Name: [redacted] Matrikelnummer: [redacted]

89

Test 2 in Programmierparadigmen am 26. 1. 2026

Beispiele für **markierte (= ausgewählte)** Auswahlfelder: ⊗ ☑ ☒⊘
Beispiele für **nicht markierte (= nicht ausgewählte)** Auswahlfelder: ○ ☒ ☒○

Beurteilung: 1 Punkt pro Zeile mit gänzlich richtig markierten Auswahlfeldern

Aufgabe 1 (10 Punkte)

Bitte markieren Sie jedes Auswahlfeld, bei dem der links stehende Typausdruck (mit Typnamen aus den Paketen `java.util.function` und `java.lang`) ein Typ des darüber stehenden Lambdas ist. Es können keines, eines oder mehrere Felder pro Zeile auszuwählen sein.

	`x->x+2`	`x->{}`	`x->y->x+y`	`(x,y)->x+y`	`Long::sum`
`Consumer<String>`	○	⊗	○	○	○
`BinaryOperator<String>`	○	○	○	⊗	○
`LongUnaryOperator`	⊗	○	○	○	○
`LongBinaryOperator`	○	○	○	⊗	⊗
`BiFunction<Long,String,Long>`	○	○	○	○	○
`BiFunction<String,Long,String>`	○	○	○	⊗	○
`BiFunction<Long,Long,Long>`	○	○	○	⊗	⊗
`Function<Long,Function<Long,Long>>`	○	○	⊗	○	○
`Function<Long,Function<String,String>>`	○	○	⊗	○	○
`Function<Long,LongUnaryOperator>`	○	○	⊗	○	○

7

Aufgabe 2 (10 Punkte)

`A` und `B` seien definiert durch `interface A<T> {}` und `interface B<R, S> extends A<S> {}`.
Bitte markieren Sie jedes Auswahlfeld, bei dem der links stehende Typ ein Untertyp des darüber stehenden Typs ist. Es können keines, eines oder mehrere Felder pro Zeile auszuwählen sein.

	`A<String>`	`A<? super String>`	`B<? extends String, ?>`
`A<Object>`	○	⊗	[redacted]
`A<String>`	⊗	⊗	[redacted]
`B<Object, String>`	⊗	⊗	○
`B<String, Object>`	○	⊗	⊗
`B<A<String>, Object>`	○	⊗	○
`B<A<Object>, String>`	⊗	⊗	○
`B<?, ? extends String>`	[redacted]	[redacted]	○
`B<? extends String, ?>`	[redacted]	[redacted]	⊗
`B<? extends String, ? super String>`	[redacted]	⊗	[redacted] (X)
`B<? super String, ? extends String>`	[redacted]	○	○

4

Aufgabe 3 (10 Punkte)

Bitte markieren Sie in jeder Zeile das eine Auswahlfeld, bei dem die links stehende Aussage am ehesten eine Eigenschaft der darüber stehenden Parametrisierungsform in Java oder AspectJ ist.

	Annotationen	Aspekte	Generizität
Lücken werden durch spezielle Parameter dargestellt	○	●	⊗
`around()` ersetzt den spezifizierten Porgammtext	○	⊗	○
mit `@Retention` wird die Lebensdauer festgelegt	⊗	○	○
Typschranken legen vorausgesetzte Funktionalität fest	○	○	⊗
Join-Points sind dynamisch identifizierbare Programmstellen	○	⊗	○
kann die Werkzeuge zur Programmerzeugung beeinflussen	⊗	○	○
„`*`“ steht für beliebige Zeichen außer „`.`“ (Punkt)	○	⊗	○
für binäre Methoden wird Rekursion benötigt	○	○	⊗
zur Laufzeit sind Daten über Reflexion zugreifbar	⊗	○	○
wird für Kernfunktionalität meist gemieden	○	⊗	○

10

Aufgabe 4 (10 Punkte)

Bitte markieren Sie jedes Auswahlfeld, bei dem die links stehende Kommandozeile (in `bash` ausgeführt) die darüber stehende Auswirkung hat. Es können keines, eines oder mehrere Felder pro Zeile auszuwählen sein.

