.

Zu Beginn ist die Straßenbahntüre verriegelt und geschlossen. Wird der Verriegelungsschalter vom Fahrer vorgestellt, dann ist die Türe entriegelt (bleibt aber geschlossen). Nach dem Zurückstellen des Schalters ist sie verriegelt. Das Drücken der Taste durch einen Fahrgast bewirkt das Folgende: Ist die Türe entriegelt, dann öffnet sie sich. Ist die Türe verriegelt, dann öffnet sie sich erst, wenn der Fahrer den Verriegelungsschalter vorstellt. Eine offene Türe schließt sich, wenn der Fahrer den Verriegelungsschalter zurückstellt.

Modellieren Sie das oben beschriebene Verhalten einer Straßenbahntüre durch ein UML-Zustandsdiagramm. Das Öffnen und Schließen der Türe kann als ununterbrechbare Aktion aufgefasst werden. Stellen Sie klar, welche Ereignisse Zustandsänderungen der Straßenbahntüre bewirken und welche Aktionen ggf. von einem Ereignis ausgelöst werden.

Teilaufgabe 2: Datenbanksysteme

Aufgabe 1 (Konzeptioneller Entwurf)

Im Folgenden ist die Beschreibung einer Bibliotheksverwaltung gegeben. Erstellen Sie zu dieser ein ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in (min, max)-Notation an. Modellieren Sie keine Attribute oder Entitytypen, die nicht aus dem Text hervorgehen.

In der Bibliotheksverwaltung werden Medien verwaltet. Jedes Medium wird über eine eindeutige Signatur identifiziert und hat einen Titel. Medien sind entweder Bücher oder Zeitschriften. Bücher haben ein Erscheinungsdatum. Zeitschriften haben einen Jahrgang und eine Ausgabenummer. Von jedem Medium kann es kein, ein oder mehrere Exemplare geben, die eine für das jeweilige Medium eindeutige Exemplarnummer haben. Ein Medium stammt von einem oder mehreren Autoren und ist von einem Verlag verlegt. Zu einem Autor werden Vorname und Nachname abgelegt, sowie das

Fortsetzung nächste Seite!

Geburtsdatum und der Geburtsort. Ein Autor wird über diese vier Eigenschaften eindeutig identifiziert. Verlage haben einen eindeutigen Namen. Autoren und Verlage sollen auch schon angelegt werden können, wenn noch kein zugehöriges Medium existiert. Nutzer haben einen Nachnamen, einen Vornamen und eine eindeutige Nutzernummer. Nutzer können Medienexemplare (auch mehrmals hintereinander) ausleihen. Zu jeder Ausleihe werden dauerhaft Entleihdatum und Rückgabedatum gespeichert. Nutzer können Bücher in verschiedenen Bewertungskategorien mit einer Anzahl Sterne bewerten. Eine Bewertungskategorie hat einen eindeutigen Namen und eine Beschreibung.

Aufgabe 2 (Normalformen)

Gegeben ist die Relation R(a, b, c, d), deren Attributwertebereiche alle atomar sind, mit den funktionalen Abhängigkeiten abc → d und d → b. Geben Sie an, in welcher höchsten Normalform R ist. Begründen Sie Ihre Antwort.

Aufgabe 3 (SQL)

Gegeben sind die folgenden Relationen mit den in Klammern angegebenen Attributen. Primärschlüssel sind unterstrichen, bei Fremdschlüsseln ist der Name der Relation, auf die sie sich beziehen, hinter dem Attributnamen in eckigen Klammern angegeben.

```
Laender (ID, Landesname)
Kunden (Kundennummer, Nachname, Vorname, Geburtsdatum, Land[Laender])
Hat_Geworben (Werber[Kunden], Geworbener[Kunden])
```

Keines der Attribute kann NULL-Werte enthalten.

