Prüfung
Software Engineering (SE)

Prof. Dr.-Ing. Christian Heller <christian.heller@ba-leipzig.de>

Student	
Vor- und Nachname	Ihre Daten werden von der Klausuraufsichtsperson schriftlich auf der Anwesenheitsliste festgehalten.
Matrikelnummer	
Studienrichtung und Jahr	CS 2017-1
Anmeldename	Das Login "klaus..." muss unbedingt schriftlich auf der Anwesenheitsliste festgehalten werden, da sonst keine Zuordnung des Logins zu Ihrem Namen und damit keine Korrektur der Klausur möglich ist!

Prüfung	
Datum	Juni 2019
Dauer [min]	100
Hilfsmittel	Dokumentation im lokalen Netzwerk (Intranet) sowie Recherche im Internet. NICHT gestattet: * Kommunikation in jeglicher Form * Anmeldung via SSH auf dem Rechner "fileserv" * Anmeldung mit Klausur-Login nach Ende der Prüfung Dies kann leicht geprüft werden (last cs16*, Server-Log-Dateien). Bitte unterlassen Sie also Täuschungsversuche in Ihrem eigenen Interesse.
Bemerkungen	Hinterlegen Sie alle Programme und Antworten in elektronischer Form! Es wird kein Papier angenommen. Möchten Sie Lösungen erläutern, so nutzen Sie Quelltext-Kommentare oder legen eine Text-Datei an. Speichern Sie sämtliche Daten im HOME-Verzeichnis des Nutzers, d. h. unter Windows auf Laufwerk H:\ (NICHT auf C:\ oder "Eigene Dateien")! Idealerweise legen Sie dort ein Unterverzeichnis namens "klausur" an. Lesen Sie die Aufgaben komplett durch, bevor Sie sie lösen! Die Reihenfolge der Lösung ist Ihnen überlassen. Probieren Sie immer, eine Aufgabe zu lösen, da auch auf richtige Teile nicht vollständiger Lösungen Punkte vergeben werden! Falls vom Prinzip her richtig, so werden auch alternative Lösungen akzeptiert. Sie dürfen beliebig viele Bildschirmausgaben von Werten in den Quelltext einbauen, um ein Programm besser nachvollziehen zu können. Bitte duplizieren Sie Ihre Quelltextdateien (workspace) NICHT, da beim Korrigieren dann beide durchsucht werden müssen, was sinnlosen Aufwand verursacht. Diese Aufgabenstellung in Papierform können Sie nach dem Ende der Klausur behalten.

Bewertung						
Aufgabe	1	2	3	4	5	Summe
Punkte	30	20	20	20	10	100

Im Rahmen der Klausur sind Aufgaben mit Bezug zur Unified Modeling Language (UML) zu lösen. Insofern modelliert werden soll, ist das BOUML-Werkzeug zu verwenden.

Aufgabe 1: Struktur [30]

Zweck: Modellieren eines Sinfonieorchesters mittels Klassendiagramm.

a) Es gibt verschiedene Arten von Orchestern: SinfonieOrchester, KammerOrchester, Gamelan, BigBand. [2]

b) Ein Orchester kann einen Dirigenten haben. [2]

c) Er besitzt einen „taktStock“ als privates Attribut. [2]

d) Der Dirigent führt die Tätigkeit „leiten“ aus, welcher ein „werk“ übergeben wird. [2]

e) Neben dem Orchester gibt es einen Chor. Beide sind unabhängig voneinander. [2]

f) Zum SinfonieOrchester gehört mindestens eine Gruppe. [2]

g) Vier Kategorien von Gruppen werden unterschieden: Streicher, HolzBläser, BlechBläser, SchlagInstrument. [2]

Hinweis: In den folgenden Aufgaben brauchen Rollenbezeichnungen und Kardinalitäten NICHT angegeben zu werden (wenn in der Aufgabe nicht anders vermerkt).