	Standardeingabe von `cat` umgeleitet	Standardausgabe von `cat` umgeleitet	Fehlerausgabe von `cat` umgeleitet	Prozesse laufen im Hintergrund	mehrere Prozesse laufen gleichzeitig	Prozesse über *Pipeline verbunden*
`cat < a | wc > b`	⊗	⊗	○	○	⊗	⊗
`cat a || wc b`	○	○	○	○	○	○
`if test 'cat a' = "a b c" ; then wc b ; else wc c ; fi`	○	○	○	○	○	○
`for i in * ; do cat $i \| wc > ../wc/$i ; done`	○	⊗	○	○	⊗	⊗
`for i in * ; do ( cat $i > ../bak/$i & ) ; done`	○	⊗	○	⊗	⊗	○
`cat a |& wc &>> b`	○	⊗	⊗	○	⊗	⊗
`cat < a |& wc &> b &`	⊗	⊗	⊗	⊗	⊗	⊗
`cat a ; wc a`	○	○	○	○	○	○
`cat a &> b && wc b`	○	⊗	⊗	○	○	○
`cat | wc`	○	⊗	○	○	⊗	⊗

10

Aufgabe 5 (10 Punkte)

Bitte markieren Sie jedes Auswahlfeld, bei dem die links stehende Aussage eine Eigenschaft des darüber stehenden Entwurfsmusters ist. Es können keines, eines oder mehrere Felder pro Zeile auszuwählen sein.

	Singleton	Visitor	Factory-Method	Template-Method
verwendet häufig Hooks	☐	☐	☒	☒
Anzahl erzeugter Objekte kontrollierbar	☒	☐	☐	☐
mehrere Arten primitiver Operationen	☐	☐	☐	☒
beruht auf dem Hollywood-Prinzip	☐	☐	☒	☒
verwaltet verwandte Operationen zentral	☐	☒	☐	☐
für Lazy-Initialization geeignet	☒	☐	☐	☐
gibt Grundgerüst von Algorithmus vor	☐	☐	☐	☒
kann zu vielen Objekten führen	☐	☐	☐	☐
kann zu vielen Klassen führen	☐	☐	☒	☐
kann zu vielen Methoden führen	☐	☒	☐	☐

10

Aufgabe 6 (10 Punkte)

Bitte markieren Sie in jeder Zeile das eine Auswahlfeld, bei dem die links stehende Aussage am ehesten eine Eigenschaft des darüber stehenden Entwurfsmusters ist.

	Singleton	Visitor	Factory-Method	Template-Method
kann als Anti-Pattern gesehen werden	☒	☐	☐	☐
Umgang mit Unterklassen problematisch	☒	☐	☒	☐
eine Methode ist häufig `final`	☒	☐	☐	☒
globale Variable ist Alternative dazu	☒	☐	☐	☐
Element ist ein Bestandteil	☐	☒	☐	☐
instance ist meist `static`	☒	☐	☐	☐
accept hat einen Parameter	☐	☒	☐	☐
hilft Duplikate im Code zu vermeiden	☐	☐	☐	☒
Lambdas können Schreibaufwand reduzieren	☐	☐	☒	☐
unterstützt direkte Code-Wiederverwendung	☐	☐	☐	☒

8

Aufgabe 7 (10 Punkte)

Bitte markieren Sie jedes Auswahlfeld, bei dem die links stehende Eigenschaft auf den darüber stehenden Methodenaufruf zutrifft. Es können keines, eines oder mehrere Felder pro Zeile auszuwählen sein.