Verwenden Sie in dieser Aufgabe nur standardkonformes SQL. Produktspezifische Erweiterungen (z.B. `limit`, `rownum`) werden als Fehler gewertet.

a) Schreiben Sie eine SQL-Anweisung, die Kundennummer, Vorname und Nachname aller Kunden ausgibt, aufsteigend sortiert nach Alter.

b) Schreiben Sie eine SQL-Anweisung, die eine Liste mit den Kundennummern der Kunden ausgibt, deren Geburtsdatum nicht einmalig unter den Kunden ist. Das heißt, ein Kunde muss ausgegeben werden, wenn es zu ihm einen anderen Kunden gibt, der dasselbe Geburtsdatum hat, sonst nicht. Jede Kundennummer soll maximal einmal ausgegeben werden.

c) Schreiben Sie eine SQL-Anweisung, die die Anzahl der Kunden pro Land ausgibt. Die Ausgabe soll den Namen des Landes und die Kundenanzahl enthalten. Sollten zwei Länder mit unterschiedlicher ID den gleichen Namen besitzen, sind beide getrennt voneinander auszugeben.

d) Schreiben Sie eine SQL-Anweisung, die pro Land den Kunden ermittelt, der am meisten andere Kunden (unabhängig von deren Land) geworben hat. Ausgegeben werden sollen pro Land der Name des Landes zusammen mit dem Vor- und Nachnamen des Kunden.

Fortsetzung nächste Seite!

Wenn es in einem Land mehrere Kunden mit gleicher höchster Geworbenenanzahl gibt, sollen diese alle ausgegeben werden. Wenn es in einem Land keine Kunden gibt, die andere Kunden geworben haben, soll das Land nicht ausgegeben werden. Sollten zwei Länder mit unterschiedlicher ID den gleichen Namen besitzen, sind beide getrennt voneinander auszugeben. Es wird empfohlen, eine View als Zwischenschritt zu verwenden.

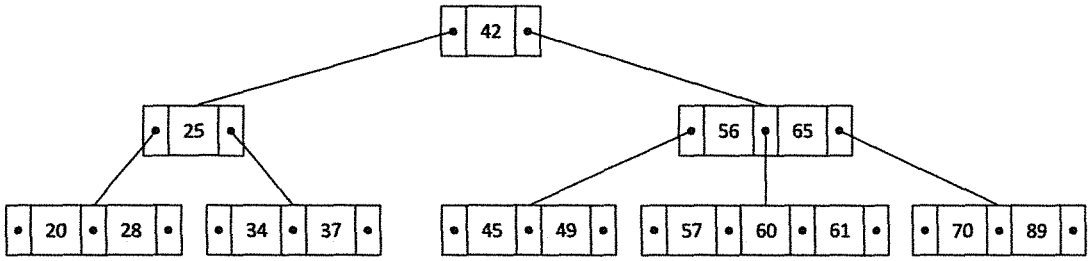
e) Schreiben Sie eine SQL-Anweisung, die bei allen Kunden aus *'Deutschland'* Vorname und Nachname vertauscht. Die Änderung soll auch dann korrekt stattfinden, falls *'Deutschland'* (mit verschiedenen IDs) mehrmals in der Tabelle Laender gespeichert ist.

Aufgabe 4 (Physische Datenorganisation)

a) Nennen Sie, aus welchen zwei Teilen ein TID (= Tupel-Identifikator) besteht.

b) Geben Sie an, wie viele Seiten maximal gelesen werden müssen, um das zu einem TID gehörende (unfragmentierte) Tupel zu lesen. Begründen Sie Ihre Antwort in zwei bis drei Sätzen.

c) Nennen Sie zwei Gründe, warum der Baum in der folgenden Skizze kein korrekter B-Baum ist. Eingetragen sind nur die Schlüsselwerte.



d) Nennen Sie den wesentlichen Unterschied eines B+-Baums (auch als B*-Baum bezeichnet) zu einem B-Baum.

e) Geben Sie an, welche Seite bei der Seitenersetzung im Datenbankpuffer idealerweise ersetzt werden sollte.

f) Gegeben sind die zwei Indexstrukturen B+-Baum (auch als B*-Baum bezeichnet) und Hash-Index, basierend auf erweiterbarem Hashing.
Geben Sie an, welche der beiden Indexstrukturen für Bereichsanfragen (z.B. Personen mit Alter zwischen 20 und 40) besser geeignet ist. Erklären Sie in ein bis drei Sätzen, wie eine solche Anfrage mit Hilfe des besser geeigneten Index ausgewertet wird.

Fortsetzung nächste Seite!

