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	CS 2019-1
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 2019
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	30	10	20	100

Im Rahmen der Klausur sind fünf verschiedene Aufgaben zu lösen.

Aufgabe 1: Structure „Rechteck“ [20]

Zweck: Berechnen von Fläche oder Umfang eines Rechtecks.

a) Erstellen Sie eine Startklasse namens „Launcher“ mit „main“-Methode! [2]

b) Erzeugen Sie darin ein „Scanner“-Objekt, das Eingaben von der Konsole entgegennimmt! [2]

c) Fordern Sie den Anwender zur Eingabe der „Seite A“ des Rechtecks auf! Nehmen Sie sie via „Scanner“ als Gleitkommazahl entgegen! [2]

d) Wiederholen Sie den Vorgang für „Seite B“! [2]

e) Bieten Sie dem Anwender über ein kleines Menü die Auswahl zwischen zwei Berechnungen an: 1 – Fläche; 2 – Umfang! Nehmen Sie die Auswahl via „Scanner“ entgegen! [2]

f) Nutzen Sie eine Fallunterscheidung mittels „switch“, um zwischen der Berechnung der Fläche und des Umfangs zu unterscheiden! [2]

g) Fügen Sie die Formel zur Berechnung der Fläche für den ersten Fall ein und geben Sie das Ergebnis auf der Konsole aus! [2]

h) Fügen Sie die Formel zur Berechnung des Umfangs für den zweiten Fall ein und geben Sie das Ergebnis auf der Konsole aus! [2]

i) Geben Sie für alle anderen Fälle eine Fehlermeldung aus! [2]

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

Ergebnis: Alle Werte werden korrekt berechnet.

Aufgabe 2: Array „Rätsel“ [20]

Zweck: Erstellen eines Rätsels mit drei Fragen und Antworten.

a) Initialisieren Sie im Programm eine lokale Zeichenkettenvariable namens „menue“ mit dem Literal "<1> <2> <3>"! [2]

b) Definieren Sie desweiteren drei Variablen des Types „String[]“, die jeweils eine Frage und zugehörige Antwort enthalten, mit folgenden Werten [2]:

```
"Wieviel ist 1 + 1?", "2"
"Wieviel ist 1 + 2?", "3"
"Wieviel ist 2 - 1?", "1“
```

c) Fügen Sie diese drei Variablen zu einem neu zu erstellenden, zweidimensionalen Zeichenkettenfeld (Array) namens „liste“ hinzu! [2]

d) Erzeugen Sie ein „Scanner“-Objekt, das Eingaben von der Konsole entgegennimmt! [2]

e) Deklarieren Sie schließlich noch eine Zeichenkettenvariable „antwort“ sowie eine ganzzahlige Variable „punkte“! [2]

f) Bauen Sie eine Zählschleife ein, die durch alle Elemente der Variable „liste“ läuft! [2]

g) Geben Sie auf dem Bildschirm die zum jeweiligen Schleifendurchlauf gehörende Frage aus! Geben Sie außerdem die Variable „menue“ aus! [2]

h) Fordern Sie den Anwender zur Eingabe einer Antwort auf! Nehmen Sie die Eingabe mittels „Scanner“ entgegen und speichern Sie sie in der Variablen „antwort“! [2]

i) Vergleichen Sie die Antwort des Anwenders mit der erwarteten Antwort! Geben Sie eine kurze Erfolgs- oder Mißerfolgsmeldung aus! Inkrementieren Sie im Erfolgsfall die Punkte! [2]

j) Teilen Sie dem Anwender abschließend mit, wieviele Punkte er erreicht hat! Geben Sie die Ressourcen des „Scanner“-Objekts frei! [2]

Ergebnis: Das Programm vergleicht die Antworten und liefert eine Erfolgsmeldung.