h) Die Gruppe der Streicher hat Instrumente vom Typ: Violine und Bratsche. [2]

i) Außerdem gehören zu ihr ViolonCello und KontraBass. [2]

j) Zum Typ Violine gibt es zwei Assoziationen, da man zwischen ersten und zweiten Violinen unterscheidet. Bezeichnen Sie die Assoziationen per Rolle, zur besseren Unterscheidbarkeit! [2]

k) Zu den HolzBläsern gehören Instrumente vom Typ: Flöte und Oboe. [2]

l) Außerdem gehören zu ihnen Klarinette und Fagott. [2]

m) Bei den BlechBläsern sind es Instrumente des Typs: Horn und Trompete. [2]

n) Außerdem gehören zu ihnen Posaune und Tuba. [2]

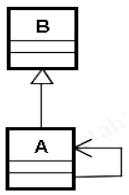
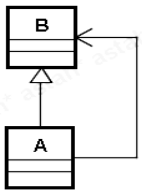
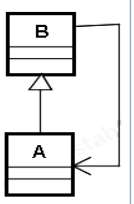
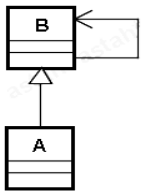
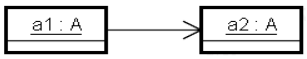
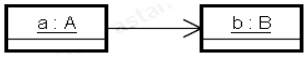
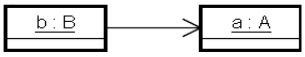
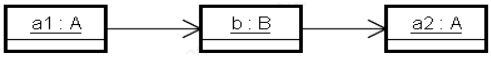
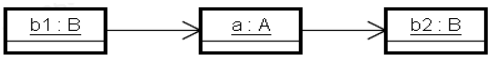
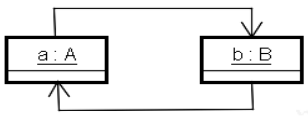
o) Zu SchlagInstrumenten zählt man die Typen: Pauke und SchlagWerk. [2]

Ergebnis: Die Domäne „Orchester“ wurde erfolgreich abgebildet.

Aufgabe 2: Objekt [20 + 4]

Zweck: Vergleichen von Klassendiagramm (Entwicklungszeit) und Objektdiagramm (Laufzeit).

Beschreibung: Gegeben sei folgende Tabelle mit Objektdiagrammen linkerhand (Zeilenanfang) und Klassendiagrammen oberhalb (Spaltenkopf).

		1	2	3	4
					
A					
B					
C					
D					
E					
F					
G					

Geben Sie für jedes Objektdiagramm an, auf welchem Klassendiagramm es basieren könnte! Schreiben Sie Ihre Lösung in eine Textdatei namens aufgabe_2.txt!

[20 Punkte gefordert, aber 24 Punkte kann man verdienen; +2 wenn richtig; -2 wenn falsch]

Hinweis: In der Lösungsdatei sollte zu Beginn einer jeden Zeile der entsprechende Buchstabe und dahinter mögliche Spalten stehen. Hypothetisches Beispiel für eine Zeile "X" mit den Spalten 1 und 3 als dazu passende Klassendiagramme:

```
X: 1, 3
```

Hinweis: Um Zeit und Denkarbeit zu sparen, könnte man die Klassen optional in Java implementieren, mit je einem Paket oder Projekt pro Klassendiagramm. Dann gibt man jedem Paket eine separate Startklasse mit „main“-Methode. Dort erzeugt man die Objekte für die Zeilen A bis G und verknüpft sie miteinander mittels Assoziation (Zuweisung zu Attributen). Ob eine Variante funktioniert oder nicht, zeigt dann der Compiler an.

Ergebnis: Die möglichen Klassendiagramme wurden jedem Objektdiagramm zugeordnet.

Aufgabe 3: Paket [20]

Zweck: Modellieren einiger Aspekte des Turnens in einem Paketdiagramm.

Hinweise: Da in BOUML kein Paketdiagramm existiert, kann ein beliebiges passendes anderes verwendet werden, um Pakete darzustellen. Auch hat BOUML keine „White Box View“ für Pakete. Daher sind Schachtelungen grafisch selbst abzubilden, indem auch innere Pakete oder Klassen extra mit eingezeichnet werden.

a) Erstellen Sie ein passendes Diagramm, um Pakete abbilden zu können! [2]

b) Setzen Sie seine Eigenschaften so, dass Paketnamen im Karteireiter erscheinen! Stellen Sie als Hintergrundfarbe für Pakete sehr helles Orange ein! [2]

c) Es gibt ein Paket „turnen“, zu dem alle folgenden Inhalte gehören. [2]

