Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

11. Februar 2013

Aufgaben zur Klausur **Softwaredesign** im WS 2012/13 (BInf v310, BMInf v300, BWInf v310, BWInf 23)

Zeit: 90 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Nutzen Sie die Rückseiten der Klausur zur Entwicklung der Lösungen und übertragen die fertigen Lösungen in das Aufgabenblatt.

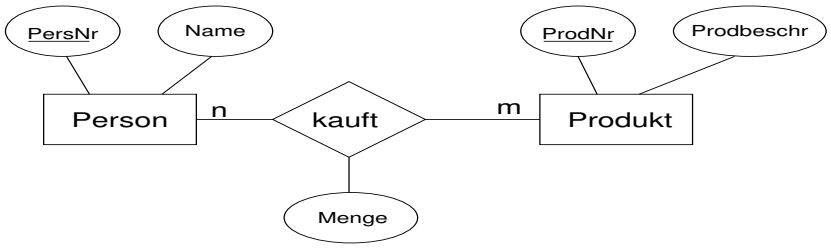
Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 10 Seiten.

Aufgabe 1:

Gegeben sei das folgende ER-Diagram mit den Entity-Typen *Person* und *Produkt* und einer n-m-Beziehung *kauft*. Die Schlüssel-Attribute der Entity-Typen sind entsprechend gekennzeichnet. Die Beziehung besitzt ein Attribut *Menge*.



Entwickeln Sie ein Datenmodell in abstrakter Syntax in Haskell-Notation mit gleicher Semantik wie dieses ER-Diagramm.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 2:

Das Fliegengewicht ist ein Strukturmuster. Welcher Zweck wird mit dem Fliegengewichts–Muster verfolgt?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Wann ist das Muster anwendbar?

1) . . . . . .

2) . . . . . .

3) . . . . . .

4) . . . . . .

5) . . . . . .

Warum erzeugt man Fliegengewichte nicht durch einen direkten Aufruf eines Konstruktors?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 3:

JavaScript ist eine ungetypte Sprache, d.h. es gibt einen universellen Wertebereich und alle Werte aus diesem Wertebereich können in einer Variablen gespeichert werden. In dieser Aufgabe soll so ein Datenmodell für diesen universellen Wertebereich *JSValue* entwickelt werden.

In JavaScript gibt es zwei Spezialwerte *undefined* und *null*, es gibt die beiden booleschen Werte *false* und *true* und es gibt Zahlen. Alle Zahlen werden als doppelt genaue Fließkommazahlen dargestellt. Dafür gibt es (in Haskell) den vordefinierten Datentype *Double*. Weiter gibt es Zeichenreichen (*String*s).

Neben diesen einfachen Datentypen gibt es in JavaScript Objekte. Wenn Objekte in Werten gespeichert werden, so wird nicht das Objekt selbst sondern eine Referenz, d.h. ein Schlüssel zur eindeutigen Identifizierung des Objekts, gespeichert. Für die Modellierung von Referenzen nutzen Sie bitte einen vordefinierten Typ *Ref*.

Für einen JavaScript-Wert gibt es also insgesamt 6 Ausprägungen. Modellieren sie den *JSValue*-Datentyp in Haskell-Notation:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Ein JavaScript-Objekt (*JSObject*) besteht aus einer Menge von Schlüssel-Wert-Paaren. Die Schlüssel sind die sogenannten Properties. Eine *Property* ist nichts anderes als ein String. Die Werte dieser Properties sind beliebige JavaScript-Werte. Alle Objekte werden in einem Objektspeicher (*ObjectStore*) gehalten. Auf die Objekte kann über Referenzen zugegriffen werden.

Modellieren Sie die Datentypen *JSObject* und *ObjectStore* in Haskell-Notation:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 4:

Welches sind die Vorteile von objektbasierten gegenüber klassenbasierten Entwurfsmustern?

1) .....

2) .....

3) .....

Welches sind die Vorteile von klassenbasierten gegenüber objektbasierten Entwurfsmustern?

1) .....

2) .....

3) .....

Aufgabe 5:

Entwickeln Sie ein Datenmodell in Form einer Abstrakten Syntax für die Verarbeitung einer einfachen Programmiersprache. In dieser Sprache soll ein Programm (*Program*) aus einer Anweisung (*Statement*) bestehen. Als Ausprägungen für Anweisungen gibt es Anweisungsfolgen (*Sequence*), Verzweigungen (*IfStatement*), Schleifen (*WhileStatement*) und Zuweisungen (*Assignment*). Eine Zuweisung besteht aus einer Variablen (*Variable*) als linker Seite und einem Ausdruck (*Expr*) als rechter Seite. Ein Ausdruck ist entweder eine Variable (*Variable*), eine Konstante (*Constant*) oder ein zweistelliger Ausdruck (*BinaryExpr*). Eine Variable besteht aus einer Zeichenreihe für den Variablennamen, eine Konstante aus einer ganzen Zahl. Als Operatoren (*Operator*) in einem zweistelligen Ausdruck sollen $+, -, *, div, =$ und $\neq$ erlaubt sein.

Verwenden Sie in dem Datenmodell pro Typdefinition nur einen Typkonstruktor. *String* sei ein vordefinierter Datentyp für Zeichenreihen. Verwenden Sie weiterhin die *kursiv* geschriebenen Namen als Datentypnamen.

Das Datenmodell als abstrakte Syntax in Haskell-Notation:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Die Abstrakte Syntax transformiert in ein gleichwertiges OMT–Diagramm:

Welche Strukturmuster werden in diesem Datenmodell verwendet? Nennen Sie den jeweiligen Musternamen, die beteiligten Klassen und die beteiligten Referenzen.

1. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

2. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

3. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

4. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .