Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 2017

66116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Datenbanksysteme, Softwaretechnologie**

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

Anzahl der Druckseiten dieser Vorlage: **16**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

1. Modellierung

Modellieren Sie mit Hilfe des Entity-Relationship-Modells die Formel 1. Jedes Team hat pro Saison 2 Fahrer, die jeweils ein eigenes Auto fahren. Für die Siegerehrung ist wichtig zu wissen, welcher Nationalität die Fahrer angehören.
Es gibt mehrere Rennen (8 bis 20), die man über den Ort und das Jahr identifiziert. Die Platzierungen der Fahrer müssen für jedes Rennen gespeichert werden. Pro Rennen werden Punkte gemäß folgender Tabelle vergeben:

Platzierung	Punkte
1	25
2	18
3	15
4	12
5	10
6	8
7	6
8	4
9	2
10	1

Der Fahrer mit der höchsten Gesamtpunktzahl wird Weltmeister. Das Team, dessen beide Fahrer zusammen die meisten Punkte erzielt haben, wird Konstrukteursweltmeister. Überführen Sie Ihren ER-Entwurf in ein relationales Schema. Markieren Sie Schlüssel und Fremdschlüssel.

- Formulieren Sie diese beiden Abfragen, die den Weltmeister bzw. den Konstrukteursweltmeister ermitteln, jeweils in SQL.
- Formulieren Sie in SQL die Abfrage: Welcher Fahrer hat die meisten Siege in der Saison 2015 erzielt?
- Formulieren Sie in SQL die Abfrage: Welcher Fahrer war in keinem Rennen besser als Hamilton?

Fortsetzung nächste Seite!

2. Datenbanksysteme: Aufbau eines B-Baums

Konstruieren Sie einen B-Baum, dessen Knoten maximal 4 Einträge enthalten können, indem Sie der Reihe nach diese Suchschlüsssel einfügen:

- 8, 10, 12, 20, 5, 30, 25, 11
- Anschließend löschen Sie den Eintrag mit dem Suchschlüssel 8.

Zeigen Sie jeweils graphisch den entstehenden Baum nach relevanten Zwischenschritten; insbesondere nach Einfügen der 5 sowie nach dem Einfügen der 11 und nach dem Löschen der 8.

3. Datenbanksysteme: Normalisierung

Gegeben sei dieses Relationenschema:

Allerlei: {[MatrikelNr, SName, VNr, VTitel, PersNr, PName]}

Hier werden also Studenten (mit den Attributen MatrikelNr, SName), Vorlesungen (mit den Attributen VNr, VTitel) sowie Professoren (mit den Attributen PersNr und PName) modelliert und in Beziehung gesetzt. Es sollen natürlich viele Studenten dieselbe Vorlesung hören können, aber nur ein/e Professor/in kann die Vorlesung lesen:

- Geben Sie die nicht-trivialen funktionalen Beziehungen der Relation Allerlei an.
- Geben Sie alle Kandidatenschlüssel der Relation an.
- Normalisieren Sie die Relation systematisch mit Erläuterung.

Fortsetzung nächste Seite!

4. Normalisierung und SQL

Gegeben sei eine Relation Zehnkampf mit dem Schema

{SportlerName, Disziplin, ZeitHöheWeite, Punkte}:

SportlerName	**Disziplin**	**ZeitHöheWeite**	**Punkte**
"Jim Knopf"	"100 Meter"	10,89	986
"Usain Bolt"	"100 Meter"	9,81	1723
"Jim Knopf"	"StabHoch"	4,56	976
"Usain Bolt"	"StabHoch"	2,80	430
.....	.....	.....	.....