Aufgabe 3: Mathematics „Bruch“ [30]

Zweck: Arbeiten mit beliebig großen Brüchen.

a) Erstellen Sie eine Klasse namens „Fraction“, die einen Dezimalbruch darstellt! Geben Sie ihr zwei Attribute „numerator“ und „denominator“ vom Typ „BigInteger“! [2]

b) Erstellen Sie eine Konstruktormethode, die Zähler und Nenner entgegennimmt, um die beiden Attribute zu initialisieren! [2]

c) Bestimmen Sie unter Verwendung einer Instanzmethode der Klasse „BigInteger“ das größte gemeinsame Vielfache (GGV) (größter gemeinsamer Teiler) von Zähler und Nenner! Speichern Sie es in einer lokalen Variablen! [2]

d) Kürzen Sie nun Zähler und Nenner mit dem zuvor bestimmten GGV! [2]

e) Verschieben Sie ein möglicherweise negatives Vorzeichen aus dem Nenner in den Zähler! Prüfen Sie dazu per „signum“-Methode des „denominator“-Attributes, ob sein Vorzeichen negativ ist! Falls ja, so negieren Sie sowohl Zähler als auch Nenner mittels „negate“-Methode! [2]

f) Bauen Sie abschließend gleich zu Beginn des Konstruktors eine Prüfung ein, ob der übergebene Parameterwert für den Nenner ungleich arithmetisch Null ist, um Nulldivision auszuschließen! Werfen Sie für den Fall, dass er Null ist, eine „ArithmeticException“! [2]

Hinweis: Verwendung der Methoden „equals“ und „valueOf“ der „BigInteger“-Klasse.

g) Fügen Sie zwei weitere Konstruktormethoden ein, wovon die erste zwei „long“-Werte für Zähler und Nenner entgegennimmt und die zweite nur einen „long“-Wert für den Zähler! Delegieren Sie die Aufrufe weiter an den oben erstellten Konstuktor! [2]

Hinweis: Für den fehlenden Nenner sollte das Literal 1 übergeben werden.

h) Implementieren Sie die geerbte „toString“-Methode so, dass Sie für ganze Brüche mit dem Nenner 1 nur den Zähler ausgibt und für alle anderen Fälle sowohl Zähler als auch Nenner! Verwenden Sie dazu den ternären Operator! [2]

i) Implementieren Sie die geerbte „equals“-Methode zum Vergleich zweier Brüche! [2]

j) Implementieren Sie eine „negate“-Methode, die den Zähler des aktuellen Objektes (this) negiert und damit via Konstruktor ein neues „Fraction“-Objekt erzeugt und zurückgibt! [2]

Hinweis: Verwendung der Methode „negate“ der „BigInteger“-Klasse.

k) Implementieren Sie die Methoden „add“ und „subtract“, wobei jeweils ein Bruch als Operand übergeben und das Ergebnis ebenfalls als Bruch zurückgegeben wird! [2]

Hinweis: Man kann sich die Subtraktion sparen, wenn der Bruch negiert und an die Addition weitergereicht wird.

l) Implementieren Sie die Methoden „multiply“ und „divide“, wobei jeweils ein Bruch als Operand übergeben und das Ergebnis ebenfalls als Bruch zurückgegeben wird! [2]

m) Erstellen Sie eine Startklasse „Launcher“ mit „main“-Methode! Geben Sie folgende vier Brüche als „Fraction“-Objekte auf der Konsole aus! [2]

```
Brüche: 1/3, 2/-6, 3, 0
```

n) Führen Sie die vier Grundrechenarten mit folgenden Operanden aus und schreiben Sie die Ergebnisse auf die Konsole! [2]

```
Brüche: 1/3 + 1/6; 1/3 – 1/6; 2/3 * 1/6; 1/3 / 1/6
```

o) Vergleichen Sie die folgenden beiden Brüche und geben Sie das Ergebnis auf der Konsole aus! [2]

```
Brüche: 1/-3 mit -2/6
```

Ergebnis: Die „Fraction“-Klasse funktioniert einwandfrei.

Aufgabe 4: OOP „Ostern“ [10]

Zweck: Nachbauen eines objektorientierten Klassengerüstes.

