Prüfung
Computer Programming (CP)

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	CS20-1
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	Dezember 2020
Dauer [min]	120
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	20	20	20	30	10	100

Im Rahmen der Klausur sind fünf verschiedene Aufgaben zu lösen.

Aufgabe 1: Array „Buchstaben“ [20]

Zweck: Ausgeben der Kleinbuchstaben a bis l auf dem Bildschirm, im Format:

```
abcd
efgh
ijkl
```

a) Erstellen Sie ein Java-Programm mit Hauptklasse und Startmethode! [2]

b) Deklarieren Sie eine lokale Variable namens „a“ als zweidimensionales Feld (Array) mit Elementen des Typs „char“! [2]

c) Erzeugen Sie dieses Feld mit den Dimensionen 3 und 4! [2]

d) Deklarieren Sie eine weitere lokale Variable „c“ vom Typ „char“ mit dem Initialwert 97! [2]

Hinweis: Dies ist ein ASCII- bzw. Unicode.

e) Definieren Sie eine Klassenmethode namens „fill“, die keinen Rückgabewert hat! [2]

f) Rufen Sie sie aus der „main“-Methode heraus auf und übergeben Sie beide lokalen Variablen (Feld und char-Wert) als Parameter! [2]

g) Befüllen Sie das Feld innerhalb der „fill“-Methode per geschachtelter Zählschleifen! [2]

h) Weisen Sie dem ersten Element den übergebenen „char“-Parameter zu und inkrementieren Sie ihn einfach, so dass schließlich die Kleinbuchstaben a-d, e-h sowie i-l zugewiesen sind! [2]

i) Definieren Sie eine weitere statische Methode namens „print“, welche das zweidimensionale Feld als Parameter entgegennimmt und rufen Sie sie aus der Startmethode heraus auf! [2]

j) Geben Sie den Inhalt des Feldes in der „print“-Methode aus, indem Sie zwei „for-each“-Schleifen ineinander schachteln! [2]

Ergebnis: Drei Zeilen mit je vier Buchstaben wurden auf der Konsole ausgegeben.

Aufgabe 2: Wrapper Class „Zahlen“ [20 + 2]

Zweck: Nutzung statischer Methoden der Wrapper-Klassen zur Ausgabe von Zahlenwerten.

Im ersten Abschnitt geht es um Gleitkommazahlen.

a) Erzeugen Sie mittels des „new“-Operators ein „Double“-Objekt mit dem Initialwert 0.0 und weisen Sie es einer lokalen Variablen namens „arg“ zu! [2]

b) Prüfen Sie per Bedingung, ob genau ein Kommandozeilenargument übergeben wurde! [2]

c) Falls ja, so wandeln Sie das Kommandozeilenargument in den „Double“-Objekttyp um und weisen den resultierenden Wert der Variablen „arg“ zu! [2]

Hinweis: Verwendung der statischen Methode einer passenden „Wrapper“-Klasse.

d) Bestimmen Sie aus dem „arg“-Objekt per Instanzmethode seinen primitiven Wert und speichern Sie ihn in einer lokalen Variablen „d1“! [2]

e) Deklarieren Sie eine Variable „d2“ und initialisieren Sie sie mit dem Wert „0.5“! [2]

f) Addieren Sie d1 und d2 und speichern Sie das Ergebnis in einer Variablen namens „sum“! [2]

g) Kapseln Sie den Variablenwert von „sum“ in einem neu zu erstellenden Objekt „d“! [2]

Hinweis: Manuelles Kapseln, OHNE Verwendung von „Autoboxing“.

h) Geben Sie den Wert des Objektes „d“ auf der Konsole aus! Wandeln Sie ihn dazu explizit in eine Zeichenkette um! [2]

Im zweiten Abschnitt geht es um Ganzzahlen.

i) Deklarieren Sie eine primitive Ganzzahl und initialisieren Sie sie mit dem arithmetischen Wert „106“! [2]

j) Geben Sie den Wert unter Verwendung passender, statischer Methoden der „Integer“-Klasse in dreierlei Form auf der Konsole aus: als Binär-, Oktal- und Hexadezimalzahl! [2]

Zusatzaufgabe k) Stellen Sie die Hexadezimalzahl in Großbuchstaben dar! [2]

Ergebnis: Folgende Ausgaben sind auf der Konsole zu sehen:

```
Gleitkommazahl: 0.5
Dezimalzahl: 106
Dualzahl: 1101010
Oktalzahl: 152
Hexadezimalzahl: 6A
```

Aufgabe 3: Debugging „Zähler“ [20]

Zweck: Säubern eines vorgegebenen Programmes von Fehlern sowie dessen Inspektion.

Gegeben seien zwei Dateien namens „Counter.java“ und „Launcher.java“. Bereinigen Sie das vorgegebene Programm, so dass es fehlerfrei läuft!

a) Paket [2]

b) Vererbung [2]

c) Initialwert [2]

d) Sichtbarkeit [2]

e) Attributzugriff [2]

f) Kommandozeilenfeld [2]

g) Standardkonstruktor [2]

h) Variable [2]

Hinweis: Schreiben Sie die Antworten auf die folgenden beiden Fragen als Kommentar in die Zählschleife der „count“-Methode der „Counter“-Klasse!

i) Was muss man setzen, um den Debugger den Programmablauf an einer bestimmten Stelle unterbrechen zu lassen? [2]

