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	CS20-2
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	März 2021
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	30	10	10	30	100

Im Rahmen der Klausur sind fünf verschiedene Aufgaben zu lösen.

Aufgabe 1: Static Method „Betriebssystem“ [20]

Zweck: Ausgabe des Namens des Betriebssystems mit einer fiktiven Versionsnummer.

a) Erstellen Sie eine Startklasse namens „Launcher“ mit Einstiegsmethode! [2]

b) Deklarieren Sie eine lokale Zeichenkettenvariable „os“! [2]

c) Bestimmen Sie mittels einer geeigneten Methode der „java.lang.System“-Klasse die Systemeigenschaft „os.name“! [2]

d) Weisen Sie sie der Variablen „os“ zu und geben Sie diese auf der Konsole aus! [2]

e) Erzeugen Sie unter Verwendung einer Methode der „Math“-Klasse eine Zufallszahl zwischen 0.0 (inklusive) und 1.0 (exklusive)! Speichern Sie sie in einer lokalen Gleitkommavariablen namens „r“! [2]

f) Transponieren Sie die Zahl „r“ in den Bereich zwischen 0 (inklusive) und 10 (exklusive)! Speichern Sie das Ergebnis in einer lokalen Gleitkommavariablen namens „t“! [2]

g) Runden Sie die Zahl „t“ unter Verwendung einer Methode der „Math“-Klasse auf eine Nachkommastelle! Speichern Sie das Ergebnis in einer lokalen Gleitkommavariablen „d“! [2]

h) Konvertieren Sie den Wert von „d“ in eine Zeichenkette! Speichern Sie sie in einer lokalen Variablen namens „s“! [2]

i) Verketten Sie die weiter oben definierte Variable „os“ mit der Variablen „s“, wobei zwischen beiden ein Leerzeichen stehen soll! Speichern Sie das Ergebnis in einer lokalen Variablen namens „v“! [2]

j) Geben Sie das Literal „Betriebssystem: “ mit dem Variablenwert v auf der Konsole aus! [2]

Ergebnis: Der Name des Betriebssystems wurde mit einer Versionsnummer ausgegeben.

Aufgabe 2: Structure „Grundrechenarten“ [30]

Zweck: Erstellung einer Anwendung für Grundrechenarten mit Ganzzahlen ohne Kommastellen.

a) Erzeugen Sie ein „Scanner“-Objekt zur Entgegennahme von Eingaben via Konsole! [2]

b) Fordern Sie den Benutzer durch Ausgabe eines kleinen Textes auf der Konsole zur Eingabe eines ersten Operanden auf! Nehmen Sie den Operanden über die Konsole mittels „Scanner“-Objekt als Ganzzahl entgegen! Speichern Sie ihn in einer lokalen Variablen! [2]

c) Wiederholen Sie das Prozedere für einen zweiten, ganzzahligen Operanden! [2]

d) Nehmen Sie schließlich noch den Operator als Zeichenkette entgegen! [2]

e) Deklarieren Sie eine ganzzahlige Variable „r“ für das Speichern des Ergebnisses! Initialisieren Sie sie mit dem arithmetischen Wert „0“! [2]

f) Verwenden Sie eine switch-basierte Fallunterscheidung, um zu prüfen, ob es sich bei dem eingegebenen Operator um die arithmetische Grundrechenart „+“, „-“, „*“ oder „/“ handelt! [2]

g) Verwenden Sie für den Standardfall (default) die leere Anweisung „;“! [2]

h) Führen Sie in den jeweiligen Zweigen die passende Operation aus! [2]

i) Geben Sie den Wert von „r“ als Ergebnis auf der Konsole aus! [2]

j) Geben Sie die durch das „Scanner“-Objekt belegten Systemressourcen wieder frei! [2]

k) Umrahmen Sie den bisher geschriebenen Quelltext in der Weise durch eine „while“-Schleife, dass das Programm endlos läuft und nicht nach nur einer Operation beendet wird! [2]

l) Deklarieren Sie eine Zeichenkettenvariable namens „quit“ mit dem Initialwert „n“! [2]

m) Lassen Sie die Schleife laufen, solange „quit“ den Wert „n“ hat! [2]

n) Fragen Sie den Benutzer in jedem Schleifenzyklus, ob das Programm beendet werden soll! Nehmen Sie seine Eingabe mittels „Scanner“-Objekt als Zeichenkette entgegen und weisen Sie sie der Variablen „quit“ zu! [2]

o) Versehen Sie Startklasse und auch „main“-Methode mit einem Javadoc-Kommentar! [2]

Ergebnis: Die Grundrechenarten werden durch das Programm korrekt ausgeführt.