Für die Bearbeitung dürfen folgende Abkürzungen verwendet werden:

SN für SportlerName

D für Disziplin

ZHW für ZeitHöheWeite

P für Punkte

- Welche funktionalen Abhängigkeiten gelten?
- Welches sind die (Kandidaten-) Schlüssel der Relation?
- Normalisieren Sie die Relation.
- Formulieren Sie eine Abfrage in SQL, die die Sportler sortiert nach der Gesamtzahl ihrer Punkte ausgibt. Verwenden Sie die Original-Relation Zehnkampf ohne der Normalisierung.
- Formulieren Sie folgende Abfrage in SQL und in der Relationenalgebra. Verwenden Sie die Original-Relation Zehnkampf ohne der Normalisierung:
  Wer ist in der Disziplin „StabHoch“ besser als „Usain Bolt“?
- Formulieren Sie folgende Abfragen in SQL. Verwenden Sie die Original-Relation Zehnkampf ohne der Normalisierung:
  - Wer hat insgesamt mehr Punkte erzielt als „Jim Knopf“?
  - Wer hat Medaillen gewonnen? (Nur Standard SQL 92 verwenden und keine Limit-Klausel!)
  - Wer hat in mindestens einer Disziplin mehr Punkte erzielt als „Usain Bolt“ (egal in welcher)?

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1 (Qualitätskriterien von Software)

Nennen Sie vier Qualitätskriterien von Software. Geben Sie davon mindestens ein Qualitätskriterium an, das durch objektorientierte Softwareentwicklung besonders unterstützt wird und begründen Sie dies.

Aufgabe 2 (Vorgehensmodelle)

Ein wichtiges Vorgehensmodell in der Softwareentwicklung ist das V-Modell. Zeichnen Sie das V-Modell, so dass deutlich wird, welche Testphasen welchen Entwicklungsphasen gegenüberstehen.

Aufgabe 3 (Modellierung von Interaktionen)

Am Eingang eines Hallenbads befindet sich ein Drehkreuz, das mit einer Lichtanzeige verbunden ist. Das Drehkreuz und die Lichtanzeige werden durch das Klassendiagramm in Abb. 1 modelliert.

Abbildung 1: Klassendiagramm für Drehkreuz mit Lichtanzeige

Durch das Drehkreuz können Badegäste, die im Besitz einer gültigen Eintrittskarte sind, in das Hallenbad eintreten. Bevor das Drehkreuz passiert werden kann, muss ein Badegast die Eintrittskarte in einen Schlitz an der Seite des Drehkreuzes eingeben. Es gibt einen Anwendungsfall *Am Drehkreuz eintreten*, der durch folgendes Primär- und Sekundärszenario beschrieben wird. Als Vorbedingung für den Anwendungsfall gilt, dass das Drehkreuz gesperrt ist.

Primärszenario (Drehkreuz passieren):

1. Der Badegast gibt seine Eintrittskarte am Drehkreuz ein.
2. Das Drehkreuz veranlasst, dass an der Lichtanzeige das grüne Licht eingeschaltet wird, und gibt dann die Eintrittskarte wieder aus.
3. Der Badegast entnimmt die Karte, woraufhin sich das Drehkreuz entsperrt.
4. Der Badegast passiert das Drehkreuz, woraufhin das Drehkreuz veranlasst, dass an der Lichtanzeige das grüne Licht wieder ausgeschaltet wird.
5. Danach sperrt sich das Drehkreuz wieder.

Fortsetzung nächste Seite!

Sekundärszenario (Karte ungültig):

1. Der Badegast gibt seine Eintrittskarte am Drehkreuz ein.
2. Das Drehkreuz veranlasst, dass an der Lichtanzeige das rote Licht eingeschaltet wird, und gibt dann die Eintrittskarte wieder aus.
3. Der Badegast entnimmt die Karte, woraufhin das Drehkreuz veranlasst, dass an der Lichtanzeige das rote Licht wieder ausgeschaltet wird.

