Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

24. Januar 2012

Aufgaben zur Klausur **Softwaredesign** im WS 2011/12 (BInf v310, BMInf v300, BWInf v310)

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.

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 8 Seiten.

Aufgabe 1:

Gegeben sei das folgende Datenmodell in abstrakter Syntax nach Haskell:

.0 type $Map_1 \quad = \mathsf{Map}\ K_1\ \ (Attr_1, Map_2)$

.1 type $Attr_1 \quad = (A_1, List_1)$

.2 type $List_1 \quad = [\,(A_2, A_3)\,]$

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

.4 type $Attr_2 \quad = \mathsf{Set}\ K_1$

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

Transformieren Sie dieses Modell 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 Tabelle 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 2:

Grafische Oberflächen werden sinnvollerweise mit Hilfe einer GUI-Bibliothek implementiert. So eine BUI-Bibliothek stellt im allgemeinen ein sogenanntes Widgetset zur Verfügung. Hierunter versteht man eine Hierachie von Klassen für verschiedene Arten von grafischen Elementen, deren gemeinsame Eigenschaften, die einheitliche Schnittstelle, in einer Oberklasse festgelegt werden.

In dieser Aufgabe soll ein Ausschnitt eines solchen Widgetsets modelliert werden. Dabei soll es folgende Arten von *Widget*s geben.

1. Ein *Label* zur Anzeige eines Textes.
2. Ein *Button* mit einem Text als Beschriftung.
3. Ein *Text*feld mit einer Anzahl von Zeilen und Spalten und einem editierbaren Text (*String*).
4. Ein Menü (*Menue*) bestehend aus einer Liste von Menüeinträgen (*MenueEntry*). Dabei kann ein Eintrag zwei Ausprägungen besitzen, einmal kann er ein einfacher *Button* sein, zum anderen ein benanntes Untermenü.
5. Eine Menueleiste (*MenueBar*) besteht aus einer Liste von benannten Menüs.
6. Zum strukturierten Anordnen von *Widget*s sollen zwei Container zur Verfügung stehen.
   (a) Ein Rahmen (*Frame*) zur Anordnung einer geordneten Menge von Teil-*Widget*s. Dabei soll jeder Eintrag markiert werden mit einer der vier Himmelsrichtungen (Nord, Ost, Süd, West). Dieses Attribut bestimmt die Anordnung auf dem Anzeigegerät.
   (b) Eine Tabelle (*Table*) die in jeder Zelle ein Unter-*Widget* aufnehmen kann.

Entwickeln Sie für diese Struktur eine abstrakte Syntax in Haskell-Notation. Verwenden Sie in dem Datenmodell pro Typdefinition nur einen Typkonstruktor. *String* sei ein vordefinierter Datentyp für Zeichenreihen, *Int* für ganze Zahlen.

Die abstrakte Syntax für *Widget*:

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

11) .....

12) .....

13) .....

14) .....

15) .....

Aufgabe 3:

Entwickeln Sie zu einer kontextfreien Grammatik für den Anweisungsteil einer einfachen Programmiersprache eine abstrakte Syntax für die interne Repräsentation und Verarbeitung dieser Anweisungen.

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 $Stmt$ ::= $Assign$

.1 $Stmt$ ::= $Stmtlist$

.2 $Stmt$ ::= $IfStmt$

.3 $Stmt$ ::= $WhileStmt$

.4 $Stmt$ ::= $DoStmt$

.5 $Assign$ ::= $Ident$ ':=' $Expr$

.6 $StmtList$ ::= | $StmtList$ ';' $Stmt$

.7 $IfStmt$ ::= 'if' $Expr$ 'then' $Stmt$ $ElsePart$ 'fi'

.8 $ElsePart$ ::= | 'else' $Stmt$

.9 $WhileStmt$ ::= 'while' $Expr$ 'do' $Stmt$ 'done'

.10 $DoStmt$ ::= 'do' $Stmt$ 'until' $Expr$

In der Sprache gibt es also Zuweisungen, Anweisungsfolgen, Verzweigungen, while– und do–Schleifen.

