Examination
Computer Programming (CP)

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

Personal Data	
First and Last Name	Ihre Daten werden von der Klausuraufsichtsperson schriftlich auf der Anwesenheitsliste festgehalten.
Matriculation Number	
Subject and Year	CS 2017-2
Login	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!

Examination Data	
Date	
Duration [min]	108 von insgesamt 180
Maximum Points [Point]	60
Permitted Study Aids	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.
Remarks	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.

Evaluation											
Task	1	2	3	4	5	6	7	8	9	10	Summe
Points	10	10	10	20	10	0	0	0	0	0	60

Ihre Programmierkenntnisse werden anhand kleiner, voneinander unabhängiger Java-Programme geprüft.

Task 1: Structured Programming [10 + 2]

Zweck: Zerlegung einer Zahl in Faktoren.

a) Erstellen Sie ein Programm mit zwei Ganzzahlen als lokale Variablen “n” und “t”! Initialisieren Sie n mit dem Wert 18844 und t mit dem Wert 2! [2]

b) Implementieren Sie eine “while”-Schleife mit dem Laufkriterium n > 1! [2]

c) Prüfen Sie mittels Modulo-Operator, ob eine ganzzahlige Division möglich ist! [2]

d) Falls ja, so führen Sie die Division durch und weisen das Ergebnis der Variablen n zu! Geben Sie anschließend den Teiler t und die Zahl n auf der Konsole aus! [2]

e) Falls nein, so inkrementieren Sie die Variable t! [2]

Zusatz f) Welche Primzahl bleibt am Ende übrig? [2]

Ergebnis: Die Zahl 18844 wurde in Faktoren zerlegt und ihre Teiler ausgegeben.

Task 2: String Usage [10]

Zweck: Verwendung von Instanzmethoden der “String”-Klasse.

a) Nehmen Sie drei Kommandozeilenargumente entgegen und speichern Sie sie in lokalen Variablen! Geben Sie sie zur Kontrolle auf der Konsole aus! [2]

Hinweis: Das erste Argument repräsentiert eine Zeichenkette; das zweite und dritte jeweils einen Index innerhalb der Zeichenkette.

b) Bestimmen Sie mittels geeigneter Instanzmethode der “String”-Klasse die durch die beiden Indexe begrenzte Teilzeichenkette und speichern Sie sie in einer Variablen! [2]

c) Hängen Sie mittels Instanzmethode der “String”-Klasse die Endung “.png” an die Zeichenkette an und geben Sie die resultierende Zeichenkette auf der Konsole aus! [2]

d) Prüfen Sie, ob der Anfangsindex nicht negativ und kleiner als der Endindex ist! [2]

e) Stellen Sie außerdem sicher, dass der Endindex kleiner als der Zeichenkette Länge ist! [2]

Ergebnis: Ein Dateiname wurde testweise zusammengebaut und ausgegeben.

Task 3: Static Attribute [10]

Zweck: Instanzen einer Klasse sollen automatisch eine laufende Nummer erhalten.

a) Erstellen Sie eine Klasse namens “Beleg” und geben Sie ihr ein ganzzahliges Instanzattribut namens “nummer”, wofür Zugriffsmethoden zu erstellen sind! [2]

b) Führen Sie eine Klassenvariable “zaehler” des Typs “int” ein und initialisieren Sie sie mit dem Wert 10.000! [2]

c) Stellen Sie mittels Standardkonstruktor sicher, dass jede neue Instanz eine automatisch erhöhte Belegnummer zugeordnet bekommt! [2]

d) Erzeugen Sie in einer Startklasse drei Objekte der Klasse “Beleg” und geben Sie ihre jeweilige Nummer auf der Konsole aus! [2]

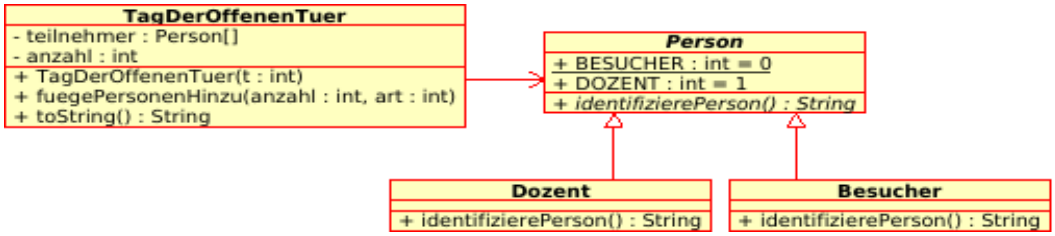
Hinweis: Eine Kapselungsmethode ist nötig.