In den folgenden Teilen a) und b) sind Sequenzdiagramme zu erstellen, in denen der Badegast als Aktor und Interaktionen durch Nachrichten zu modellieren sind. Beachten Sie, dass sich ein Objekt innerhalb des Systems auch selbst eine Nachricht schicken kann.

a) Geben Sie ein Sequenzdiagramm an, das die Interaktionen zwischen einem Badegast, dem Drehkreuz und der Lichtanzeige gemäß des oben beschriebenen Primärszenarios zeigt.

b) Geben Sie ein Sequenzdiagramm an, das die Interaktionen zwischen einem Badegast, dem Drehkreuz und der Lichtanzeige gemäß des oben beschriebenen Sekundärszenarios zeigt.

c) Welche Operationen müssen zu der Klasse *Drehkreuz* hinzugefügt werden, damit die Nachrichten aus den Sequenzdiagrammen in den beiden Aufgabenteilen a) und b) an das Drehkreuz durch Operationsaufrufe realisiert werden können?

d) Welche Operationen müssen zu der Klasse *Lichtanzeige* hinzugefügt werden, damit die Nachrichten aus den Sequenzdiagrammen in den beiden Aufgabenteilen a) und b) an die Lichtanzeige durch Operationsaufrufe realisiert werden können?

Fortsetzung nächste Seite!

Aufgabe 4 (Zustandsdiagramme und deren Implementierung)

Ein einfacher Taschenrechner wird durch die Klasse *Calculator* in Abb. 2 modelliert. Das Attribut *ac* repräsentiert den Accumulator, das Attribut *m* das Memory des Taschenrechners und *Calculator*() den Konstruktor. Das Verhalten des Taschenrechners wird durch das Zustandsdiagramm in Abb. 3 beschrieben. Alle Ereignisse an den Transitionen repräsentieren Operationsaufrufe (*call events*) für die in der Klasse angegebenen Operationen.

Der Taschenrechner unterstützt die Addition nicht-negativer ganzer Zahlen. Die Eingabe einer Zahl erfolgt ziffernweise durch Eingabe von Ziffern zwischen 0 und 9. Die Eingabe einer Ziffer *k* wird durch ein Ereignis der Form *digit(k)* modelliert. Das Drücken der Tasten +, = und AC wird durch die Ereignisse *plus*(), *result()* und *AC()* modelliert. Ereignisse, für die es in einem Zustand keine ausgehende Transition gibt, können zwar eintreten, haben in diesem Zustand aber keine Wirkung.

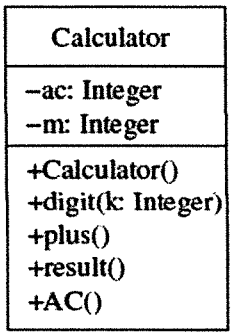


Abbildung 2: Klasse Calculator

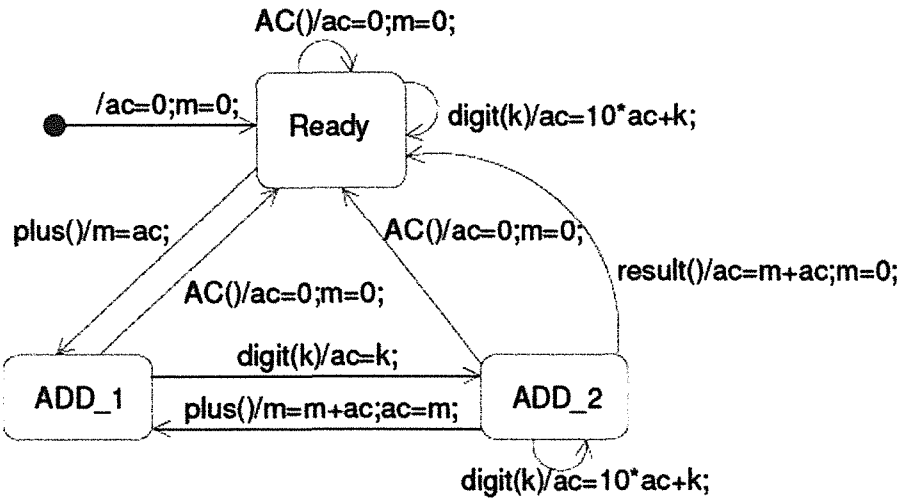


Abbildung 3: Zustandsdiagramm für Calculator

Fortsetzung nächste Seite!

