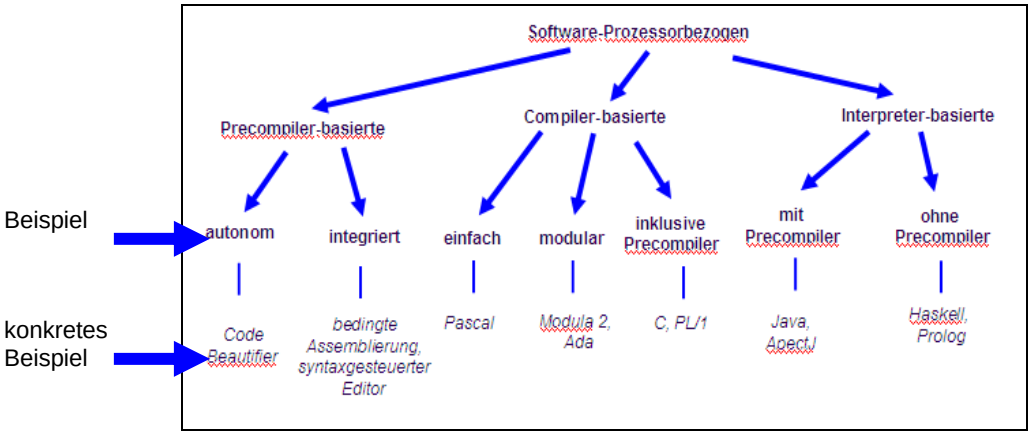
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Prof. Dr. R. Dumke, 23. Juli 2010

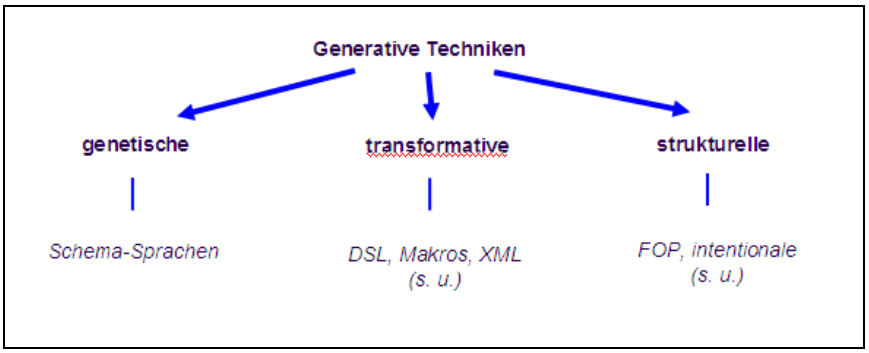
Prüfungsklausur Programmierparadigmen

Bewertung

Aufgabe 1 (8 Punkte): *Nennen Sie jeweils ein Beispiel für die Klassifikationsformen der Programmierungstechniken „precompiler-basierte“, „compiler-basierte“, „interpreter-basierte“ und „generative“ und geben Sie jeweils ein konkretes Beispiel an.*

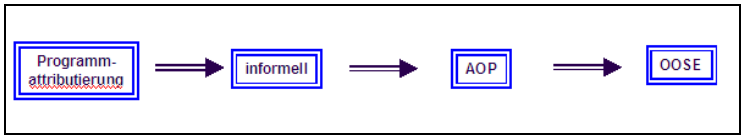
LÖSUNG aus:



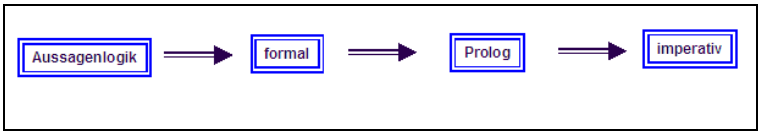


Aufgabe 2 (4 Punkte): *Beschreiben Sie zwei ausgewählte Programmierparadigmen hinsichtlich ihrer vier möglichen/sinnvollen Merkmalsinhalte.*

LÖSUNG:



2 P

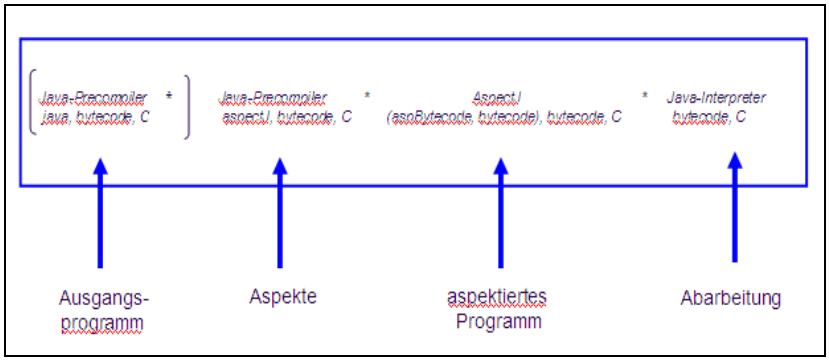


2 P

u.v.a.m.