Entwickeln Sie für diese Sprache eine abstrakte Syntax in Haskell Notation. Versuchen Sie durch Abstraktion ein möglichst einfaches Modell mit wenigen Datentypen zu entwickeln. 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?

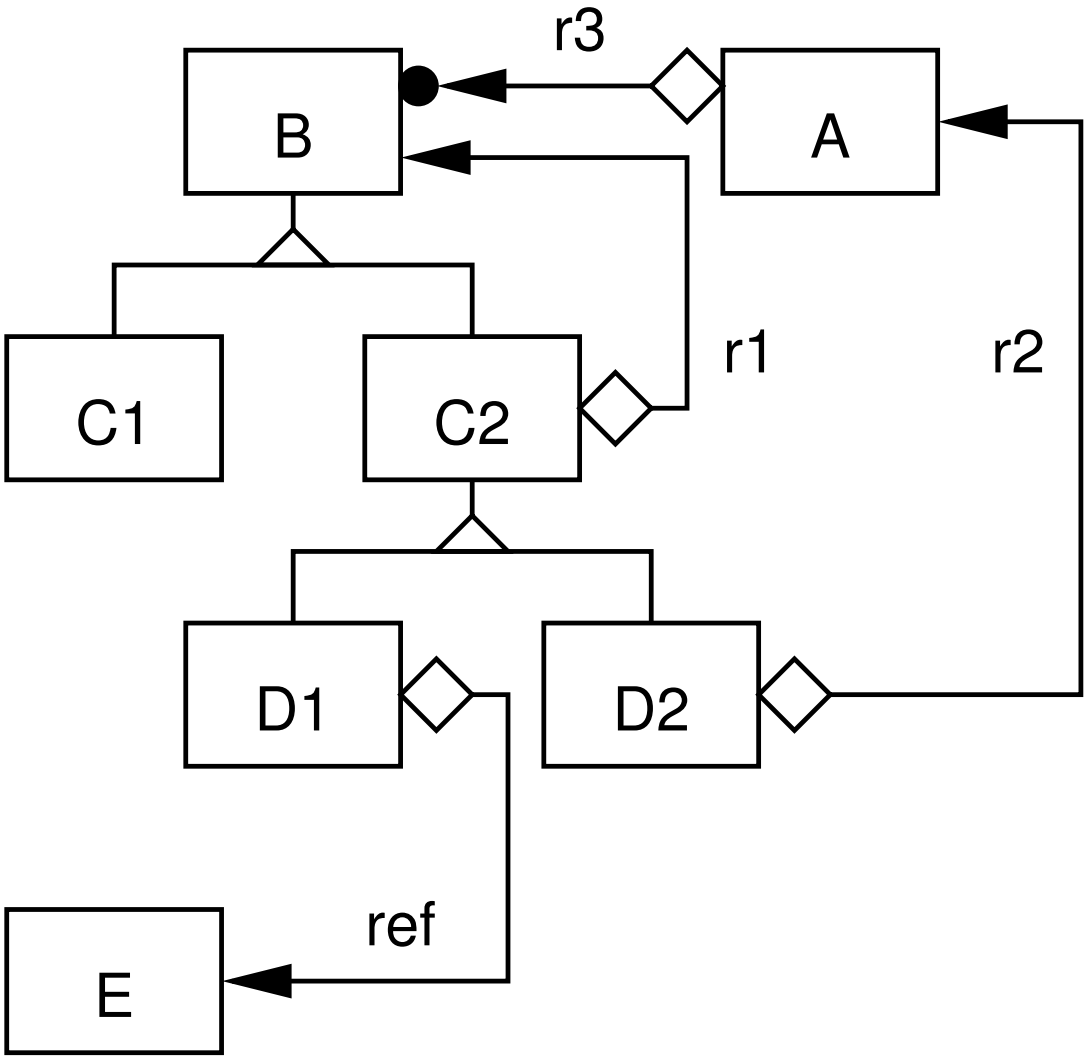
1) .....

2) .....

3) .....

Aufgabe 4:

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. .....

   .....

   .....