a) Für ein neu erzeugtes Calculator-Objekt sollen der Reihe nach folgende Ereignisse eintreten: *digit(5), plus(), digit(2), plus(), digit(3), result(), plus(), digit(6), digit(1), result()*. In welchem Zustand ist das Calculator-Objekt zu Beginn und wie sind dann seine Attributwerte? Geben Sie an, in welchem Zustand das Calculator-Objekt nach jedem einzelnen Ereignis ist und wie dann seine Attributwerte jeweils sind.

b) Implementieren Sie die Klasse *Calculator* in einer objektorientierten Programmiersprache Ihrer Wahl (verwendete Sprache bitte angeben), so dass das in dem Zustandsdiagramm beschriebene Verhalten realisiert wird. Insbesondere sind der Konstruktor und alle angegebenen Operationen zu implementieren. Die Implementierung soll nach einer der beiden folgenden Strategien erfolgen:

   (A) Einführung eines Enumerationstyps zur Darstellung der Zustände und Implementierung der Operationen (Methoden) durch Fallunterscheidung nach dem jeweiligen Zustand.

   (B) Anwendung des Design Patterns namens *State*, in dem Zustände durch Objekte geeigneter zusätzlicher Klassen repräsentiert werden.

Fortsetzung nächste Seite!

Aufgabe 5 (Systemarchitektur mit Interfaces)

Gegeben sind die in Abb. 4 dargestellten Klassen zur Modellierung eines objektorientierten Systems.

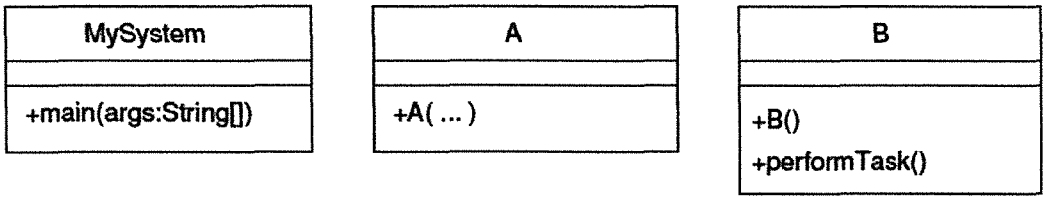


Abbildung 4: Klassen eines objektorientierten Systems

Zur Laufzeit soll

i) genau ein Objekt der Klasse *A* mit genau einem Objekt der Klasse *B* fest verbunden werden,
ii) das Objekt der Klasse *A* die Operation *performTask()* des *B*-Objekts aufrufen.

Zur Programmierzeit sollen sich die Klassen *A* und *B* nicht kennen (d.h. es gibt keine Abhängigkeiten zwischen den beiden Klassen). Zur Lösung ist ein Interface zu verwenden. Erstellen Sie ein Klassendiagramm, das die drei Klassen *MySystem*, *A* und *B* beinhaltet und zusätzlich Folgendes zeigt:

a) das benötigte Interface und welche Operationen es anbietet,

b) eine gerichtete Assoziation (mit Rollennamen und Multiplizität am gerichteten Assoziationsende), die zeigt, welche Klasse das Interface benutzt,

c) eine Realisierungsbeziehung, die zeigt, welche Klasse das Interface implementiert,

d) einen geeigneten formalen Parameter für den Konstruktor von *A*.

Des Weiteren sind in einer objektorientierten Programmiersprache Ihrer Wahl anzugeben:

e) die Implementierung des Konstruktors der Klasse *A*,

f) die Implementierung der main-Methode der Klasse *MySystem*.