Gegeben sei dazu ein UML-Klassendiagramm.

a) Erstellen Sie sämtliche Klassen! Berücksichtigen Sie Vererbungsbeziehungen! [2]

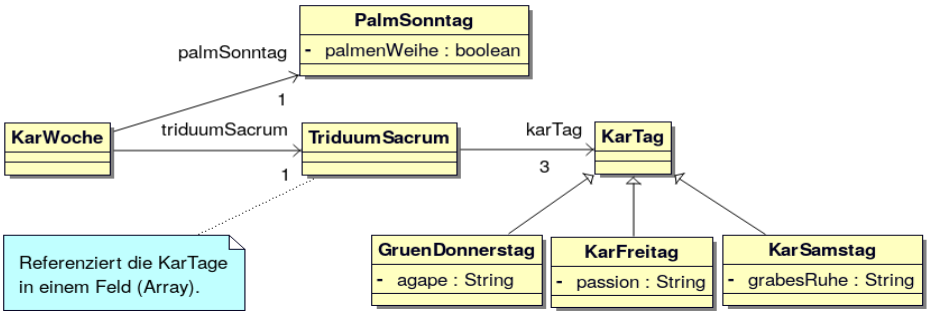
b) Implementieren Sie die in den Klassen gezeigten Attribute! Berücksichtigen Sie die Sichtbarkeiten! Stellen Sie öffentliche Zugriffsmethoden bereit! [2]

c) Implementieren Sie auch für die gezeigten Assoziationen passende Attribute! Stellen Sie sicher, dass im Konstruktor sowohl die zwei von „KarWoche“ referenzierten Objekte als auch das Behälterfeld (Container Array) in der Klasse „TriduumSacrum“ erzeugt und befüllt wird! [2]

d) Geben Sie der Klasse „KarWoche“ eine sinnvolle „toString“-Methode, die referenzierte Objekte und deren Eigenschaften auszugsweise ausgibt! [2]

e) Erstellen Sie in der Startklasse „Launcher“ mit „main“-Methode ein Objekt des Typs „KarWoche“! Weisen Sie beispielhaft einigen Attributen der von ihm referenzierten Objekte einen Wert zu! Geben Sie das „KarWoche“-Objekt auf der Konsole aus! [2]

Ergebnis: Die Klasse Karwoche wurde erfolgreich instanziiert und ihr Inhalt ausgegeben.



Aufgabe 5: Exception „Osternest“ [20]

Zweck: Methodenaufrufe und Abfangen von Ausnahmen im Programm.

a) Erstellen Sie eine Klasse namens „OsterNest” mit den ganzzahligen Attributen „kapazitaet“ und „eier“! [2]

b) Fügen Sie einen initialisierenden Konstruktor ein! [2]

c) Erstellen Sie eine Methode namens „hinzufuegen“, die ein Objekt vom Typ „OsterNest“ als Parameter empfängt! Addieren Sie die Eier des übergebenen zum aktuellen Nest hinzu! [2]

d) Erstellen Sie eine Methode namens „fuellen“, welche die Anzahl der Eier im aktuellen Nest um jeweils eins erhöht! Rufen Sie die Methode so lange rekursiv auf, bis das Osternest gefüllt ist! Bringen Sie in letzterem Falle eine Meldung auf der Konsole! [2]

e) Erstellen Sie eine Ausnahmeklasse „PlatzMangelException“! Geben Sie ihr einen Konstruktor, der die als Parameter übergebene Meldung weiterreicht an den Superkonstruktor! [2]

f) Lassen Sie die Methode „hinzufuegen” eine Ausnahme vom Typ „PlatzMangelException” werfen, falls die Anzahl der hinzuzufügenden Eier die Kapazität des Nestes übersteigt! [2]

g) Erzeugen Sie in der „main“-Methode der Startklasse namens „Launcher“ zwei Osternester, eines mit der Kapazität 10 und 2 Eiern und ein weiteres mit der Kapazität 20 und 20 Eiern! Geben Sie ihre Eierzahl auf der Konsole aus! [2]

h) Fügen Sie nun per „hinzufuegen“-Methode das zweite zum ersten Osternest hinzu! Geben Sie wiederum die aktuelle Eierzahl aus! [2]

i) Fangen Sie auftretende Ausnahmen ab! Geben Sie die entsprechende Nachricht auf der Fehlerkonsole aus! [2]

j) Befüllen Sie das erste Osternest mittels „fuellen“-Methode bis zu seiner Maximalkapazität! Geben Sie anschließend die aktuelle Eierzahl aus! [2]

Ergebnis: Eine Ausnahme wird beim Überschreiten der Kapazität geworfen.

Viel Erfolg!