Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

1. Februar 2008

Aufgaben zur Klausur **Softwaredesign** im WS 2007/08 (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:

Worin besteht der softwaretechnische Nutzen bei der Verwendung von Erzeugungsmustern?

Präzise Stichworte (bitte keine Allgemeinplätze und Phrasen):

1) .....

2) .....

3) .....

4) .....

5) .....

Aufgabe 2:

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

Das Modell in abstrakter Syntax nach Haskell:

.0	$Index$	$= \mathsf{Map}\ Wort\ Positionen$
.1	$Positionen$	$= \mathsf{Set}\ Position$
.2	$Position$	$= (DokumentName, Pos)$
.3	$Wort$	$= String$
.4	$DokumentName$	$= String$
.5	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 3:

Gegeben sei die folgende Datenstruktur in Haskell–Notation:

```
.0  data Tree a    = Leaf a
.1                 | Chain a (Tree a)
.2                 | Fork a (Tree a) (Tree a)
.3                 | Triple a (Tree a) (Tree a) (Tree a)
```

Welche Strukturmuster sind in dieser Datenstruktur enthalten?

1) .....

2) .....

3) .....

4) .....

Vereinfachen (verallgemeinern) Sie diese Datenstruktur so, dass nicht mehr mit den vier verschiedenen Varianten (in OOP/OOD: Nicht mehr mit Vererbung) gearbeitet werden muss. Bitte benutzen Sie pro Typdefinition nur einen Typkonstruktor.

Das vereinfachte Datenmodell:

.....

.....

.....

.....

Aufgabe 4:

Modellieren Sie eine einfache Form einer (Struktur–) Stückliste. In diesem Modell werden alle Produkte durch eine Produktidentifikation (*Pid*) eindeutig bestimmt. Dieser Datentyp sei vordefiniert.

Alle Produkte werden in einer Produkttabelle gespeichert. Zu jeder *Pid* wird zusätzlich gespeichert, aus welchen Teilprodukten dieses Produkt zusammengesetzt wird und wieviele dieser Teilprodukte dafür benötigt werden. Entwerfen Sie hierfür einen Datentyp *Stueckliste*. Verwenden Sie pro Typgleichung bitte nur einen Typkonstruktor.

1) .....

2) .....

3) .....

4) .....

5) .....

Woran erkennt man, dass ein Produkt einfach (nicht zusammengesetzt) ist?

.....

Welche zusätzlichen Konsistenzbedingungen müssen für diese Stückliste erfüllt sein.

1) .....

2) .....

3) .....

4) .....

Wie muss man dieses Modell erweitern, dass zu jedem Produkt auch noch eine Produkt–*Beschreibung* gespeichert werden kann?

1) .....

2) .....

3) .....

4) .....

Aufgabe 5:

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 ...;
    }
    ...
}
```

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

1. ... 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.

   .....

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

   .....

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. ... 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.

   .....

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

   .....
