Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

29. Januar 2010

---

Aufgaben zur Klausur **Softwaredesign** im WS 2009/10 (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 16 Seiten.

---

**Aufgabe 1:**

Entwerfen Sie ein Datenmodell zur Beschreibung von einfachen Web–Fragebögen.

Ein Fragebogen besteht aus einer Liste von Fragen. Zu einer Frage gehören der Fragetext und eine Antwort. Die Antworten können unterschiedlich strukturiert sein. Es sollen hier folgende Antwortarten erlaubt sein:

1. Einzeilige Textfelder in einer anzugebenden Länge. Beispiel:
   *Wie heißt Knöhler mit Vornamen?* ☐
2. Ja-Nein-Fragen, bei der die beiden Auswahlmöglichkeiten durch Texte beschrieben werden.
   *Kennen Sie Murkel?* *ja* ☐ *nein* ☐
3. 1-aus-n Fragen, bei denen jede Auswahlmöglichkeit durch einen Text beschrieben wird. Zwei Beispiele:
   *Kennen Sie Oosterwelle?* *ja* ☐ *nein* ☐ *oberflächlich* ☐
   *Welche Note würden Sie Gantenberg geben?*
   *1* ☐ *2* ☐ *...* *6* ☐
   Eine Notenskala ist hiermit also auch realisierbar.
   Diese Antwortarten kann man in HTML zum Beispiel mit Radio Buttons darstellen.
4. m-aus-n Fragen, diese haben die gleichen Bestandteile wie die 1-aus-n Fragen, allerdings haben die Antworten eine andere Gestalt.
5. Aus Teilfragen zusammengesetzte Fragen. Schachtelungen in solchen Teilfragen sollen in der Datenstruktur möglich sein. Beispiel:
   *Was wissen Sie über Schräuble?*
   *a) Vorname?* ☐
   *b) Alter?* *unter 20* ☐ *21-69* ☐ *älter* ☐

Tipp: Beachten Sie bitte, dass die Struktur eines Fragebogens beschrieben werden soll, nicht die Werte eines ausgefüllten Exemplars.

Eine Abstrakte Syntax für Fragebögen in Haskell Notation. Benutzen Sie bitte pro Datentyp nur einen Typkonstruktor. Versuchen Sie, das Datenmodell möglichst einfach zu halten.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

10) .....

Ein gleichwertiges OMT–Diagramm

**Aufgabe 2:**

Entwickeln Sie zu einer kontextfreien Grammatik für arithmetisch–logische Ausdrücke eine abstrakte Syntax für die interne Repräsentation und Verarbeitung dieser Ausdrücke.

Die konkrete Syntax sei durch folgende in BNF–Notation gegebene kontextfreie Grammatik beschrieben. Dabei sind Terminalsymbole in ' gesetzt. Des weiteren sind $Ident$ und $IntConst$ Terminalsymbole, die aus einem Namen bzw. einer ganzen Zahl bestehen.

| | | |
|---|---|---|
| .0 | $ALExpr$ | $::= ALExpr\ 'OR'\ LExpr_1 \mid LExpr_1$ |
| .1 | $LExpr_1$ | $::= LExpr_1\ 'AND'\ LExpr_2 \mid LExpr_2$ |
| .2 | $LExpr_2$ | $::= AExpr_3\ '=='\ AExpr_3 \mid AExpr_3\ '!='\ AExpr_3 \mid AExpr_3$ |
| .3 | $AExpr_3$ | $::= AExpr_4\ '+'\ AExpr_3 \mid AExpr_4\ '-'\ AExpr_3 \mid AExpr_4$ |
| .4 | $AExpr_4$ | $::= Expr_5\ '*'\ AExpr_4 \mid Expr_5\ '/' AExpr_4 \mid Expr_5$ |
| .5 | $Expr_5$ | $::= '('\ ALExpr\ ')' \mid Expr_6$ |
| .6 | $Expr_6$ | $::= Ident \mid IntConst \mid 'true' \mid 'false'$ |

In der Sprache gibt es also die logischen Operatoren UND und ODER, Gleichheits– und Ungleichheitstests, und die 4 Grundrechenarten. Die Prioritäten und Assoziativitäten sind durch die verschiedenen Grammatikregeln festgelegt. Als elementare Ausdrücke sind Bezeichner, ganzzahlige Konstante und die beiden Wahrheitswerte erlaubt. Ausdrücke können beliebig komplex werden (Regel .5).

Entwickeln Sie für diese Sprache eine abstrakte Syntax in Haskell Notation. Versuchen Sie durch Abstraktion ein möglichst einfaches Modell mit wenigen Datentypen (maximal 5) zu entwickeln. Verwenden Sie keine geschachtelten Typdefinitionen.