Aufgabe 3: String „Osterhase“ [10]

Zweck: Bearbeitung einer Zeichenkette.

a) Erstellen Sie ein Programm mit einer lokalen Variablen des Types „String“! Weisen Sie ihr den folgenden Wert zu: „Ostereier suchend hoppelte das Häschen durch den weißen Schnee, juchhe!“! Geben Sie sie auf der Konsole aus! [2]

b) Ersetzen Sie mittels geeigneter Instanzmethode der „String“-Klasse die Zeichenfolge „den weißen Schnee“ durch „das grüne Gras und fand etwas“! Geben Sie die resultierende Zeichenkette auf der Konsole aus! [2]

c) Zerlegen Sie die Zeichenkette unter Verwendung einer geeigneten Instanzmethode der „String“-Klasse mittels regulären Ausdrucks als Parameter in ihre Wortbestandteile! [2]

Hinweis: Das „\s“ ist dabei eine Kurzform für alle Arten von Leerzeichen als Trennzeichen. Beachten Sie, dass der rückwärtsgerichtete Schrägstrich (Backslash) maskiert werden muss!

d) Speichern Sie das zurückgegebene Feld (Array) von Einzelwörtern in einer lokalen Variablen namens „result“! [2]

e) Geben Sie den Inhalt des Feldes per Zählschleife auf der Konsole aus! [2]

Ergebnis: Die Zeichenkette wurde verändert und in Wörter zerlegt.

Aufgabe 4: Procedure „Grußformel“ [10]

Zweck: Ausgabe einer Begrüßung über die Konsole.

a) Erstellen Sie ein Programm, dass neben „main“ auch eine statische Methode namens „greet“ mit paketweiter Sichtbarkeit hat! [2]

b) Lassen Sie die Methode „greet“ einen Bool‘schen Wert als Parameter namens „evening“ entgegennehmen und eine Zeichenkette als Rückgabewert liefern. [2]

c) Definieren Sie darin eine lokale Zeichenkette „s“ und weisen Sie ihr den Wert „Guten Tag!“ zu! Führen Sie anhand des übergebenen Parameters eine Fallunterscheidung durch! [2]

d) Weisen Sie der Variablen „s“ im Erfolgsfall den Wert „Guten Abend!“ zu! Bestimmen Sie sie zum Rückgabewert der Methode! [2]

e) Rufen Sie die „greet“-Methode aus „main“ heraus zweimal auf, im ersten Fall mit dem Argument „false“ und im zweiten mit „true“! Geben Sie jeweils die zurückgelieferte Zeichenkette auf der Konsole aus! [2]

