Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

18. August 2004

Aufgaben zur Klausur **Softwaredesign** im SS 2004 (WI h252, WI h253, II h752, MI h403, MI h404, MI h405)

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 13 Seiten

---

Aufgabe 1:

Gegeben sei das folgende Datenmodell für einen Wortindex für Freitextsuche in Dokumenten:

Das Modell in Abstrakter Syntax nach VDM:

.0	$Index$	$= \mathsf{map}\ Wort\ \mathsf{to}\ Positionen$
.1	$Positionen$	$= \mathsf{set\ of}\ (DokumentName \times Pos)$
.2	$Wort$	$= String$
.3	$DokumentName$	$= String$
.4	Pos	$= \mathsf{N}_0$

Das Modell in Abstrakter Syntax nach Haskell:

.0	$Index$	$= \mathsf{Map}\ Wort\ Positionen$
.1	$Positionen$	$= \mathsf{Set}\ (DokumentName, Pos)$
.2	$Wort$	$= String$
.3	$DokumentName$	$= String$
.4	Pos	$= \mathsf{N}_0$

Beim Auswerten von Anfragen über einem Index werden häufig Mengenoperationen auf den *Positionen* ausgewertet. Hierbei wird häufig eine Projektion auf die *DokumentName*n benötigt. Außerdem werden Mengenoperationen benötigt, die nur die *DokumentName*n–Komponente berücksichtigen, z.B. bei UND-Anfragen, bei denen Dokumente gesucht werden, die zwei bestimmte Wörter enthalten.

Verfeinern Sie dieses Datenmodell so, dass die Mengenoperationen bei der Anfrageauswertung effizienter implementiert werden können.

Das verfeinerte Datenmodell:

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

Aufgabe 2:

Entwickeln Sie zu einer gegebenen Anfragesprache für eine Freitextsuche in einem Dokumentenindex eine abstrakte Syntax. Die konkrete Syntax sei durch folgende in BNF–Notation gegebene kontextfreie Grammatik beschrieben. Dabei sind Terminalsymbole in ' gesetzt.

.0 $Query \quad ::= Query\ 'OR'\ Query_1 \mid Query_1$

.1 $Query_1 \quad ::= Query_1\ 'AND'\ Query_2 \mid Query_2$

.2 $Query_2 \quad ::= Query_3\ 'NEAR'\ Query_2 \mid Query_3$

.3 $Query_3 \quad ::= Query_4\ Query_3 \mid Query_4$

.4 $Query_4 \quad ::= '('\ Query\ ')' \mid Query_5$

.5 $Query_5 \quad ::= Query6 \mid Query_6\ '...'$

.6 $Query_6 \quad ::= Word$

In der Anfragesprache gibt es also Und– und Oder–Verknüpfungen, es gibt eine Kontext–Suche, in der ein Wort in der Nähe eines anderen gesucht werden soll, man kann nach Folgen von Wörtern suchen, nach einfachen Wörtern und nach Präfixen (Operator '...'). Wie ein Wort dabei aufgebaut sein soll, bleibt hier offen, ein Wort wird durch eine beliebige Zeichenreihe repräsentiert.

Entwickeln Sie für diese Sprache eine Abstrakte Syntax für die interne Verarbeitung von Anfragen. Versuchen Sie durch Abstraktion ein möglichst einfaches Modell zu entwickeln.

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 3:

Gegeben sei das folgende Datenmodell:

Das Modell in Abstrakter Syntax nach VDM:

.0 $M_1 \quad = \mathsf{map}\ K_1\ \mathsf{to}\ M_2$

.1 $M_2 \quad = \mathsf{map}\ K_2\ \mathsf{to}\ (A_2 \times S_3)$

.2 $S_3 \quad = \mathsf{set\ of}\ String$

.3 $K_1 \quad = \mathsf{N}_1$

.4 $K_2 \quad = \mathsf{N}_0$

.5 $A_2 \quad = String$

Das Modell in Abstrakter Syntax nach Haskell:

.0 $M_1 \quad = \mathsf{Map}\ K_1\ M_2$

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

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

.3 $K_1 \quad = \mathsf{N}_1$

.4 $K_2 \quad = \mathsf{N}_0$

.5 $A_2 \quad = String$

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.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

Aufgabe 4:

Gegeben sei das folgende Datenmodell:

Das Modell in Abstrakter Syntax nach VDM:

.0 $Table = Leaf \mid Entry \mid Switch \mid Single \mid Empty$

.1 $Leaf = String \times Attr$

.2 $Entry = Table \times Attr$

.3 $Switch = \mathsf{map}\ Char\ \mathsf{to}\ Table$

.4 $Single = Char \times Table$

.5 $Empty = ()$

.6 $String = Char^*$

.7 $Attr = \mathsf{set\ of}\ \mathsf{N}_0$

Das Modell in Abstrakter Syntax nach Haskell:

.0 $Table = Leaf\ \ String\ \ Attr$

.1 $\mid\ \ Entry\ \ Table\ \ Attr$

.2 $\mid\ \ Switch\ \ (\mathsf{Map}\ Char\ Table)$

.3 $\mid\ \ Single\ \ Char\ \ Table$

.4 $\mid\ \ Empty$

.5 $String = [Char]$

.6 $Attr = \mathsf{Set}\ \mathsf{N}_0$

Welche Strukturmuster kommen in diesem Modell vor?
Geben Sie jeweils den Musternamen und die beteiligten Datentypen an.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

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) .....

Aufgabe 6:

Bei dem Erzeugen von Objekten durch einen Prototyp spielt das Kopieren (Klonen) von Objekten eine wichtige Rolle.

Gegeben sei das folgende Java–Programmfragment:

```
public class PrototypeFactory {

    public X makeX() {
        return pt.copy();
    }

    private final
        X pt = new X(new Y(new Z(3),4),5);
}

class X {
    Y d1;
    int d2;
    X(Y d1, int d2) {
        this.d1 = d1;
        this.d2 = d2;
    }
    X copy() {
        return ...;
    }
    ...
}

class Y {
    Z d1;
    int d2;
    Y(Z d1, int d2) {
        this.d1 = d1;
        this.d2 = d2;
    }
    Y copy() {
        return ...;
    }
    ...
}

class Z {
    int d2;
    Z(int d2) {
        this.d2 = d2;
```

```
    }
    Z copy() {
        return ...;
    }
    ...
}
```

Wieviele neue Objekte müssen beim Kopieren des Prototypen in einem Aufruf von *makeX* erzeugt werden, wenn ...

1. ... die Variablen d1 und d2 in X über Methoden verändert werden können, für alle anderen Variablen in den übrigen Klassen dieses aber nicht möglich ist.

   .....

2. ... die Variable d2 in Z über Methoden verändert werden kann, für alle anderen Variablen in den übrigen Klassen dieses aber nicht möglich ist.

   .....

3. ... die Variablen d1 und d2 in Y über Methoden verändert werden können, für alle anderen Variablen in den übrigen Klassen dieses aber nicht möglich ist.

   .....

4. ... keine der Variablen in den Klassen X, Y und Z über Methoden verändert werden können.

   .....

5. ... nur die Variable d1 in X über Methoden verändert werden kann, für alle anderen Variablen in den Klassen X, Y und Z dieses aber nicht möglich ist.

   .....
