
Aufgaben zur Klausur **Software design** im SS 2003 (WI h252, WI h253, II h752, MI h403, MI h404, MI h405)

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.

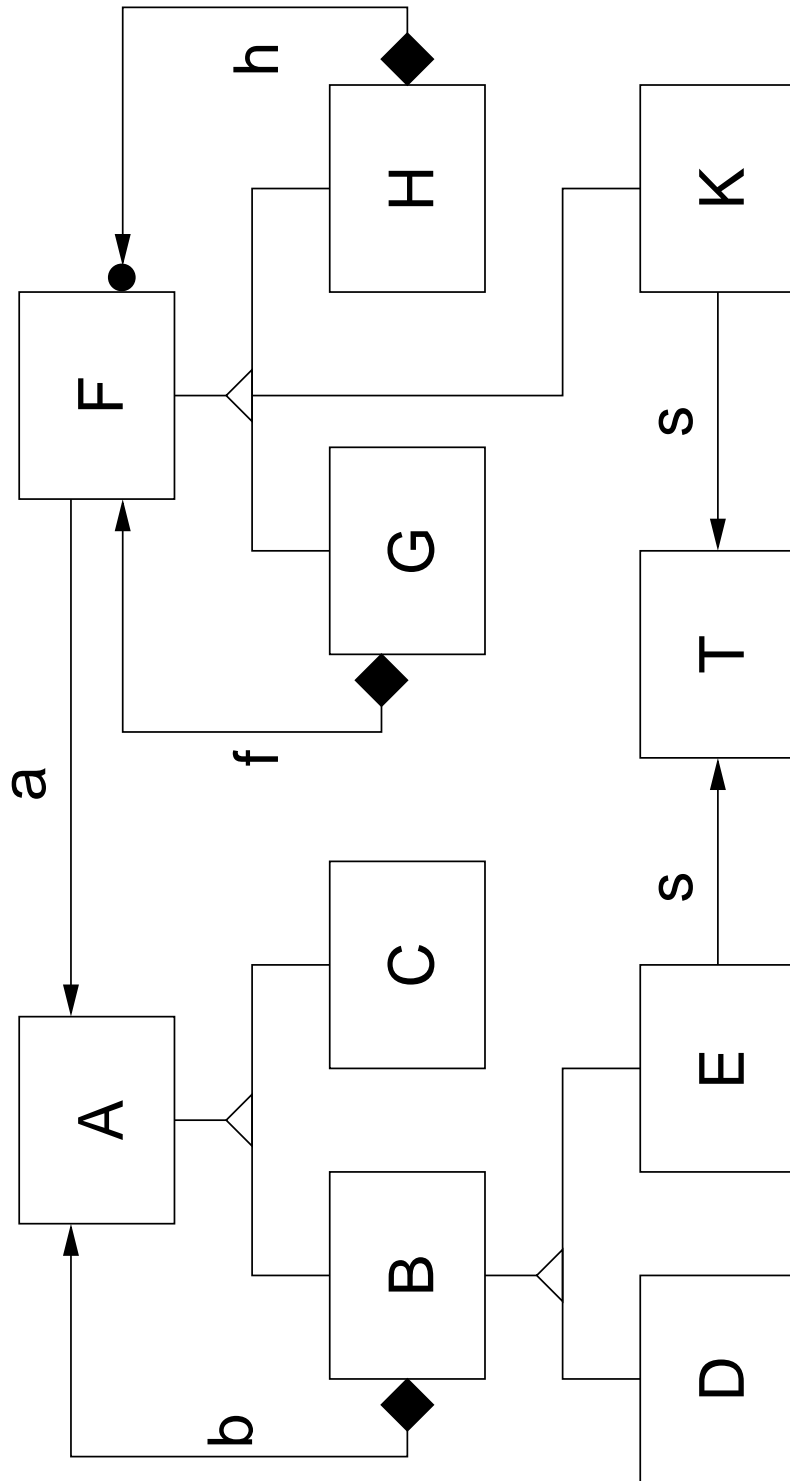
Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 16 Seiten

Aufgabe 1:

Gegeben sei das folgende OMT-Diagramm:



Welche Strukturmuster sind in diesem Diagramm enthalten? Geben sie die Muster und die an den einzelnen Mustern beteiligten Klassen und Referenzen an.

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

Aufgabe 2:

Geben Sie zu der folgenden abstrakten Syntax ein gleichwertiges OMT-Diagramm an.
String sei dabei ein elementarer Datentyp.

.0 $A = S \times B$

.1 $B = C^*$

.2 $C = A \mid B \mid D \mid S$

.3 $D = \text{map } S \text{ to } C$

.4 $S = \textit{String}$

Aufgabe 3:

Entwickeln Sie ein Datenmodell für die Verarbeitung von XML und HTML. In diesem etwas vereinfachten Modell ohne Kommentare und Processing Instructions soll ein XML-Dokument aus einem (Wurzel-)Tag bestehen. Ein Tag setzt sich zusammen aus einem Tagnamen, einer Menge von Attributen und einem Inhalt. Der Inhalt ist eine Folge von einfachem Text und Tags.

Verwenden Sie in dem Datenmodell pro Typdefinition nur einen Typkonstruktor. *String* sei ein vordefinierter Datentyp für Zeichenreihen.

Eine Abstrakte Syntax für ein *XMLDokument*:

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

Aufgabe 4:

Klassifizieren Sie das Iterator-Muster.

.....
.....

Welches ist der Zweck des Iterator-Musters?

.....
.....

Wo ist das Muster typischerweise anwendbar?

.....
.....
.....

Warum ist das Muster nicht geeignet für die Verarbeitung von Komposita?

.....
.....
.....

Geben Sie das Klassendiagramm für das Iterator-Muster an:

Aufgabe 5:

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 ll = l;  
        while (i != 0) {  
            ll = ll.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

.....
.....
.....