Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

7. Februar 2007

Aufgaben zur Klausur **Softwaredesign** im WS 2006/07 (WI h252, WI h253, II h752, MI h403, MI h404, MI h405, 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 11 Seiten

---

Aufgabe 1:

Entwickeln Sie zu einer kontextfreien Grammatik für 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 (maximal 3) 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?

1) .....

2) .....

3) .....

Welche Verhaltensmuster sind zur Verarbeitung dieser Strukturen geeignet?

1) .....

2) .....

3) .....

Aufgabe 2:

Klassifizieren Sie das Iterator–Muster.

. . . . . .

. . . . . .

Welches ist der Zweck des Iterator–Musters?

. . . . . .

. . . . . .

Wo ist das Muster typischerweise anwendbar?

. . . . . .

. . . . . .

. . . . . .

Warum ist das Muster nicht geeignet für die Verarbeitung von Komposita?

. . . . . .

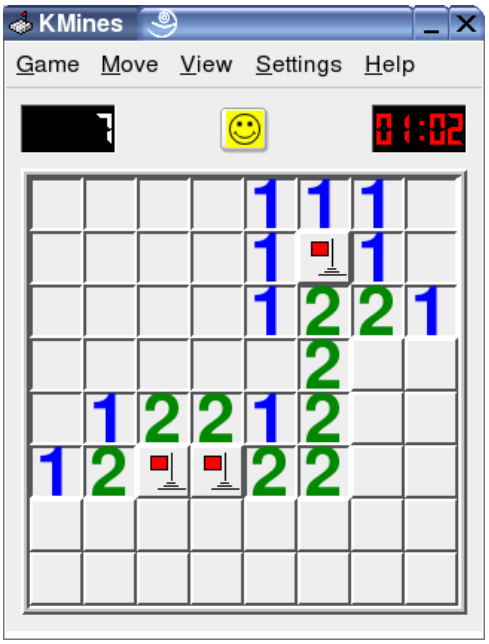
. . . . . .

. . . . . .

Geben Sie das Klassendiagramm für das Iterator–Muster an:

Aufgabe 3:

Für das Spiel MineSweeper (und Varianten davon) ist ein Datenmodell für die Spielzustände auf dem Feld zu modellieren. Das Bild zeigt eine Spielsituation:



In diesem Spiel geht es darum, auf einem $n \times m$ Feld sogenannte Bomben zu finden, ohne diese versehentlich aufzudecken. Der Zustand eines Feldes besteht also einmal aus der Information, ob dieses Feld aufgedeckt worden ist, ob es noch nicht aufgedeckt wurde, oder ob es mit einer Fahne als Bombenfeld markiert worden ist. Weiter enthält ein Feld die Information, ob eine Bombe darin plaziert wurde oder, wenn nicht, wie viele Bomben in den angrenzenden Feldern gelagert sind.

Das Datenmodell in abstrakter Syntax in Haskell Notation. Verwenden Sie pro Typdefinition nur einen Typkonstruktor.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 4:

Gegeben sei das folgende Datenmodell in abstrakter Syntax nach Haskell:

.0 type M_1 $= \mathsf{Map}\ K_1\ (M_2, A_3)$

.1 type M_2 $= \mathsf{Map}\ K_2\ (A_2, S_3)$

.2 type S_3 $= \mathsf{Set}\ String$

.3 type K_1 $= Int$

.4 type K_2 $= String$

.5 type A_2 $= Float$

.6 type A_3 $= String$

Transformieren Sie dieses Modell in eines in 1.Normalform, also in eines, das nur noch aus einer Sammlung von Relationstypen besteht, bei denen alle Attribute unstrukturierte Wertebereiche haben. Eine n–stellige Relation ist eine Menge von n-Tupeln. *String* wird in dieser Aufgabe als unstrukturierter Typ betrachtet. In Haskell Syntax wird ein Relationstyp nach folgendem Muster definiert:

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

Beachten Sie, dass in dem Relationenmodell keine Redundanzen entstehen.

1) .....

2) .....

3) .....

4) .....

5) .....

Aufgabe 5:

Abstrakte Fabrik und Prototyp sind zwei Muster mit ähnlichem Zweck.

Wann ist es günstiger, das Prototyp–Muster zu verwenden?

1) .....

2) .....

3) .....

Wann ist es günstiger, eine abstrakte Fabrik zu verwenden?

1) .....

2) .....

3) .....