	`wait()`	`notifyAll()`	`notify()`	`Thread.sleep(5)`
muss mit `InterruptedException` umgehen	☒	☒	☒	☒
darf nur in einem Monitor verwendet werden	☒	☒	☒	☐
soll nicht in einem Monitor verwendet werden	☐	☐	☐	☒
gehört nicht zu Synchronisationsmechanismen	☐	☐	☐	☒
suspendiert Ausführung des aktuellen Threads	☒	☐	☐	☒
suspendiert Ausführung auf bestimmte Zeit	☐	☐	☐	☒
weckt systemweit alle wartenden Threads auf	☐	☒	☐	☐
weckt höchstens einen wartenden Threads auf	☐	☐	☒	☐
weckt mindestens einen wartenden Thread auf	☐	☒	☒	☐
gibt den Lock vorübergehend frei	☒	☐	☐	☐

8

Aufgabe 8 (10 Punkte)

Bitte markieren Sie jedes Auswahlfeld, bei dem die links stehende Aussage eine Eigenschaft des darüber stehenden Entwurfsmusters ist. Es können keines, eines oder mehrere Felder pro Zeile auszuwählen sein.

	Prototype	Proxy	Decorator	Iterator	Factory-Method
ist Entwurfsmuster für Struktur	☐	☒	☒	☐	☐
ist erzeugendes Entwurfsmuster	☒	☐	☐	☐	☒
hilft große Zahl an Klassen zu vermeiden	☒	☐	☐	☐	☐
unterstützt Umkehrung der Abhängigkeiten	☐	☐	☐	☐	☒
häufig als innere Klasse implementiert	☐	☐	☐	☒	☐
führt zu vielen kleinen Objekten	☐	☐	☒	☐	☐
für oberflächliche Erweiterungen geeignet	☐	☐	☒	☐	☐
oft große Anzahl an Unterklassen nötig	☐	☐	☐	☐	☒
Objektidentität ist damit unzuverlässig	☐	☐	☒	☐	☐
beruht auf Delegation	☐	☒	☒	☐	☐

Aufgabe 9 (10 Punkte)

Bitte markieren Sie in jeder Zeile das eine Auswahlfeld, bei dem die links stehende Aussage am ehesten eine Eigenschaft des darüber stehenden Entwurfsmusters ist.

	Prototype	Proxy	Decorator	Iterator	Factory-Method
Smart-Reference ist eine Variante davon	☐	☒	☐	☐	☐
Creator ist ein Bestandteil	☐	☐	☐	☐	☒
Component ist ein Bestandteil	☐	☐	☒	☐	☐
Aggregate ist ein Bestandteil	☐	☐	☐	☒	☐
Subject ist ein Bestandteil	☐	☒	☐	☐	☐
Product ist ein Bestandteil	☐	☐	☐	☐	☒
wird auch *Surrogate* genannt	☐	☒	☐	☐	☐
wird auch *Virtual-Constructor* genannt	☐	☐	☐	☐	☒
wird auch *Wrapper* genannt	☐	☐	☒	☐	☐
wird auch *Cursor* genannt	☐	☐	☐	☒	☐

Aufgabe 10 (10 Punkte)

Bitte markieren Sie in jeder Zeile das eine Auswahlfeld, bei dem die links stehende Aussage am ehesten eine Eigenschaft des darüber stehenden Entwurfsmusters ist.

	Prototype	Proxy	Decorator	Iterator	Factory-Method
kein Decorator, aber gleiche Struktur möglich	☐	☒	☐	☐	☐
zyklische Strukturen bereiten Probleme	☒	☐	☐	☐	☐
flache von tiefen Kopien unterschieden	☒	☐	☐	☐	☐
schlecht geeignet für umfangreiche Objekte	☐	☐	☒	☐	☐
mehrere gleichzeitige Abarbeitungen möglich	☐	☐	☐	☒	☐
Verantwortlichkeiten wieder entziehbar	☐	☐	☒	☐	☐
führt zu parallelen Klassenhierarchien	☐	☐	☐	☐	☒
es gibt externe und interne Varianten	☐	☐	☐	☒	☐
robuste Varianten werden bevorzugt	☐	☐	☐	☒	☐
kann mit kovarianten Problemen umgehen	☐	☐	☐	☐	☒

30