Thema Nr. 2

Teilaufgabe 1

Aufgabe 1: ER-Modellierung

Stadt, Land, Fluss mal anders: Die relationale Datenbank eines geographischen Systems beinhaltet Städte, Länder und Flüsse mit folgenden Einzelheiten:

- Eine Stadt wird durch ihren eindeutigen Namen und das Land gekennzeichnet, in dem sie liegt. Außerdem hat jede Stadt eine Einwohnerzahl.
- Ein Land hat einen eindeutigen Namen, eine Fläche und ebenfalls eine Einwohnerzahl. Ein Land kann an mehrere Länder angrenzen. In einem Land liegt mindestens eine Stadt, wobei jede Stadt nur in genau einem Land liegt. Außerdem hat jedes Land eine Hauptstadt.
- Ein Fluss wird durch seinen Namen bestimmt und hat eine gewisse Länge. Jeder Fluss kann durch mehrere Städte und auch durch mehrere Länder fließen und kann am Ende in einen anderen Fluss münden. Eine Stadt kann an mehreren Flüssen liegen und durch ein Land können mehrere Flüsse fließen.

Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank. Markieren Sie die Funktionalitäten der Relationships und unterstreichen Sie den Primärschlüssel jeder Entität.

Aufgabe 2: Tupelkalkül

Gegeben sei das folgende Datenbankschema. Alle Attribute seien vom Datentyp String.

```
Wirt(WNr, WName, WStadt)
Speise(SNr, Bezeichnung, Preis)
Gast(GNr, GName, GStadt)
Bestellung(WNr, SNr, GNr)
```

Geben Sie für die folgende, verbal formulierte Anfrage einen äquivalenten Ausdruck im **Tupelkalkül** an.
Gesucht sind Nummer, Name und Stadt der Gäste, die niemals ein Schnitzel bestellt haben.

Fortsetzung nächste Seite!

Aufgabe 3: Relationale Algebra

Gegeben seien folgende Relationen:

X

A	B	C	D
1	4	6	3
6	2	5	1
4	5	6	3
1	4	5	2
1	4	6	2
3	3	5	2
4	3	1	2
4	5	6	2

Y

C	D	E
5	2	3
6	2	1
6	3	1

Z

D	F
6	2
5	6
1	3

Geben Sie die Ergebnisrelationen folgender Ausdrücke der relationalen Algebra als Tabellen an.

1. $X \underset{D=F}{\bowtie} \sigma_{F>2}(\pi_F(Z))$

2. $(\pi_{C,D}(X) - \pi_{C,D}(Y)) \times \pi_D(Z)$

3. $\pi_{A,D}(X) \div \pi_D(Y)$

Aufgabe 4: SQL-Abfragen

Die Spieler in einer Fußballmannschaft in der Saison 2015 werden in folgender Relation verwaltet:

```
Saison2015(PersonalID, OK1, OK2, AnzahlSpiele, AnzahlTore).
```

Alle Attribute sind Integer. Für jeden Spieler wird ein Tupel mit der entsprechenden PersonalID angelegt, die ihn über die gesamte Vereinshistorie hinweg eindeutig identifiziert. Die Werte für die Attribute OK1 und OK2 haben den Defaultwert 0. Besteht ein Spieler den 1. Fitnesstest des Jahres, dann wird OK1 auf 1 aktualisiert. Besteht ein Spieler den 2. Fitnesstest des Jahres, dann wird OK2 auf 1 aktualisiert. Die Werte der Attribute AnzahlSpiele und AnzahlTore werden nach jedem Spiel, in dem der Spieler angetreten ist, aktualisiert.
Für die Saison 2014 liegt eine weitere Relation vor, die die Spieler des Jahres 2014 und ihre Ergebnisse in den Fitnesstests des Jahres 2014 in gleicher Weise abspeichert:

```
Saison2014(PersonalID, OK1, OK2, AnzahlSpiele, AnzahlTore).
```

Fortsetzung nächste Seite!