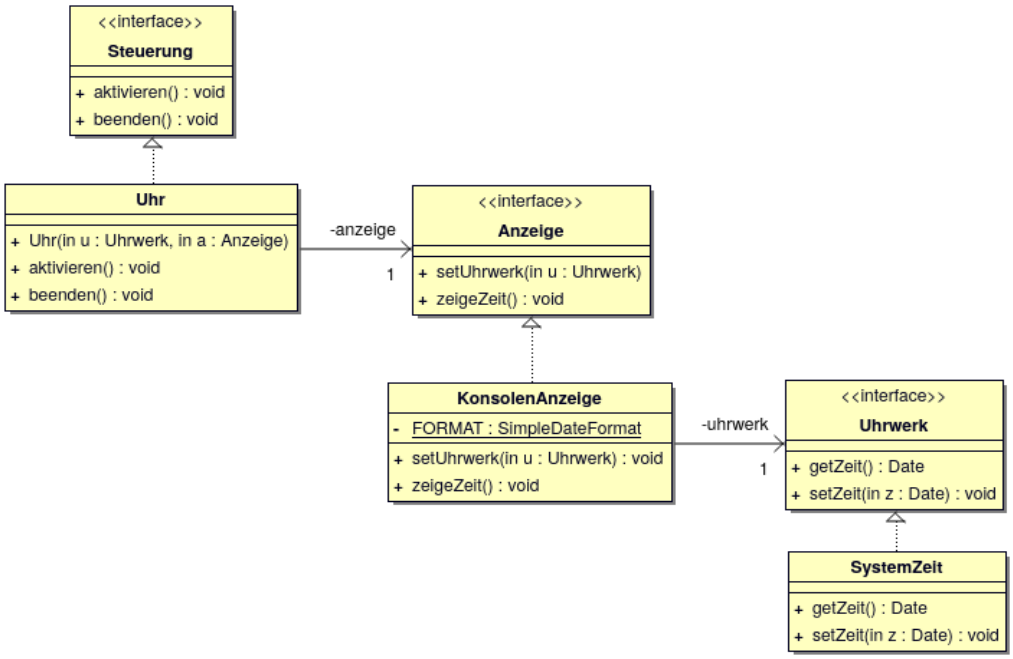
Ergebnis: Beide Grußformeln wurden auf der Konsole korrekt ausgegeben.

Aufgabe 5: OOP „Uhrenanzeige“ [30]

Zweck: Ausgabe der aktuellen Uhrzeit auf der Konsole im folgenden Format (Beispiel):

„Es ist gerade 12:35 Uhr und 48 Sekunden.“

Zur Orientierung sei die lauffähige Programmdatei „EinfacheUhr.java“ gegeben.



Eine Ausgabe der Uhrzeit im gleichen Format soll nun erzeugt werden durch ein Klassengerüst, wie dargestellt im obigen Klassendiagramm. Implementieren Sie es in Java-Quelltext!

Hinweis: Der Aufwand zum Bau eines solchen Frameworks ist zwar zunächst deutlich höher, zahlt sich jedoch später in größeren Anwendungen durch seine erweiterbare Architektur aus.

a) Schnittstelle „Uhrwerk“ und Methoden [2]

b) Klasse „SystemZeit“ und Methoden [2]

c) Lassen Sie die Methode „getZeit“ ein neu erzeugtes „Date“-Objekt zurückgeben! [2]

d) Lassen Sie die Methode „setZeit“ eine Ausnahme werfen, die vom Typ „UnsupportedOperationException“ ist! [2]

e) Schnittstelle „Anzeige“ und Methoden [2]

f) Klasse „KonsolenAnzeige“ mit privatem Attribut [2]

g) Konstante der Klasse „KonsolenAnzeige“ [2]

Hinweis: Weisen Sie der Konstante initial ein „SimpleDateFormat“-Objekt zu! Verwenden Sie als Parameter für den Konstruktor das folgende Literal:

```
"'Es ist gerade' HH:mm 'Uhr und' ss 'Sekunden.'"
```

h) Methoden der Klasse „KonsolenAnzeige“ [2]

i) Bestimmen Sie in der Methode „zeigeZeit“ die Uhrzeit via Attribut „uhrwerk“! Übergeben Sie sie als Argument an die „format“-Methode der „FORMAT“-Konstante! [2]

j) Schnittstelle „Steuerung“ und Methoden [2]

k) Klasse „Uhr“ mit privatem Attribut [2]

l) Initialisierender Konstruktor der Klasse „Uhr“ [2]

m) Methoden der Klasse „Uhr“ [2]

Die Methode „beenden“ bleibt leer; „aktivieren“ gibt via Attribut „anzeige“ die Zeit aus.

n) Erzeugen Sie in der „main“-Methode der Startklasse „Launcher“ ein „Uhr“-Objekt! [2]

o) Rufen Sie zunächst seine „aktivieren“-Methode und anschließend „beenden“ auf! [2]

Ergebnis: Auch das neue Programm ist lauffähig und gibt die aktuellen Uhrzeit aus.

Viel Erfolg!