Hinweis: Dieses Paket muss nicht zwingend im Diagramm erscheinen.

d) In ihm befindet sich ein Paket „geraet“ mit zwei Unterpaketen „maenner“ und „frauen“. [2]

e) Zum Paket „maenner“ gehören die folgenden Klassen: Boden, Sprung, Barren, Reck, Pauschenpferd, Ringe. [2]

f) Zum Paket „frauen“ gehören die Klassen: Boden, Sprung, StufenBarren, Schwebebalken. [2]

g) Das Paket „disziplin“ hat drei Unterpakete: geraetTurnen, trampolinTurnen, gymnastik. [2]

h) Drei weitere Pakete befinden sich unterhalb des Paketes „turnen“: uebung, wettkampf, sportler. [2]

i) Die Pakete „uebung“ und „wettkampf“ hängen über eine Standardbeziehung vom Paket „geraet“ ab. Stellen Sie auch das entsprechende Stereotyp dar! [2]

j) Das Paket „sportler“ greift öffentlich auf „disziplin“ zu. Das Unterpaket „geraetTurnen“ wiederum hängt von „geraet“ auf eine Weise ab, dass dessen Inhalte NICHT an von „disziplin“ abhängige Pakete weitergegeben werden. [2]

Ergebnis: Die geforderten Pakete mitsamt ihren Abhängigkeiten wurden korrekt dargestellt.

Aufgabe 4: Zustand [20]

Zweck: Darstellen eines „commit“-Vorganges mittels Zustandsmaschinendiagramm.

Beschreibung: Vorausgesetzt wird das Versionierungssystem Subversion (SVN), dessen Client-seitigen Zustände während eines commit Vorganges dargestellt werden sollen.

Hinweis: Es sind Zustände zu modellieren! Insofern Aktionen gegeben sind, so führen diese zum Übergang von einem in einen anderen Zustand.

a) Erstellen Sie eine Zustandsmaschine namens „commit“! Stellen Sie sie im Diagramm dar! [2]

b) Markieren Sie Anfang und Ende durch entsprechende Elemente! [2]

c) Modellieren Sie einen konfliktfreien „commit“-Vorgang, bestehend aus den Schritten: update, enterStatement, commit! [2]

d) Während jedes dieser Schritte kann der Anwender die SVN-Hilfe konsultieren, was als Zustand „consultingHelp“ darzustellen ist! Verwenden Sie dafür einen parallelen Vorgang vom Anfang bis zum Ende der Maschine! [2]

e) Falls geänderte Dateien Konflikte hervorrufen, so wird dies nach dem update-Vorgang festgestellt und führt zum Zustand „conflictDetermined“. [2]

f) Bezeichnen Sie diesen Strang mittels Bedingung „[conflict]“! [2]

Hinweis: Um Bedingungen darzustellen, sind die Einstellungen des Diagrammes zum Zeichnen anzupassen.

g) Das Auflösen der Konflikte kann auf dreierlei Weise geschehen. [2]

h) Die resultierenden Zustände sind: „revertedToCurrentVersion“, „committedOwnVersion“, „mergedBothVersions“. [2]

i) Führen Sie diese drei wieder zusammen! [2]

j) Führen Sie den Hauptstrang und Konfliktlösungsstrang wieder zusammen! [2]

Ergebnis: Die Zustandsmaschine stellt den „commit“-Vorgang vollständig und korrekt dar.

Aufgabe 5: Muster [10]

Zweck: Umsetzen eines Software-Musters in Programmquelltext.

Beschreibung: Die Ressourcen einer Anwendung sollen durch eine „ResourceManager“-Klasse zentral verwaltet werden. Die Klasse soll zumindest eine „load“ und eine „save“-Methode haben, welche beide ein Argument vom Typ „File“ entgegennehmen. Um sicherzustellen, dass anwendungsweit nur eine Instanz dieser Klasse existiert, ist das „Singleton“ Pattern anzuwenden.

Geben Sie den Quelltext der Klasse in Java-Syntax an!

[2: Klasse; 2: Attribut; 2: Konstruktor; 2: Zugriffsmethode; 2: andere Methoden]

Ergebnis: Das Muster wurde korrekt implementiert.

Viel Erfolg!