Bitte beachten Sie: Ein Spieler, der in der Saison 2015 in dieser Mannschaft spielt, muss nicht schon 2014 in dieser Mannschaft gespielt haben. Die PersonalID eines Spielers ist eindeutig über alle Saisonen hinweg.
Wegen finanzieller Engpässe muss sich der Verein von einigen Spielern trennen und möchte daher:

1. in **einer einzigen SQL-Abfrage** alle Spieler ermitteln,
   - die im Jahr 2015 beide Fitnesstests bestanden haben **oder**
   - die in den Jahren 2014 und 2015 zusammen mindestens 3 von 4 Fitnesstests bestanden haben.

   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

2. in **einer einzigen SQL-Abfrage** die Spieler ermitteln, die in mindestens einer der beiden Saisonen in weniger als fünf Spielen angetreten sind.
   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

3. in **einer einzigen SQL-Abfrage** die Spieler ermitteln, die in jeder der beiden Saisonen in mindestens einem Spiel angetreten sind und bei denen das Verhältnis der geschossenen Tore pro Spiel im Durchschnitt in mindestens einer der beiden Saisonen unter 10 % liegt.
   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

4. in **einer einzigen SQL-Abfrage** die Spieler ermitteln, die 2015 mehr Tore geschossen haben als 2014.
   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

5. in **einer einzigen SQL-Abfrage** ermitteln, welche Spieler 2015 die meisten Tore geschossen haben (mehrere Spieler können gleich viele Tore geschossen haben).
   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

Aufgabe 5: Entwurfstheorie

In der folgenden Datenbank sind die Ausleihvorgänge einer Bibliothek gespeichert:

Ausleihe	LNr	Name	Adresse	BNr	Titel	Kategorie	ExemplarNr
	1	Müller	Winklerstr.	1	Datenbanksysteme	Informatik	1
	1	Müller	Winklerstr.	1	Datenbanksysteme	Informatik	2
	2	Huber	Friedrichstr.	2	Anatomie I	Medizin	5
	2	Huber	Friedrichstr.	3	Harry Potter	Literatur	20
	3	Meier	Bismarkstr.	4	OODBS	Informatik	1
	4	Meier	Marktpl.	5	Pippi Langstrumpf	Literatur	1

Für die Datenbank gilt:
Jeder Leser hat eine eindeutige Lesernummer (LNr), einen Namen und eine Adresse. Ein Buch hat eine Buchnummer (BNr), einen Titel und eine Kategorie. Es kann mehrere Exemplare eines Buches geben, welche durch eine, innerhalb einer Buchnummer eindeutigen, Exemplarnummer unterschieden werden.

Fortsetzung nächste Seite!

1. Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können.

2. Nachfolgend sind alle nicht-trivialen funktionalen Abhängigkeiten, welche in der obigen Datenbank gelten, angegeben:

(1) *LNr* → *Name*
(2) *LNr* → *Adresse*
(3) *BNr* → *Titel*
(4) *BNr* → *Kategorie*
(5) *LNr, BNr, ExemplarNr* → *Name, Adresse, Titel, Kategorie*

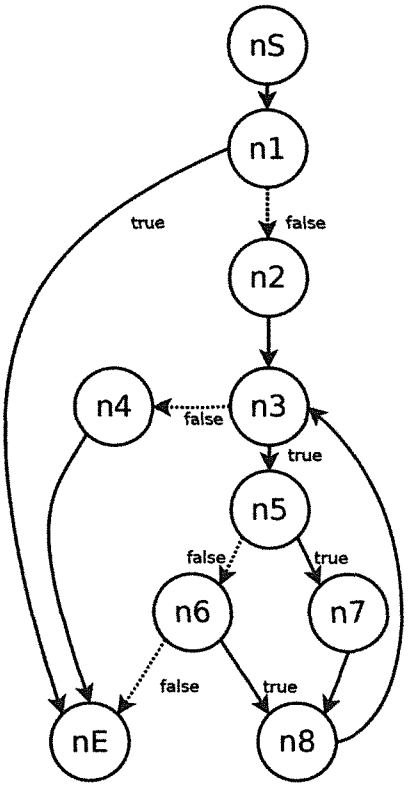
Einziger Schlüsselkandidat ist {*LNr, BNr, ExemplarNr*}.
Überführen Sie das Schema mit Hilfe des Synthesealgorithmus für 3NF in die dritte Normalform.