e) Setzen Sie willkürlich die Nummer des dritten Belegobjektes auf 5 und geben Sie sie zur Kontrolle via Konsole aus! [2]

Ergebnis: Die erzeugten Testobjekte wurden erfolgreich durchnummeriert.

Task 4: Object Oriented Programming (OOP) [20]

Zweck: Einige Teilnehmer am Tag der offenen Tür sollen erfasst werden.



Erstellen Sie die Klassen gemäß gegebenem UML-Klassendiagramm! Lassen Sie die Methoden “identifiziere Person()” jeweils eine beliebige, eindeutige Zeichenkette zurückgeben!

Hinweis: Der Klassenname “Person” sowie die Methode “identifiziere Person()” stehen kursiv. Die beiden Attribute “BESUCHER” und “DOZENT” sind unterstrichen und sollen konstant sein.

a) Klasse “Person” [2]

b) Klasse “Dozent” [2]

c) Klasse “Besucher” [2]

d) Klasse “TagDerOffenenTuer” [2]

e) Erzeugen Sie das Feld (Array) “teilnehmer” im Standardkonstruktor! [2]

f) Erzeugen Sie in der Methode “fuegePersonenHinzu()” per “for”-Schleife so viele Personen, wie per Parameter vorgegeben! [2]

g) Nehmen Sie dazu per “if-else”-Verzweigung unter Verwendung der beiden Konstanten der Klasse “Person” eine Fallunterscheidung vor, um den Typ der zu erzeugenden Person herauszufinden! [2]

h) Lassen Sie die “toString()”-Methode alle Teilnehmer zurückgeben! [2]

i) Erstellen Sie eine Startklasse namens “Launcher” mit “main”-Methode, in welcher die Klasse “TagDerOffenenTuer” mit einer Kapazität von 100 Teilnehmern instanziiert wird! [2]

j) Fügen Sie der Instanz mittels der Methode “fuegePersonenHinzu()” 10 Dozenten und 50 Besucher hinzu! Geben Sie alle Teilnehmer via “toString()”-Methode aus! [2]

Ergebnis: Das Klassengerüst wurde erfolgreich verwendet.

Task 5: Exception Handling [10]

Zweck: Eine Ausnahme soll provoziert und abgefangen werden.

a) Erstellen Sie eine Startklasse namens “Launcher” mit der Sichtbarkeit “public”! Erstellen Sie im gleichen Modul eine nicht öffentliche Klasse namens “Wrapper”, welche ein Attribut namens “value” vom Typ “int” enthält, das mit 0 initialisiert wird! [2]

b) Erstellen Sie eine Klasse namens “TestException”, welche eine Ausnahme repräsentiert! Implementieren Sie einen Konstruktor, der eine Fehlernachricht entgegennimmt! [2]

c) Geben Sie der “Launcher”-Klasse eine statische Methode namens “add” ohne Rückgabewert, aber mit den zwei Parametern “w” vom Typ “Wrapper” und “i” vom Typ “int”! Führen Sie eine Fallunterscheidung durch, indem Sie prüfen, ob der Wert von “i” kleiner als 10 ist! [2]

d) Falls ja, so erhöhen Sie die Eigenschaft “value” des übergebenen “Wrapper”-Objektes um den Wert von “i”! Falls nein, so werfen Sie eine Ausnahme vom Typ “TestException” mit übergebener Fehlernachricht! [2]

e) Instanziieren Sie in der “main”-Methode die “Wrapper”-Klasse! Rufen Sie ihre “add”-Methode auf! Provozieren Sie einen Fehler und fangen Sie ihn ab! [2]

Ergebnis: Das Programm läuft auch im Fehlerfall stabil und wird kontrolliert beendet.

Viel Erfolg!