Aufgabe 5 (Transaktionen)

a) Nennen Sie die vier wesentlichen Eigenschaften einer Transaktion und erläutern Sie jede Eigenschaft kurz (ein Satz pro Eigenschaft).

b) Gegeben ist die folgende Historie (Schedule) dreier Transaktionen:
$r_1(B) \rightarrow w_1(C) \rightarrow r_3(C) \rightarrow r_1(A) \rightarrow c_1 \rightarrow r_2(C) \rightarrow r_3(C) \rightarrow r_2(C) \rightarrow w_2(B) \rightarrow c_2 \rightarrow c_3$

Zeichnen Sie den Serialisierbarkeitsgraphen zu dieser Historie und begründen Sie, warum die Historie serialisierbar ist oder nicht.

c) Geben Sie an, wodurch die erste und die zweite Phase des Zwei-Phasen-Sperrprotokolls jeweils charakterisiert sind (ein Satz pro Phase).

Thema Nr. 2

(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1: Softwaretechnologie

Aufgabe 1: „Formale Spezifikation Abstrakter Datentypen (ADT)"

WICHTIG: In dieser Aufgabe stehen **keine** anderen Datentypen (also **kein int**, **String** usw.), **keine** anderen Konstanten (auch **nicht** 0, 1 usw.) und **keine** anderen Operationen (also **kein** <, +, =, ≠ usw.) zur Verfügung: Sie dürfen ausschließlich die vorgegebenen oder zu ergänzenden **adt**s mit deren **ops** verwenden!
Gegeben seien die folgenden Gerüste der ADTs *Nat* (Repräsentation natürlicher Zahlen $\mathbb{N}_0$) und *NatStack* (Stapel mit *Nat*-Elementen):

---

adt *Nat*
sorts *Nat*
ops

NaN:		→ *Nat*	*// „Not a Nat" – vergleichbar mit Double.NaN*
zero:		→ *Nat*	*// entspricht der natürlichen Zahl „0"*
succ:	*Nat*	→ *Nat*	*// Nachfolger „(x + 1)" der Nat-Zahl x*
add:	*Nat* × *Nat*	→ *Nat*	*// Addition im Nat-Raum*
sub:	*Nat* × *Nat*	→ *Nat*	*// Subtraktion im Nat-Raum (z.B. „42 − 666 = 0")*
mul:	*Nat* × *Nat*	→ *Nat*	*// Multiplikation im Nat-Raum*
...			

axs

succ(NaN) = NaN	*// alle Operationen mit NaN ergeben stets wieder // NaN*
...	*// weitere Axiome aus Platzgründen weggelassen*

end *Nat*

---

adt *NatStack*
sorts *NatStack Nat*
,
ops

empty:		→ *NatStack*	*// erzeugt einen neuen leeren Stapel*
push:	*NatStack* × *Nat*	→ *Nat*	*// legt ein Nat **auf** den Stapel*
peek:	*NatStack*	→ *Nat*	*// gibt bei leerem Stapel NaN und bei // nicht-leerem Stapel das „**oberste**" Nat zurück*
pop:	*NatStack*	→ *NatStack*	*// gibt bei leerem Stapel empty und bei nicht - // leerem // Stapel den Stapel ohne das „**oberste**" Nat // zurück*
...			*// Axiome aus Platzgründen weggelassen*

axs

end *NatStack*

Fortsetzung nächste Seite!

a) Ergänzen Sie *NatStack* um diejenigen Axiome, die das Zusammenspiel von *peek* bzw. *pop* mit den Primärkonstruktoren *empty* und *push* spezifizieren.

b) Ergänzen Sie den ADT *NatStack* um Signaturen und Axiome für die Operation *size*, die die Anzahl der Elemente im Stapel als *Nat*-Zahl berechnet.

