Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

8. Februar 2006

Aufgaben zur Klausur **Softwaredesign** im WS 05/06 (WI h252, WI h253, II h752, MI h403, MI h404, MI h405)

Zeit: 60 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 12 Seiten

---

Aufgabe 1:

Gegeben seien die folgenden Java Schnittstellen und Klassen:

```
public interface SimpleImage { }

public abstract class Image {
        public static final double black = 0.0;
        public static final double white = 1.0;
        public static final double grey = 0.5;

        public final int w;
        public final int h;

        protected Image(int w, int h){
                this.w = w; this.h = h;
        }
        protected abstract double at(int i, int j);

        public double pixelAt(int i,int j){
                return
                Math.max(Math.min(at(i,j),1.0),0.0);
        }
}

public class Uni extends Image implements SimpleImage {
        private final double color;

        public Uni(int w, int h, double color){
                super(w,h);
                this.color = color;
        }
        protected double at(int i, int j) {
                return color;
        }
}

public class GreyMap extends Image implements SimpleImage {

        private final double [][] pixels;

        public GreyMap(double [][] pixels){
                super(pixels.length,pixels[0].length);
                this.pixels = pixels;
        }
        protected double at(int i, int j) {
                return pixels[i][j];
```

```
        }
}

public class ImageCache extends Image implements SimpleImage {
        protected final Image img;

        public ImageCache(Image img){
                super(img.w,img.h);
                this.img = new GreyMap(eval(img));
        }

        private
        double [][] eval(Image img) {
                double [][] a = new double[img.w][img.h];
                for (int i=0; i<img.w;++i)
                        for(int j=0; j<img.h;++j)
                                a[i][j] = img.pixelAt(i,j);
                return a;
        }

        protected double at(int i, int j) {
                return img.at(i,j);
        }
}

public class Superimpose extends Image {
        protected final Image top,bottom;

        public Superimpose(Image top, Image bottom){
                super(Math.max(top.w,bottom.w),Math.max(top.h,bottom.h));
                this.top = top;
                this.bottom = bottom;
        }

        protected double at(int i, int j) {
                if (i<top.w && j<top.h)
                        return top.at(i,j);
                if (i<bottom.w && j<bottom.h)
                        return bottom.at(i,j);
                return black;
        }
}

public class SideBySide extends Image {
        protected final Image left, right;

        public SideBySide(Image left, Image right){
```

```
                super(left.w + right.w,Math.max(left.h,right.h));
                this.left = left;
                this.right = right;
        }

        protected double at(int i, int j) {
                if (i < left.w && j < left.h)
                        return left.at(i,j);
                if (i − left.w < right.w && j < right.h)
                        return right.at(i−left.w,j);
                return black;
        }
}

abstract class HalfImage extends Image {
        protected final Image img;

        protected HalfImage(int w, int h, Image img) {
                super(w,h);
                this.img = img;
        }
}

public class HalfWidth extends HalfImage {

        public HalfWidth(Image img){
                super((img.w +1)/2,img.h,img);
        }
        protected double at(int i, int j) {
                return
                (img.at(2∗i,j) + img.at(2∗i+1,j))/2.0;
        }
}

public class HalfHeight extends HalfImage {

        public HalfHeight(Image img){
                super(img.w,(img.h + 1)/2,img);
        }
        protected double at(int i, int j) {
                return
                (img.at(i,2∗j) + img.at(i,Math.min(2∗j+1,img.h)))/2.0;
        }
}

public class HalfSize extends HalfImage {
```

```
        public HalfSize(Image img){
                super((img.w + 1)/2,(img.h + 1)/2,
                                new HalfWidth(new HalfHeight(img)));
        }
        protected double at(int i, int j) {
                return
                img.at(i,j);
        }
}

public class ImageGenerator {
        public Image uni(int w, int h, double c) {
                return new Uni(w,h,c);
        }
        public Image imageMap(Image img) {
                return new ImageCache(img);
        }
        public Image halfSizeImage(Image img) {
                return new HalfSize(img);
        }
}

public class SmartImageGenerator extends ImageGenerator {
        public Image imageMap(Image img) {
                if (img instanceof SimpleImage)
                        return img;
                return super.imageMap(img);
        }
        public Image halfSizeImage(Image img) {
                return new HalfWidth(new HalfHeight(img));
        }
}
```

Welche Strukturmuster kommen in diesem Modell vor?
Geben Sie jeweils den Musternamen und die beteiligten Klasse, Schnittstellen und Referenzen an.