Teilaufgabe 2

Aufgabe 1: „Testen"

Gegeben sei folgende Methode und ihr Kontrollflussgraph:

```
int binToInt(String bin) {
      if (bin.isEmpty())
            return -1;
      int place = 1, value = 0;
      int length = bin.length() - 1;
      for (int i = length; i >= 0; --i) {
            char ch = bin.charAt(i);
            if (ch == '1') {
                  value += place;
            } else if (ch == '0') {
                  // do nothing
            } else {
                  return -1;
            }
            place *= 2;
      }
      return value;
}
```



a) Geben Sie je einen Repräsentanten aller Pfadklassen ***im Kontrollflussgraphen*** an, die zum Erzielen einer vollständigen

   i. Verzweigungsüberdeckung (Branch-Coverage, C_1)

   ii. Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

mit ***minimaler*** Testfallanzahl genügen würden.

Fortsetzung nächste Seite!

b) Welche der vorangehend ermittelten Pfade für die C_1-Überdeckung sind mittels Testfälle tatsächlich überdeckbar („feasible")? Falls der Pfad ausführbar ist, geben Sie bitte den Testfall (also den Wert des Parameters bin) an – andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

e) Geben Sie zu jedem Knoten die jeweilige Datenfluss-Annotation (defs bzw. uses, sofern überhaupt vorhanden) für jede betroffene Variable in der zeitlichen Reihenfolge ihres Auftretens zur Laufzeit an.

Aufgabe 2: „UML-Klassendiagramm"

Das Warenhaussystem des Online-Händlers Rhein soll modelliert werden. Erstellen Sie dazu ein UML-Klassendiagramm basierend auf den folgenden Angaben. Dabei sind Attribute nur dann einzutragen, wenn es sich um Klassenattribute handelt.

Den Kern des Systems bildet die Verwaltung des Rhein-Systems. Von dieser gibt es immer nur eine Objekt-Instanz. Zum System gehören ein Verkaufsservice und ein Inventar. Verkaufsservice, Inventar und die Objekt-Instanz sollen über öffentliche Getter-Methoden zur Verfügung gestellt werden.
Ein Inventar setzt sich aus einer Liste von Artikeln zusammen. Diese Liste soll über eine Getter-Methode zur Verfügung gestellt werden. Über eine Methode sollen der Liste neue Artikel hinzugefügt werden können.
Artikel stellen die Obermenge der im Webshop angebotenen Waren dar. Eine angebotene Ware muss dabei immer eine weiter spezifizierte Variante von „Artikel" sein. Artikel verfügen über einen Lagervorrat, einen Namen, eine Beschreibung und eine unverbindliche Preisempfehlung. Diese sollen über öffentliche Methoden auslesbar sein. Des Weiteren soll es möglich sein, den Lagerbestand um eine bestimmte Menge zu erhöhen.
Als Varianten des Artikels sind Bücher und DVDs zu spezifizieren. Diese verfügen über alle Eigenschaften von Artikeln, Bücher bieten aber zusätzlich ein Inhaltsverzeichnis und DVDs einen Regionalcode, die jeweils öffentlich per Methode zur Verfügung gestellt werden.
Ein Verkaufsservice verwaltet Angebote und Regeln, die diese Angebote Artikeln zuordnen. Neue Angebote und Regeln sollen hinzugefügt werden können. Für einen Artikel kann das günstigste Angebot abgefragt werden.
Zur Preisbestimmung ist ein Interface „Pricer" zu definieren, welches für einen Artikel einen Preis zurückliefern kann. Für dieses gibt es mehrere Implementierungen: den Standard-Pricer, der die unverbindliche Kaufempfehlung zurückliefert und eine andere Implementierung für die Angebote. Letztere kann zusätzlich noch den ursprünglichen Preis (diese Methode wird erst durch die konkreten, spezialisierten Angebote implementiert) und den prozentualen Preisunterschied zurückgeben.
Als Spezialisierungen der Angebote werden das wahre Angebot und das falsche Angebot bereitgestellt. Das wahre Angebot liefert die unverbindliche Preisempfehlung als ursprünglichen Preis und einen um 20 % reduzierten Preis als den aktuellen Preis des Artikels zurück. Das falsche Angebot liefert die unverbindliche Preisempfehlung als aktuellen Preis zurück und liefert als ursprünglichen Preis eine um 20 % verteuerte unverbindliche Preisempfehlung zurück.

