Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

30. Januar 2009

Aufgaben zur Klausur **Softwaredesign** im WS 2008/09 (WI h253, MI h405, BInf v310, BMInf v300, BWInf v310)

Zeit: 75 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.

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

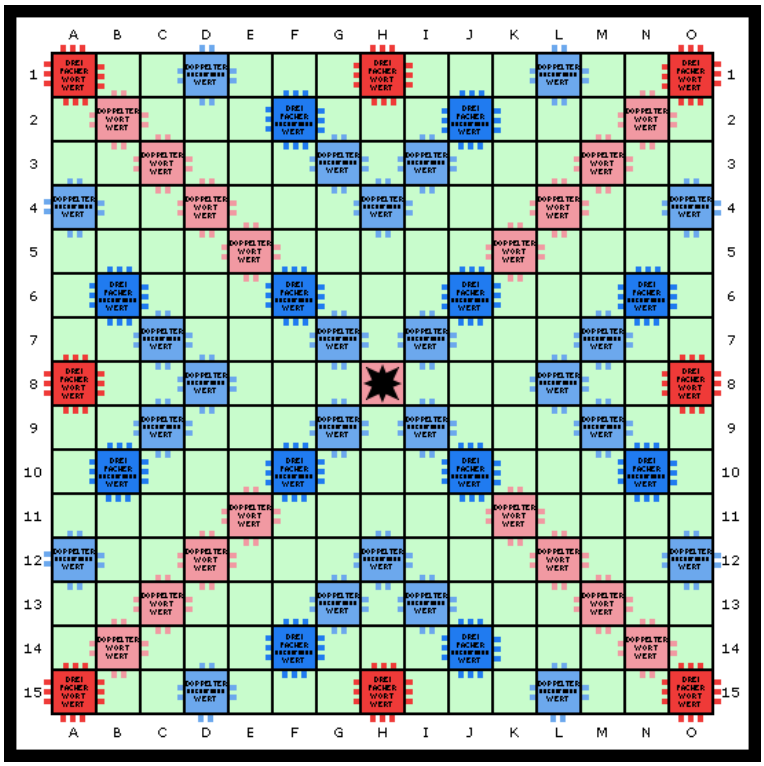
Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 9 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:

Ein gerichteter Graph, bei dem die Knoten durch einen Wertebereich *NodeId* eindeutig identifiziert werden und sowohl an den Knoten Zusatzinformation vom Typ *NodeAttr* als auch an den Kanten Zusatzinformation der Art *EdgeAttr* enthalten ist, kann durch folgende Datentypen modelliert werden.

.0 type $Graph$ $= \mathsf{Map}\ NodeId\ \ NodeInfo$

.1 type $NodeInfo$ $= (NodeAttr, Succ)$

.2 type $Succ$ $= \mathsf{Map}\ NodeId\ \ EdgeAttr$

Transformieren Sie dieses Modell in ein neues, das geeignet ist, 1–1 in Datenbankschemata für ein relationales Datenbanksystem umgesetzt zu werden. Das Modell muss also in eine Menge von Relationstypen umgesetzt werden, deren Elementbereiche unstrukturiert sind oder aus Tupeln von unstrukturierten Bereichen besteht.

Nehmen Sie an, dass *NodeId* ein unstrukturierter Datentyp ist und *NodeAttr* und *EdgeAttr* folgendermaßen definiert sind, wobei die A_i ebenfalls alle unstrukturiert sind.

