Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

12. August 2014

Aufgaben zur Klausur **Softwaredesign** im SS 2014 (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.

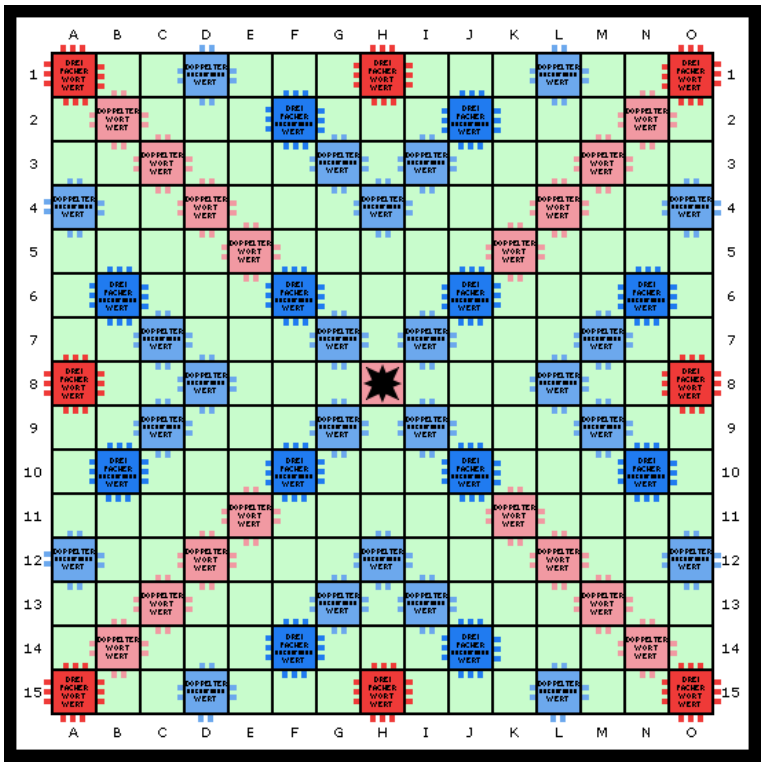
Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 11 Seiten.

Aufgabe 1:

In dieser Aufgabe geht es darum, ein Scrabble-Spiel zu beschreiben. Scrabble ist ein Brettspiel. Auf das Spielfeld können waagerecht und senkrecht Wörter gelegt werden. Jede zusammenhängende Folge von Wörtern, egal ob waagerecht oder senkrecht, muss ein Wort aus einem vorgegebenen Wörterbuch sein.

Ein Brett kann folgende Gestalt haben:



Dieses ist das Layout des Original-Bretts. In dem Modell in dieser Aufgabe soll es aber möglich sein, mit unterschiedlichen Brettgrößen, auch mit beliebigen Brettformen, also nicht nur mit quadratischen oder rechteckigen Brettern zu spielen. Jedes Feld auf dem Brett wird durch eine Koordinate eindeutig bestimmt.

Ein Brett hat statische Eigenschaften, die Größe und Form, außerdem wird zur Bewertung von gelegten Wörtern jedes Feld mit einem Wert markiert. Wie im Original sollen hier fünf Markierungen möglich sein, der normale Buchstabenwert, ein doppelter oder ein dreifacher Buchstabenwert oder ein doppelter oder dreifacher Wortwert. Diese Markierungen werden zur Berechnung der Punktzahl eines Zuges benötigt. Diese Werte sollen in diesem Modell den Positionen frei zugeordnet werden können, also nicht genau der Belegung aus dem Original entsprechen.

Der Zustand eines Brettes ist gerade die aktuelle Belegung des Brettes, er ändert sich bei jedem Zug mit dem Legen von neuen Buchstaben.

Weiter gehört Zu einem Spiel eine Menge von Spielsteinen, ein Alphabet. Dieses Alphabet soll eine Teilmenge der zur Verfügung stehenden Zeichen (*Char*) sein. Jedes Zeichen aus dem Alphabet bekommt einen (Buchstaben-)wert, mit dem zum Beispiel das Legen von Wörtern mit selten vorkommenden Buchstaben belohnt werden kann. Das Alphabet und die Bewertung der Buchstaben soll ebenfalls konfigurierbar sein. Aus den Steinen müssen Wörter gebildet werden, diese müssen zulässig sein, also in einem Wörterbuch stehen.

Zu einem Spiel gehört ein Brett, ein Sack mit zu verteilenden Steinen und beliebig viele Spieler. Diese besitzen alle einen eindeutigen Namen. Ein Spieler hat immer eine gewisse Menge von Steinen (Buchstaben) zur Verfügung. Diese werden nach jedem Zug aus dem Sack aufgefüllt. Außerdem ist jedem Spieler eine Punktzahl zugeordnet, die aus den bisher gelegten Wörtern errechnet wird.

Entwickeln Sie ein Datenmodell für dieses Spiel in Haskell-Notation. Gehen Sie dabei schrittweise vor und beschreiben Sie bitte als erstes die statischen Eigenschaften:

Das Alphabet und der Wert eines Steins

. . . . . .

. . . . . .

. . . . . .

Das Brett mit Layout und Wert eines Felds

. . . . . .

. . . . . .

. . . . . .

Das Wörterbuch

. . . . . .

. . . . . .

. . . . . .

Das gesamte statische Modell

. . . . . .

. . . . . .

. . . . . .

Der zweite Teil besteht aus der Beschreibung des Zustands, also dem Teil eines Spiels, das sich bei jedem Zug ändert.

Der Brettzustand

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Die Spieler

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Der Sack mit den Spielsteinen

. . . . . .

. . . . . .

. . . . . .