Fortsetzung nächste Seite!

Die Regeln zur Zuordnung zwischen Angeboten und Artikeln werden über ein Interface dargestellt. Dieses bietet eine boolesche Funktion an, die zurückliefert, ob ein Angebot zu einem Artikel passt. Dieses Interface wird von einer Regel für hohen Lagerbestand und einer Regel für niedrigen Preis implementiert.

Aufgabe 3: „Zustandsdiagramme"

a) Erstellen Sie ein Zustandsdiagramm für den im Folgenden beschriebenen Parkautomaten. Modellieren Sie dazu soweit nötig sowohl Zustände und Transitionsbedingungen als auch die Eingangsaktionen der Zustände.

Der Parkautomat startet im bereiten Grundzustand und wartet auf Interaktionen. Wird eine Karte eingelegt, wird diese eingezogen und der Automat wechselt in die Prüfung. In dieser wird die Karte überprüft. Ist sie ungültig, so gibt der Automat die Karte wieder aus, der Vorgang wird abgebrochen und der Parkautomat wechselt wieder in die Bereitschaft. Ist die Karte gültig, so zeigt der Automat den zu zahlenden Preis an und wartet auf die Geldeingabe. Wird Geld eingeworfen, wird, solange der eingeworfene Betrag kleiner als der Preis ist, lediglich der angezeigte Preis aktualisiert und auf weiteres Geld gewartet. Ist der gewünschte Betrag eingeworfen worden, so gibt der Automat eine Parkmünze aus und wartet auf einen eventuellen Quittungswunsch. Erfolgt dieser wird die Quittung ausgegeben und in den Ursprungszustand gewechselt. Erfolgt dieser nicht, wird nach 5 Sekunden in den Ausgangszustand gewechselt.

b) Erweitern Sie das obige Diagramm sinnvoll so, dass wenn ein zu großer Betrag eingeworfen wird, entsprechend Wechselgeld ausgegeben wird.

Fortsetzung nächste Seite!

Aufgabe 4: „Formale Verifikation"

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under-/Overflows auftreten.
Gegeben sei folgende Methode mit Vorbedingung (pre-condition) $P := x \geq 0 \wedge y \geq 0$ und Nachbedingung (post-condition) $Q := x \cdot y = z$:

```
int mul(int x, int y) {
        // P
        int z = 0, i = 0;
        while (i++ != x) z += y;
        // Q
        return z;
}
```

Betrachten Sie dazu die folgenden drei Prädikate:

1. $I_1 := z + i \cdot y = x \cdot y$
2. $I_2 := \mathit{false}$
3. $I_3 := z + (x - i) \cdot y = x \cdot y$

a) Beweisen Sie *formal* (z. B. mit der Methode der schwächsten Vorbedingung) für jedes der drei Prädikate, ob es unmittelbar vor Betreten der Schleife in `mul` gilt oder nicht.

b) Weisen Sie *formal* nach, welche der drei Prädikate Invarianten des Schleifenrumpfs in `mul` sind und welche nicht.

c) Beweisen Sie *formal*, aus welchem der drei Prädikate die Nachbedingung gefolgert werden darf bzw. nicht gefolgert werden kann.

d) Skizzieren Sie den Beweis der totalen Korrektheit der Methode `mul`. Zeigen Sie dazu auch die Terminierung der Methode.