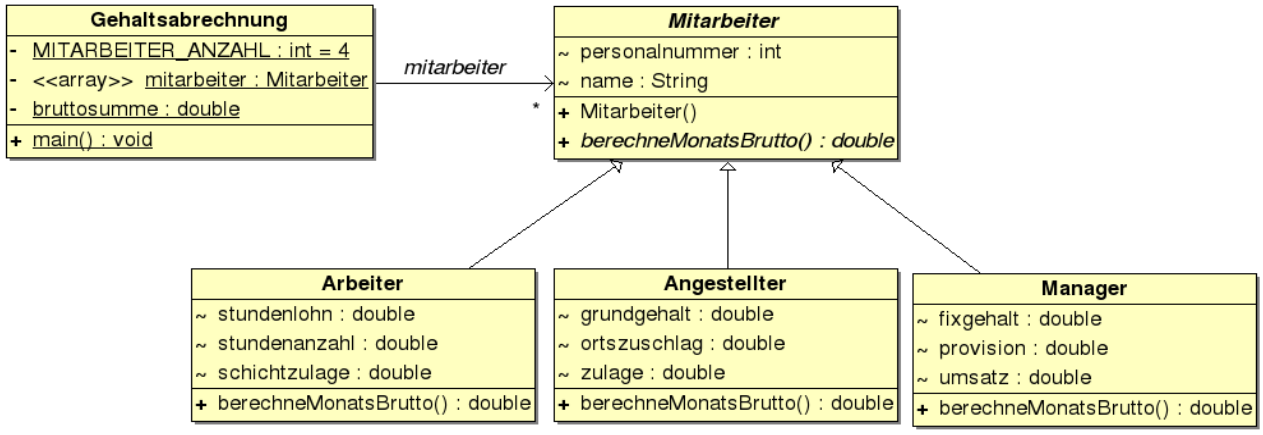
j) Finden Sie unter Verwendung des Java-Debuggers heraus, welchen Wert die Variable „result“ in der „count“-Methode der „Counter“-Klasse am Ende des fünften Schleifendurchlaufes hat!

Ergebnis: Das Programm läuft einwandfrei.

Aufgabe 4: OOP „Mitarbeiter“ [30]

Zweck: Implementierung einer vorgegebenen Anwendungsarchitektur.

Gegeben sei folgendes Klassendiagramm in Unified Modeling Language (UML) Notation:



Hinweis: In Kursivschreibweise stehen die Klasse „Mitarbeiter“ und ihre Methode „berechneMonatsBrutto()“.

Erstellen Sie die im Diagramm gezeigte Klassenstruktur als Java-Quelltext! Schreiben Sie sämtliche Klassen in eine einzige Datei namens „Gehaltsabrechnung“!

a) Klassen [2]

b) Abstrakte Klasse [2]

c) Vererbung [2]

d) Attribute [2]

e) Behälter (Container, Feld, Array) [2]

f) Statische Attribute [2]

g) Initialwerte [2]

h) Methoden [2]

i) Typen [2]

j) Sichtbarkeiten [2]

k) Vervollständigen Sie die überschriebene Methode „berechneMonatsBrutto“ in den drei Unterklassen, indem Sie den drei Arten Mitarbeitern folgende Berechnungsformel für das Gehalt verpassen [2]:

```
Arbeiter: this.stundenlohn * this.stundenanzahl + this.schichtzulage
Angestellter: this.grundgehalt + this.ortszuschlag + this.zulage
Manager: this.fixgehalt + this.umsatz * this.provision / 100
```

l) Erzeugen Sie in der „main“-Methode das in der Klasse „Gehaltsabrechnung“ definierte Feld (Array) „mitarbeiter“ als Objekt mit der Größe „MITARBEITER_ANZAHL“! [2]

m) Befüllen Sie es mit zwei Objekten des Typs „Arbeiter“ und je einem Objekt der Typen „Angestellter“ und „Manager“! [2]

n) Weisen Sie diesen Mitarbeiterobjekten zu Testzwecken nach Belieben Werte für die Attribute „name“ und/oder „personalnummer“ zu! Weisen Sie außerdem für die Berechnung nötige Werte für die Attribute der jeweiligen Unterklasse zu, beispielsweise „stundenlohn“, „grundgehalt“, „fixgehalt“ usw.! [2]

o) Berechnen Sie das Gesamtbruttogehalt („bruttosumme“) durch Iterieren über alle Mitarbeiterobjekte und geben Sie das Ergebnis auf der Konsole aus! [2]

Ergebnis: Die Anwendung ist lauffähig und gibt das Gesamtbruttogehalt auf der Konsole aus.

Aufgabe 5: Exception „Grundrechenarten“ [10]

Zweck: Ergänzen einer Anwendung für Grundrechenarten um Ausnahmebehandlung.

Gegeben sei dazu ein lauffähiges Programm bestehend aus zwei Dateien namens „Launcher.java“ und „Calculator.java“.

a) Erstellen Sie eine neue Ausnahme-Klasse namens „UnknownOperatorException“! [2]

b) Geben Sie ihr einen Konstruktor, der die Fehlernachricht als Parameter entgegennimmt und an den Konstruktor der Superklasse weiterleitet! [2]

c) Lassen Sie die „calculate“-Methode der „Calculator“-Klasse diese Ausnahme standardmäßig werfen, falls der übergebene Operator nicht zu den vier Grundrechenarten gehört! [2]

d) Fangen Sie die geworfene Ausnahme in der „main“-Methode ab! Fangen Sie sie möglichst weit innen, d. h. an der Stelle ihres Auftretens ab, so dass das Programm nach einem aufgetretenen Fehler weiterhin auf Eingaben wartet! [2]

e) Geben Sie im Fehlerfalle eine Nachricht auf der Standard-Fehlerkonsole aus! Fügen Sie auch die im Ausnahme-Objekt enthaltene Nachricht an! [2]

Ergebnis: Die Anwendung fängt Ausnahmen ab.

Viel Erfolg!