

Aufgaben zur Klausur **Software design** im SS 2009 (WI h252, WI h253, II h752, MI h403, MI h404, 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 8 Seiten.

---

**Aufgabe 1:**

Gegeben sei das folgende Datenmodell in abstrakter Syntax in Haskell-Notation:

```
.0 data Table      = Leaf String Attr
.1                | Entry Table Attr
.2                | Switch CMap
.3                | Single Char Table
.4                | Empty
.5 type CMap      = Map Char Table
.6 type String   = [Char]
.7 type Attr     = Set Int
```

Welche Strukturmuster kommen in diesem Modell vor?

Geben Sie jeweils den Musternamen und die beteiligten Datentypen und Kostruktornamen an.

- 1) .....
- 2) .....
- 3) .....
- 4) .....
- 5) .....
- 6) .....
- 7) .....
- 8) .....

## Aufgabe 2:

Entwickeln Sie zu der unten angegebenen kontextfreien Grammatik eine abstrakte Syntax in Haskell Notation. Die Grammatik beschreibt die konkrete Syntax eines Teils der Anweisungen der Sprache Java.

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 Ausdrücke. *Expressions* sollen durch einen gleichnamigen Datentyp in der abstrakten Syntax repräsentiert werden. Dieser Typ soll hier als gegeben angenommen werden.

- .0 *Stmt* ::= *ExprStmt* | *Return* | *Block* | *Throw* | *Try*
- .1 *ExprStmt* ::= *Expr* ';'
- .2 *Return* ::= 'return' *Expr*<sub>0</sub> ';'
- .3 *Block* ::= '{' *StmtList* '}'
- .4 *Try* ::= 'try' *Stmt* *Catches*
- .5 *Throw* ::= 'throw' *Expr* ';'
- .6 *Expr*<sub>0</sub> ::= | *Expr*
- .7 *StmtList* ::= | *StmtList Stmt*
- .8 *Catches* ::= | *Catch Catches*
- .9 *Catch* ::= 'catch' '(' *Ident Ident* ')' *Stmt*

In dem Sprachteil gibt es also Ausdrucksauswertung, `return`–Anweisungen, Blöcke, `try`–Anweisungen und `throw`–Anweisungen.

Entwickeln Sie für diesen Sprachteil eine abstrakte Syntax in Haskell Notation. Versuchen Sie durch Abstraktion ein möglichst einfaches Modell mit wenigen Datentypen zu entwickeln. Verwenden Sie keine geschachtelten Typdefinitionen. Die Datentypen für *Expr* und *Ident* seien vordefiniert.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

11) .....

12) .....

Welche Strukturmuster findet man in diesem Datenmodell wieder? Geben Sie zu den Muster-  
namen die beteiligten Typ- und Konstruktornamen an.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

Welche Verhaltensmuster sind zur Verarbeitung dieser Strukturen geeignet?

1) .....

2) .....

3) .....

**Aufgabe 3:**

Entwickeln Sie ein Datenmodell in Form einer abstrakten Syntax für ein hierarchisches Dateisystem.

Ein Dateisystem in dieser Aufgabe ist ein Verzeichnis, in dem Namen Einträge zugeordnet sind. Einträge können von unterschiedlicher Art sein. Eine Ausprägung sind einfache Dateien. Eine zweite Form ist ein (Unter-)Verzeichnis, eine dritte Form ein symbolischer Verweis (symbolic link), der aus einem absoluten Pfad besteht. Ein Pfad bestimmt einen Eintrag in dem hierarchischen Dateisystem.

Alle Einträge besitzen zusätzlich eine Menge von Attributen, wie zum Beispiel Zugriffsrechte und Zeitstempel. Diese Menge von Attributen soll nicht fest vorgegeben sein, sondern es soll eine variablen Anzahl von Attributen möglich sein.

Das Datenmodell in Form einer abstrakten Syntax in Haskell:

- 1) .....
- 2) .....
- 3) .....
- 4) .....
- 5) .....
- 6) .....
- 7) .....
- 8) .....
- 9) .....
- 10) .....
- 11) .....
- 12) .....

**Aufgabe 4:**

Interpretierer und Besucher sind zwei Verhaltensmuster für ähnliche Anwendungsgebiete.

Welches sind die Nachteile des Interpretierermusters gegenüber dem Besuchermuster?

Stichworte:

1) .....

2) .....

3) .....

4) .....

5) .....

Welches sind die Nachteile des Besuchermusters gegenüber dem Interpretierermuster?

Stichworte:

1) .....

2) .....

3) .....

4) .....

5) .....

**Aufgabe 5:**

Geben Sie das OMT- oder UML-Diagramm für das Brücken-Muster an.

Klassifizieren Sie das Muster.

.....

In welchen Situationen ist es anwendbar?

1) .....

2) .....

3) .....

4) .....