.0 type $NodeAttr$ $= (A_1, [\,A_2\,])$

.1 type $EdgeAttr$ $= (A_3, A_4)$

Das transformierte Datenmodell in Haskell-Notation

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

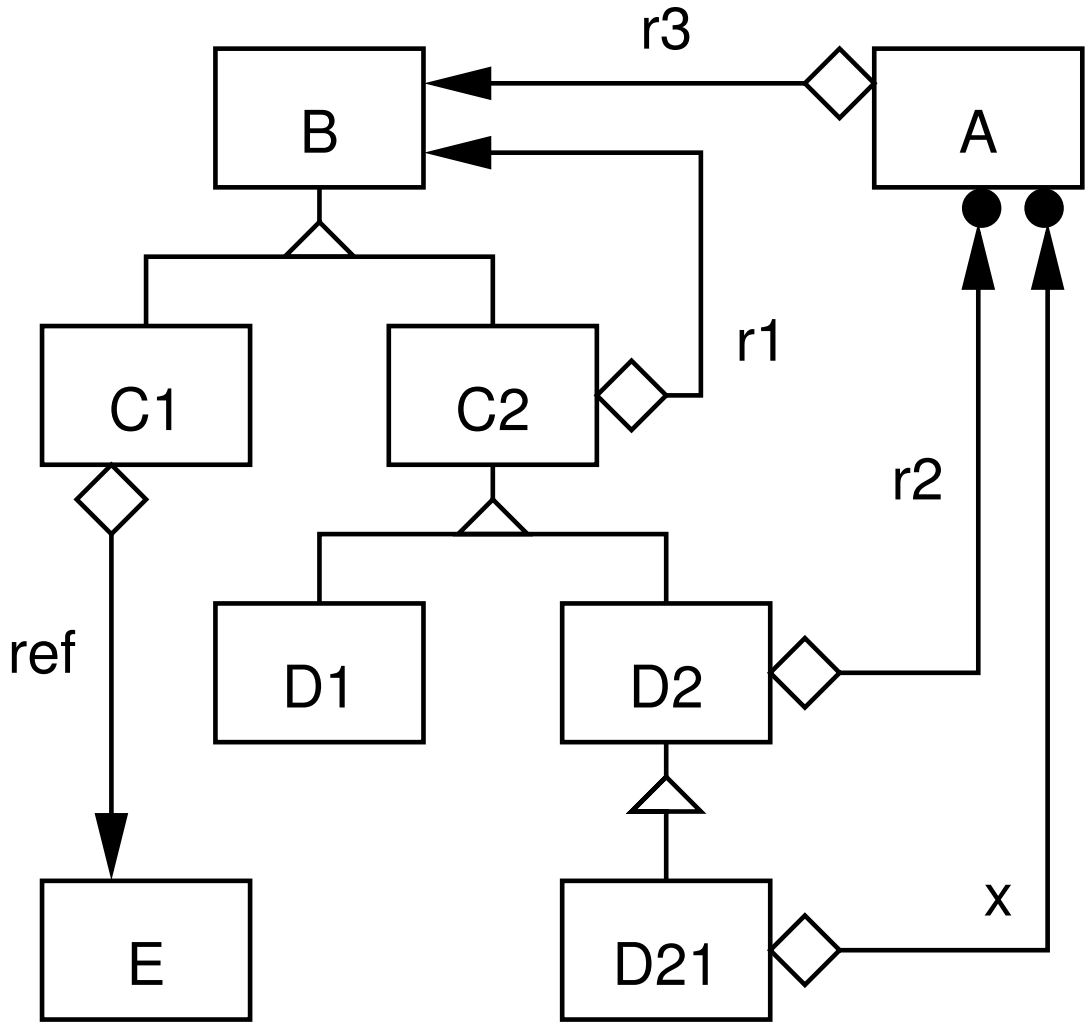
8) .....

9) .....

10) .....

Aufgabe 3:

Gegeben sei das folgende OMT-Diagramm:



Welche Strukturmuster sind in diesem Diagramm enthalten? Geben sie die Muster und die an den einzelnen Mustern beteiligten Klassen und Referenzen an.

1. .....

.....

2. .....

.....

3. .....

.....

4. .....

.....

5. .....

.....

6. .....

.....

7. .....

.....

8. .....

.....

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:

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

1) .....

2) .....

3) .....

4) .....

5) .....
