Fachhochschule Wedel
Prof. Dr. Uwe Schmidt

10. August 2000

Aufgaben zur Klausur **Softwaredesign** und **Software Entwicklungs–Methoden** im SS 2000 (WI h252, WI 56, II h752, MI h403)

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 14 Seiten

---

Aufgabe 1:

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();
}
```

```
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;
    }
}
```

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

```
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 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;
    }
}
```

```
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() + " )";
    }

}
```

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

   .....

   .....

   .....

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. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

2. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

3. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

4. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

5. Mustername, 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. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

2. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

3. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

4. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

5. Mustername, beteiligte Klassen, Methoden und Referenzen

   .....

   .....

   .....

Aufgabe 2:

Welches sind die Vorteile des Besuchermusters gegenüber dem Interpretierermuster?
Stichwörter:

1) .....

2) .....

3) .....

4) .....

5) .....

Welches sind die Nachteile des Besuchermusters gegenüber dem Interpretierermuster?
Stichwörter:

1) .....

2) .....

3) .....

4) .....

5) .....

Aufgabe 3:

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. Ein einzeiliges Textfeld in einer anzugebenden Länge.
2. eine 1 aus n Frage, bei der jede Auswahlmöglichkeit durch einen Text beschrieben ist. Einfachstes Beispiel ist eine Ja/Nein–Antwort, Beispiel:
   Mögen Sie Kohl? ja ☐ nein ☐
   Eine Notenskala von 1 bis 6 ist hiermit aber auch realisierbar. Diese Antwortarten kann man in HTML zum Beispiel mit radio buttons darstellen.
3. eine m aus n Frage, diese hat die gleichen Bestandteile wie die 1 aus n Antwort.
4. Eine aus Teilfragen zusammengesetzte Antwort. Hierbei besteht die Antwort wieder aus mehreren Teilfragen. Schachtelung dieser Teilfragen soll in der Datenstruktur möglich sein. Beispiel:
   Wie finden Sie ...
   ... Otto? prima ☐ nicht prima ☐
   ... Heino? toll ☐ gar nicht ☐

Eine Abstrakte Syntax für Fragebögen

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....

Ein gleichwertiges OMT–Diagramm

Aufgabe 4:

Im funktionalen Programmierstil, auch in Pascal und C, sind als Parameter auch Funktionen möglich. Diese werden bei ereignisgesteuerten Systemen häufig als call–back–Routinen verwendet. In objektorientierten Sprachen, z.B. Eiffel und Java, fehlt das Konzept der Funktionen als Parameter.

Mit welchem Verhaltensmuster werden Funktionen als Parameter simuliert?

.....

Geben Sie das OMT–Diagramm für die Struktur dieses Musters an.