Der gesamte Zustand

. . . . . .

. . . . . .

. . . . . .

Aufgabe 2:

Gegeben sei der folgende Teil eines Datenmodells für einen CD–Katalog:

Das Modell in abstrakter Syntax in Haskell–Notation:

.0 data $Sammlung \quad = Sammlung\,[\,Album\,]\;\; Attribute$

.1 data $Album \quad = Album\,[\,CD\,]\;\; Attribute$

.2 data $CD \quad = CD\,[\,Stueck\,]\;\; Attribute$

.3 data $Stueck \quad = Stueck\;\; Attribute$

.4 type $Attribute \quad = \mathsf{Map}\;\; Attr\;\; Wert$

.5 type $Attr \quad = String$

.6 type $Wert \quad = String$

In diesem Datenmodell ist die Hierarchie streng festgelegt: Es gibt immer genau vier Stufen in der Hierachie: Sammlung, Album, CD, Stück. Dieses ist für die Strukturierung eines Kataloges unflexibel. Zum Beispiel kann man Sammlungen nicht in Teilsamlungen aufgliedern, eine Sammlung kann auch keine CDs und Alben gleichzeitig enthalten.

Entwickeln Sie ein flexibleres Datenmodell, in dem beliebige Hierarchien von Sammlungen, Alben, CDs und Stücken möglich sind, es aber nur diese vier Arten von Knoten gibt.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

Welche zusätzlichen Konsistenzbedingungen (Datenstruktur–Invarianten) müssen für diese Datenstruktur eingehalten werden, damit ein sinnvoller CD–Katalog entsteht.

1) .....

2) .....

3) .....

4) .....

5) .....

Aufgabe 3:

Gegeben sei das folgende Datenmodell in abstrakter Syntax nach Haskell:

.0 type $Map_1 \quad = \mathsf{Map}\ K_1\ \ (A_1, Map_2, List_2)$

.1 type $List_2 \quad = [\,(A_2, A_3)\,]$

.2 type $Map_2 \quad = \mathsf{Map}\ K_2\ \ Attr_2$

.3 type $Attr_2 \quad = [\,K_1\,]$

Die Datentypen K_1, K_2, A_1, A_2 und A_3 seien dabei einfache unstrukturierte Datentypen.

Transformieren Sie dieses hierarchische Modell (Map_1) in eines, das für die Implementierung mit Hilfe relationaler Datenbanken geeignet ist, d.h. in ein Modell, das nur noch aus Relationstypen besteht, die sich in 1. Normalform befinden, bei denen also alle Attribute unstrukturierte Typen besitzen.

Tipp: Eine n–stellige Relation mit k Schlüsselkomponenten wird in abstrakter Syntax als Map modelliert.

$Rel_n \ = \ \mathsf{Map}\ (T_1, \ldots, T_k)\ \ (T_{k+1}, \ldots, T_n)$

Sind alle Attribute Teil des Schlüssels so ergibt sich der Typ

$Rel_n \ = \ \mathsf{Set}\ (T_1, \ldots, T_n)$

Beachten Sie, dass in dem Relationenmodell keine Redundanzen entstehen.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

Aufgabe 4:

Entwickeln Sie zu der unten angegebenen kontextfreien Grammatik eine abstrakte Syntax in Haskell Notation. Die Grammatik beschreibt die konkrete Syntax eines Teils der Anweisungen der Sprache Java.

Die konkrete Syntax sei durch folgende in BNF–Notation gegebene kontextfreie Grammatik beschrieben. Dabei sind Terminalsymbole in ' gesetzt. Des weiteren ist $Ident$ ein Terminalsymbol, das aus einem Namen besteht. $Expr$ ist ein Nichtterminalsymbol für Ausdrüche. $Expr$essions sollen durch einen gleichnamigen Datentyp in der abstrakten Syntax repräsentiert werden. Dieser Typ soll hier als gegeben angenommen werden.

.0	$Stmt$	::= $ExprStmt$ | $Return$ | $Block$ | $Throw$ | Try
.1	$ExprStmt$	::= $Expr$ ';'
.2	$Return$	::= 'return' $Expr_0$ ';'
.3	$Block$	::= '{' $StmtList$ '}'
.4	Try	::= 'try' $Stmt$ $Catches$
.5	$Throw$	::= 'throw' $Expr$ ';'
.6	$Expr_0$	::= | $Expr$
.7	$StmtList$	::= | $StmtList$ $Stmt$
.8	$Catches$	::= | $Catch$ $Catches$
.9	$Catch$	::= 'catch' '(' $Ident$ $Ident$ ')' $Stmt$

In dem Sprachteil gibt es also Ausdrucksauswertung, return –Anweisungen, Blöcke, try–Anweisungen und throw –Anweisungen.

Entwickeln Sie für diesen Sprachteil eine abstrakte Syntax in Haskell Notation. Versuchen Sie durch Abstraktion ein möglichst einfaches Modell mit wenigen Datentypen zu entwickeln. Verwenden Sie keine geschachtelten Typdefinitionen. Die Datentypen für $Expr$ und $Ident$ seinen vordefiniert.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

11) .....

12) .....

Welche Strukturmuster findet man in diesem Datenmodell wieder? Geben Sie zu den Mustern die beteiligten Typ- bzw. Konstruktornamen an.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

Aufgabe 5:

Welche Strukturmuster sind geeignet, um rekursive Datenstrukturen zu modellieren?

1) .....

2) .....

3) .....

4) .....

5) .....

Aufgabe 6:

Welche Verhaltensmuster sind geeignet, um die Verarbeitung von rekursiven Datenstrukturen zu modellieren?

1) .....

2) .....

3) .....

4) .....

5) .....