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 seien die folgenden Java–Schnittstellen und Klassen

---

```
interface List {

    public void prepend(Value v);

    public void append(Value v);

    public int length();

    public Value get(int i);

    public boolean isEmpty();
}
```

---

```
abstract public class MakeList {
    abstract public
        List newEmptyList();
}
```

---

```
class ListAsValueList implements List {
    private Value l;

    public ListAsValueList() {
        l = Value.nil();
    }

    public void prepend(Value v) {
        l = Value.pair(v,l);
    }

    public void append(Value v) {
        l = l.append(v);
    }

    public int length() {
        return
            l.length();
    }
```

```
    public Value get(int i) {
        Value l1 = l;
        while (i != 0) {
            l1 = l1.cdr();
            --i;
        }
        return
            l1.car();
    }

    public boolean isEmpty() {
        return
            l.isNil();
    }

    public String toString() {
        String res = "";
        int len = length();

        if (len != 0) {

            res = get(0).toString();

            for (int i = 1;
                 i < length();
                 ++i) {
                res += ",  " + get(i).toString();
            }
        }
        return
            res;
    }
}
```

---

```
public class MakeValueList extends MakeList {

    public List newEmptyList() {
        return
            new ListAsValueList();
    }
}
```

---

```
abstract class Value {

    public boolean isAtom() {
        return
            false;
    }
    public boolean isNil() {
        return
            false;
    }
    public boolean isPair() {
        return
            false;
    }
    public boolean isList() {
        return
            false;
    }
    public boolean isEqual(Value v2) {
        return
            false;
    }
    public Value car() {
        throw
            new RuntimeException("car not supported");
    }
    public Value cdr() {
        throw
            new RuntimeException("cdr not supported");
    }
    public Value append(Value v2) {
        return
            new Pair(v2,this);
    }
    public int length() {
        return
            0;
    }

    public static Value nil() {
        return
            Nil.nil;
    }
    public static Value pair(Value car, Value cdr) {
        return
            new Pair(car, cdr);
    }
```

```
    private static java.util.Dictionary atoms
        = new java.util.Hashtable();

    public static Value atom(String name) {
        Value a = (Atom)(atoms.get(name));

        if (a == null) {
            a = new Atom(name);
            atoms.put(name,a);
        }
        return
            a;
    }
}
```

---

```
final class Nil extends Value {
    static final Value nil = new Nil();

    private Nil() {}

    public boolean isAtom() {
        return
            true;
    }
    public boolean isNil() {
        return
            true;
    }
    public boolean isList() {
        return
            true;
    }
    public boolean isEqual(Value v2) {
        return
            v2 instanceof Nil;
    }
    public String toString() {
        return
            "nil";
    }
}
```

---

```
final class Pair extends Value {

    final Value car, cdr;

    Pair(Value car, Value cdr) {
        this.car = car;
        this.cdr = cdr;
    }

    public boolean isPair() {
        return
            true;
    }
    public boolean isList() {
        return
            cdr.isList();
    }

    public Value car() {
        return
            car;
    }

    public Value cdr() {
        return
            cdr;
    }
    public boolean isEqual(Value v2) {
        return
            (this == v2)
            ||
            (v2 instanceof Pair
              && car.isEqual(v2.car())
              && cdr.isEqual(v2.cdr()));
    }

    public Value append(Value v2) {
        return
            new Pair(car,
                     cdr.append(v2));
    }

    public int length() {
        return
            1 + cdr.length();
    }
```

```
    public String toString() {
        return
            "(  " + car.toString() + "  .  " + cdr.toString() + "  )";
    }

}
```

---

```
final class Atom extends Value {

    final String name;

    Atom(String name) {
        this.name = name;
    }

    public boolean isAtom() {
        return
            true;
    }

    public boolean isEqual(Value v2) {
        return
            (this == v2)
            ||
            (v2 instanceof Atom
              && name.equals(((Atom)v2).name));
    }

    public String toString() {
        return
            name;
    }
}
```

---

Welche **Erzeugungsmuster** sind in dieser Ansammlung von Klassen zu erkennen? Nennen Sie den jeweiligen Musternamen, die beteiligten Klassen und die beteiligten Methoden und/oder Referenzen.

1. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

2. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

3. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

4. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

5. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

Welche **Verhaltensmuster** sind in dieser Ansammlung von Klassen zu erkennen? Nennen Sie den jeweiligen Musternamen, die beteiligten Klassen und die beteiligten Methoden und/oder Referenzen.

1. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

2. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

3. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

4. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

5. Musternamе, beteiligte Klassen, Methoden und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

Welche **Strukturmuster** sind in dieser Ansammlung von Klassen zu erkennen? Nennen Sie den jeweiligen Musternamen, die beteiligten Klassen und die beteiligten Referenzen.

1. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

2. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

3. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

4. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .

5. Mustername, beteiligte Klassen und Referenzen

   . . . . . .

   . . . . . .

   . . . . . .