Aufgabe 3 (4 Punkte): *Beschreiben Sie die Funktionsweise des AspectJ-Aspektwebers in einer Software-Prozessorbeschreibungsform.*

LÖSUNG:



1 P **1 P** **1 P** **1 P**

Aufgabe 4 (6 Punkte): *Gegeben sei das folgende AspectJ-Programmstück:*

```
public aspect MeinAspekt {
        pointcut InitAspekt():
             initialization(public Klasse01.new());
        before(): InitAspekt() {
             System.out.println("Objektinitialisierung!"); } }
```

Welche der genannten AspectJ-Elemente beinhaltet dieses Programmstück?

a) Joinpoint b) Crosscutting concern c) Advice

d) symmetrischer Aspekt e) Pointcut f) Core concern

LÖSUNG:

Zutreffend: a), b), c) und e) **4 P**

Nicht zutreffend: d) und f) **2 P**

Aufgabe 5 (4 Punkte): *Welche vier Fehler sind im folgenden AspectJ-Advice vorhanden?*

```
void arounded(initialerWert): BerAspekt(int initialerWert)
      {if (initialerWert =< 10) System.out.println(“Wertefehler!");}
```

LÖSUNG:

```
void around(int initialerWert): BerAspekt(initialerWert)
```

3 P

```
  {if (initialerWert <= 10) System.out.println(“Wertefehler!");}
```

1 P

Aufgabe 6 (4 Punkte): *Was bewirkt der folgende AspectJ-Aspekt?*

```
pointcut MethodenAufrufe():
      call(* *.*(..)) && !within(ZahlAspekt) && !cflow(adviceexecution());
before(): MethodenAufrufe()
```

LÖSUNG:

1. gibt alle Methodenaufrufe aus, **1 P**
2. schränkt die Signaturauswahl nicht ein, **1 P**
3. vermeidet dabei Wiederholungen im Aspekt selbst, **1 P**
4. schließt den Advice-Verarbeitungsteil aus **1 P**

Aufgabe 7 (8 Punkte): *Nennen Sie drei Anfrageformen in Prolog? Welche Programmanalyse gewährleisten die Prolog-Techniken trace, gtrace, pmt-tool, profile und gxref?*

LÖSUNG:

Damit können wir drei Kategorien von Anfragen unterscheiden:

Eine ***Suchanfrage*** *(finding query)*, die für Werte von Variablen zum gegenwärtig gespeicherten Programm prüft, ob ein Ziel war ist.

Eine ***Bestätigungsanfrage*** *(confirming query)* zu einem variablenlosen Ziel, wie zum Beispiel

```
?- interessiert(lisa,prolog).

yes ?- interessiert(lisa,annet).
no
```

Eine ***Aktionsanforderung (action query)***, wie beispielsweise die Dateieingabe oder auch das Verlassen des Prolog-Systems durch

```
?- halt.
```

3 P

trace: Ablaufverfolgung **1 P**
gtrace: Callghraphbasierte Ablaufverfolgung **1 P**
pmt-tool: Kennzahlbasierte Prolog-Quellcodeanalyse **1 P**
profile: Prolog-Leistungsanalyse **1 P**
gxref: Analyse möglicher Refenrenzen zwischen gleichzeitig geladenen Prolog-Prg. **1P**

Aufgabe 8 (12 Punkte): *Ordnen Sie die folgenden Prolog-Prädikate ihren jeweiligen Bedeutungen zu:* `dynamic, cut, abolish, get, consult, read, assert, see, fail, listing, retract, tell.`

a) Eingabefile öffnen	b) false-Abbruch	c) Ausgabefile öffnen	d) Klausel-Abbruch
e) Klausel anhängen	f) Zeichen einlesen	g) Code anzeigen	h) Code änderbar machen
i) Satz einlesen	j) Programm einlesen	k) Klausel entfernen	l) Entfernen gleichartiger Klauseln

LÖSUNG:

a) `see`	b) `fail`	c) `tell`	d) `cut`	**4 P**
e) `assert`	f) `get`	g) `listing`	h) `dynamic`	**4 P**
i) `read`	j) `consult`	k) `retract`	l) `abolish`	**4 P**

Aufgabe 9 (4 Punkte): *Welche Techniken werden in Prolog mit den Operatoren "-->", "#", "=.." und "=:=" umgesetzt?*

a) Listenkonvertierer b) Termwertegleichheit c) Nichtdeterminismus d) implizite Grammatik

LÖSUNG:

a) „=.." b) „=:=" c) „#" d) „-->" **4 P**

Aufgabe 10 (6 Punkte): *Schreiben Sie in Prolog zwei Programmvarianten (mit und ohne Rekursion) zur Bestimmung des i-ten Listenelementes.*

LÖSUNG:

a) beispielsweise ohne Rekursion in einer ausführlichen Form:
b)

```
iElement(Liste,Iorg,X) :- iElement1(Liste,Iorg,0,X).
iElement1([Kopf|Rest],Iorg,Zaehler,X) :- Iorg =:= Zaehler, X = Kopf.
iElement1(Liste,Iorg,X) :- Zaehler < Iorg, Zaehler1 is Zaehler + 1,
                           iElement1(Rest,Iorg,Zaehler1,X).
```

3 P

b) beispielsweise mit Rekursion in einer sehr kurzen Form:

```
iElement(Liste,Iorg,X) :-
                    T =.. [feld|Liste], arg(larg,T,X).
```

3 P

u.v.a.m.