c) Ergänzen Sie den ADT *NatStack* um Signaturen und Axiome, die das Verhalten der folgenden Methode **nat2bin** spezifizieren, die für eine *Nat*-Zahl ihre Binärdarstellung (Zweierkomplement) als *NatStack* erzeugt:

```
public static NatStack nat2bin(Nat n) {
if (n == zero)
            return push(empty, zero);
      else
return push(nat2bin(div(n, succ(succ(zero)))),
                        mod(n, succ(succ(zero))));
}
```

$$\textit{Beispiel: } 42_{(10)} \xrightarrow{nat2bin} \left\{ \begin{matrix} \mathbf{zero} \\ succ(zero) \\ zero \\ succ(zero) \\ zero \\ succ(zero) \\ zero \end{matrix} \right\} = 0101010_{(2)}$$

d) Ergänzen Sie den ADT *NatStack* um Signaturen und Axiome für die Operation **bin2nat**, die aus einer Binärdarstellung im Stapel die zugehörige *Nat*-Zahl berechnet. Es wird garantiert, dass nur *zero* oder *succ(zero)* auf dem Stapel liegen und dass das *niederwertigste* Bit das „**oberste**" ist.

e) Ergänzen Sie den ADT *NatStack* um die Axiome für die Operation *revHelper* so, dass *reverse* die Reihenfolge der Elemente im Stapel „umdreht", d.h. u.a. dass das oberste Element zum untersten wird:

Fortsetzung nächste Seite!

ops

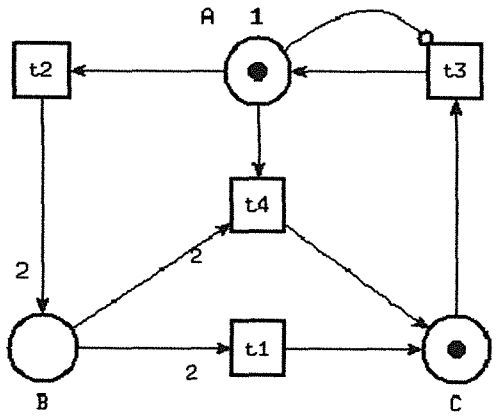
revHelper: NatStack × NatStack → NatStack
reverse: NatStack → NatStack

axs

reverse(s) = revHelper(s, empty)
revHelper(...) = ...

Aufgabe 2: „Petri-Netze"

Gegeben sei das folgende Petri-Netz:

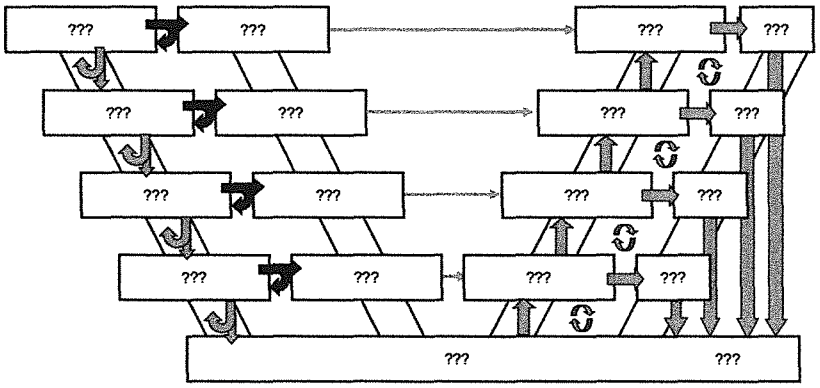


a) Erstellen Sie den zum Petri-Netz gehörenden Erreichbarkeitsgraphen. Die Belegungen sind jeweils in der Form [A, B, C] anzugeben. Beschriften Sie auch jede Kante mit der zugehörigen Transition. Beachten Sie die auf 1 beschränkte Kapazität von Stelle A oder alternativ die Inhibitor-Kante von A zu t3 (beides ist hier semantisch äquivalent).
b) Wie kann man mit Hilfe des Erreichbarkeitsgraphen feststellen, ob ein Petri-Netz lebendig ist?
c) Aufgrund von Transition t4 ist das gegebene Petri-Netz nicht stark lebendig. Wie müssten die Pfeilgewichte der Transition t4 verändert werden, damit das Petri-Netz mit der gegebenen Startmarkierung **beschränkt bleibt** und **lebendig wird**?

Fortsetzung nächste Seite!