1) .....

2) .....

3) .....

4) .....

5) .....

Welche Erzeugungsmuster kommen in diesem Modell vor?
Geben Sie jeweils den Musternamen und die beteiligten Klasse, Schnittstellen, Referenzen und Methoden an.

1) .....

2) .....

3) .....

Welche Verhaltensmuster kommen in diesem Modell vor?
Geben Sie jeweils den Musternamen und die beteiligten Klasse, Schnittstellen, Referenzen und Methoden an.

1) .....

2) .....

3) .....

4) .....

5) .....

Aufgabe 2:

Entwickeln Sie ein Datenmodell zu Katalogisierung von Papierbildern und Dias. In diesem Katalog werden nicht die Bilder direkt gespeichert, sonder nur Information über die Bilder und wie und wo diese gelagert werden.

Ein Katalog kann einmal eine Menge von Diakästen beschreiben. Diese sind innerhalb eines Katalogs eindeutig durch einen Namen identifizierbar. Diakästen können um eine Menge von beschreibenden Attributen wie Titel, Datum, Ort, Reise, Kapazität des Kastens, ..., ergänzt werden.

Jedes Dia ist innerhalb eines Kastens durch eine Nummer identifiziert. Zu jedem Dia können wieder beliebige beschreibende Attribute existieren, wie Titel, Ort, oder auch technische Daten.

Ein Katalog kann weiter Photoalben enthalten. Diese werden wieder durch einen Namen eindeutig identifiziert. Wie bei Diakästen sind bei Alben auch beschreibende Attribute möglich.

Bilder in einem Album werden eindeutig durch ihre Seitenzahl und eine Position innerhalb einer Seite identifiziert. Sonst besteht zwischen Bildern und Dias kein Unterschied.

Kleine Kataloge, z.B. die Alben in einem Regal oder die in einem Raum gespeicherten Diakästen, sollen zu einem Gesamtkatakog zusammengefasst werden können. Innerhalb eines Kataloges werden die Teilkataloge durch einen Namen identifiziert. Einem Katalog selbst werden ebenfalls beschreibende Attribute zugeordnent. Die Zusammenfassung von Alben, Diakästen und Katalogen zu Gesamtkatalogen kann wiederholt werden.

Die Menge der beschreibenden Attribute für Kataloge, Alben, Kästen, Bilder und Dias soll eine einheitliche Struktur besitzen.

Es wird weiter angenommen, dass folgende einfache Wertebereiche geeignet vordefiniert sind: *AttrName, AttrWert, Nummer, Seite, Position, Id*. Verwenden Sie diese Bereiche als einfache Datentypen. Für die anderen Bereiche verwenden sie, wenn die Lösung es erfordert, unter anderem Namen wie *Katalog, Album, Kasten, Dia, Bild, Attribute*.

Entwickeln Sie ein Datenmodell in abstrakter Syntax in der Notation von Haskell, so wie diese in der Vorlesung genutzt worden ist. Verwenden Sie pro Datentypdefinition nur einen Typkonstruktor (also bitte keine geschachtelten Datentypdefinitionen).

Welche Strukturmuster kommen in diesem Modell vor? Nennen Sie den jeweiligen Musternamen und die beteiligten Datentypen.

1. Mustername und beteiligte Klassen

   .....

   .....

   .....

2. Mustername und beteiligte Klassen

   .....

   .....

   .....

3. Mustername und beteiligte Klassen

   .....

   .....

   .....

Aufgabe 3:

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	$Expr$	$::= Expr\ 'OR'\ Expr_1 \mid Expr_1$
.1	$Expr_1$	$::= Expr_1\ 'AND'\ Expr_2 \mid Expr_2$
.2	$Expr_2$	$::= Expr_3\ '=='\ Expr_2 \mid Expr_3\ '!='\ Expr_2 \mid Expr_3$
.3	$Expr_3$	$::= Expr_4\ '+'\ Expr_3 \mid Expr_4\ '-'\ Expr_3 \mid Expr_4$
.4	$Expr_4$	$::= Expr_5\ '*'\ Expr_4 \mid Expr_5\ '/' Expr_4 \mid Expr_4$
.5	$Expr_5$	$::= '('\ Expr\ ')' \mid Expr_6$
.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) .....