Aufgabe 3: „W-Modell“

Das sogenannte W-Modell für die Softwareentwicklung erweitert das klassische V-Modell um ein zweites (leicht nach rechts versetztes) V, das die Testaktivitäten explizit und jeweils parallel zu den entsprechenden Entwicklungsaktivitäten aufführt:



a) Übernehmen Sie das skizzierte W-Modell und ergänzen Sie die jeweils fehlenden Phasen, Aktivitäten und Produkte („???“ im Bild).
b) Beschreiben Sie kurz die Tätigkeiten bzw. Produkte, die in jeder Phase des klassischen V-Modells (der linke absteigende und der linke aufsteigende Ast im W-Modell) ausgeführt respektive erzeugt werden.
c) Erklären Sie kurz den Zweck und das zugehörige Vorgehen der vier Aktivitäten im zweiten absteigenden Ast des W-Modells – oder mit anderen Worten: Welche Bedeutung haben die Doppelpfeile?

Fortsetzung nächste Seite!

Teilaufgabe 2: Datenbanksysteme

1. Modellieren Sie eine Agentur, die Konzerte veranstaltet. Auf diesen Konzerten treten mehrere Bands auf. Die Eintrittskarten sollen eindeutig nummeriert sein (Ticket#) und sollen Block, Reihe und Platz# enthalten. Die Preise sollen pro Block gleich sein. Die Reihenfolge des Auftritts der Bands ist relevant.

   a. Erstellen Sie den konzeptionellen Entwurf mittels ER- oder UML-Diagramm.
   b. Markieren Sie die Funktionalitäten der Beziehungen.

2. Setzen Sie Ihren konzeptuellen Entwurf systematisch in ein relationales Schema um. Markieren Sie Schlüssel und Fremdschlüssel. Für eine der Relationen geben Sie auch die SQL-Definition mitsamt Primär- und Fremdschlüsselspezifikationen an.

3. Ihr relationales Schema könnte (muss aber nicht) beispielsweise folgende Relation enthalten:

   KonzertBandsTickets: {[KonzertID, Ticket#, BandID]}

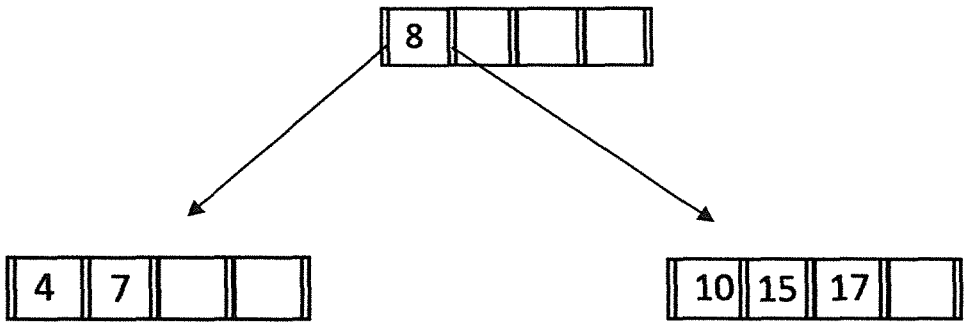
   a. Welche funktionalen Abhängigkeiten gibt es?
   b. Markieren Sie die Kandidatenschlüssel.
   c. Normalisieren Sie diese Relation systematisch.
   d. Diskutieren Sie die Qualität der resultierenden Relationen kritisch.

4. Auf der Basis der Relation KonzertBandsTickets sollen folgende Anfragen in SQL formuliert werden:

   a. Welches Konzert hatte die meisten Besucher?
   b. Welche Band-Paare wurden häufig zusammen gebucht (sind also mindestens fünf Mal gemeinsam bei Konzerten aufgetreten)?
   c. Welche Dreier-Kombinationen von Bands wurden mehrmals (also mindestens zweimal) bei Konzerten gebucht?

Fortsetzung nächste Seite!

5. Betrachten Sie folgenden B-Baum mit maximaler Knotenkapazität 4.



Zeigen Sie nach jeder nachfolgenden Operation den resultierenden Baum:

a. Fügen Sie 9 ein.
b. Fügen Sie 13 ein.
c. Löschen Sie 7.

6. Vergleich von Indexstrukturen

a. Welche Höhe hat ein B-Baum, der als Schlüssel alle 10 Milliarden Telefonnummern weltweit indexiert? Die maximale Knotenkapazität sei 200; mimimal also 100 (bis auf die Wurzel).

b. Welche Vorteile bzw. Nachteile bietet ein Hash-basierter Index im Vergleich zum B-Baum?