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 2016-1
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	2016-12-12
Duration [min]	108 von insgesamt 180
Maximum Points [Point]	60
Permitted Study Aids	Dokumentation im lokalen Netzwerkverzeichnis (Intranet); NICHT gestattet sind Kommunikationsmöglichkeiten (Internet) oder Anmeldung via SSH auf dem Rechner "fileserv", wo Ihr Homeverzeichnis liegt, oder eine Anmeldung mit Ihrem Klausur-Login nach Ende der Prüfung. Dies kann leicht geprüft werden (last cs12*, 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. 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	10	20	0	0	0	0	0	60

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

Task 1: Structured Programming [10]

Ziel dieser Aufgabe ist die Erstellung eines Programmes für die vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division, wobei das Programm mit je drei Parametern aufgerufen werden soll, z. B.:

```
java Launcher 114.5 + 2.5
```

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

b) Prüfen Sie per Verzweigung, ob genau drei Argumente an der Kommandozeile übergeben wurden! Geben Sie anderenfalls folgende Nachricht auf der Standard-*Fehlerkonsole* aus! [2]

```
"Aufruf: java Launcher <number1> <operator> <number2>"
```

c) Deklarieren Sie drei lokale Variablen *x, y* und *z* als Gleitkommazahlen sowie eine vierte

Variable namens *op* als einzelnes Zeichen vom Typ *char*, das den Operator repräsentiert! [2]

Hinweis: Die Variablen *x* und *y* sollen als Operanden dienen und *z* das Ergebnis speichern.

d) Initialisieren Sie die Variable *z* mit dem arithmetischen Zahlenwert Null (*0*)! Weisen Sie den anderen Variablen das jeweils passende Kommandozeilenargument zu! [2]

Hinweis: Achtung! Konvertierungen sind nötig.

e) Nutzen Sie ein *switch-case* Konstrukt zur Unterscheidung der Rechenoperation in Abhängigkeit von der Variablen *op*! Fügen Sie die jeweils passende Berechnung ein und speichern Sie das Ergebnis in der Variablen z, welche auf der Konsole auszugeben ist! [2]

Hinweis: Im *default*-Zweig können Sie z. B. die Nachricht "Falscher Operator" ausgeben.

Hinweis: Der Umgang mit abzufangenden Ausnahmen ist Ihnen überlassen. Sie brauchen sie nicht zwingend abzufangen, sondern können sie auch einfach "nach draußen" werfen.

Task 2: Procedural Programming [10]

Ziel ist die Berechnung einer beliebig großen Fakultät n! = 1 * 2 * ... * n innerhalb eines Programmes, das folgendermaßen aufzurufen ist:

```
java Launcher <number>
```

a) Konvertieren Sie das an die *main*-Methode übergebene Kommandozeilenargument in eine Ganzzahl und speichern Sie sie in einer lokalen Variablen namens *n*! [2]

b) Deklarieren Sie eine weitere lokale Variable namens *f* vom Typ *BigInteger*! Weisen Sie ihr den Rückgabewert der aufgerufenen (noch zu erstellenden) statischen Funktion *factorial* zu! Geben Sie *f* schließlich auf der Konsole aus! [2]

c) Erstellen Sie eine Klassenfunktion namens *factorial*, welche ein Argument *n* vom Typ *int* entgegennimmt und einen Rückgabewert des Types *BigInteger* liefert! Initialisieren Sie eine lokale Variable *r* des Types *BigInteger* mittels dessen *valueOf*-Methode auf den Wert *1*! [2]

d) Nutzen Sie eine klassische Zählschleife, um gemäß der als Argument übergebenen Variablen *n* genau n mal zu iterieren! [2]

e) Multiplizieren Sie innerhalb der Schleife den aktuellen Wert der Ergebnisvariablen r mit dem der Schleifenlaufvariablen! [2]

Hinweis: Sie benötigen die Instanzmethode *multiply* sowie die Klassenmethode *valueOf* der Klasse *BigInteger*.

Task 3: Class Method [10]

Ziel dieser Aufgabe ist die Bestimmung des aktuellen Datums mittels Klassenmethode, so dass es in folgendem Format ausgegeben wird:

```
Sonntag, der 1.1.2017
```

a) Erstellen Sie eine neue Klasse namens *CurrentDate*! Geben Sie ihr eine Klassenmethode namens *get*, ohne Argumente aber mit einem Rückgabewert vom Typ *String*! [2]

b) Bestimmen und speichern Sie das aktuelle Datum in einer lokalen Variablen namens *d* vom Typ *Date*! [2]

c) Erzeugen Sie ein Objekt des Types *SimpleDateFormat* und konfigurieren Sie es via Konstruktor in gewünschter Weise! [2]

d) Formatieren Sie das Datumsobjekt mittels des *SimpleDateFormat*-Objektes! Bestimmen Sie die resultierende Zeichenkette zum Rückgabewert der Methode! [2]

e) Rufen Sie die *get*-Methode der Klasse *CurrentDate* aus der *main*-Methode einer zweiten Klasse heraus auf und lassen Sie das Ergebnis auf der Konsole ausgeben! [2]

Task 4: String and Array [10]

Ziel dieser Aufgabe ist das Zerlegen einer Zeichenkette in ihre Wort-Bestandteile.

```
public static String[] extract(String text, String delim);
```

a) Deklarieren Sie in der *main*-Methode eines neu erstellten Programmes eine lokale Zeichenkette namens *text* und weisen Sie ihr das folgende Literal zu [2]:

```
"Hugo, Emil, Fritz, Otto, Franz, Thomas, Willi, Emil, Fritz, Hugo."
```

b) Zerlegen Sie sie unter Verwendung einer passenden Methode der Klasse *String* in ihre Wort-Bestandteile und speichern Sie diese in einer zweiten lokalen Variablen namens *words*! [2]

c) Sortieren Sie die Wörter unter Verwendung einer passenden Methode der Klasse *Arrays*! [2]

d) Iterieren Sie mittels *for-each*-Schleife durch das Feld bzw. *Array* und lassen Sie seine Elemente auf der Konsole ausgeben! [2]

e) Markieren Sie auf der Konsole ausgegebene Duplikate! [2]

Hinweis: Dazu sind nötig: Temporäre Variable zum speichern des vorherigen *String*-Wertes, *String*-Vergleich, Verzweigung, Anpassung des ausgegebenen *Strings* durch Präfix "Duplikat: ".

Task 5: Object Oriented Programming (OOP) and Exception Handling [20]

Ziel dieser Aufgabe ist das Überprüfen des Alters von zu erstellenden Personen-Objekten.

a) Erstellen Sie eine Klasse namens *Person*! Geben Sie ihr die zwei Instanzattribute *age* (als Gannzahl) und *name* (als Zeichenkette)! Geben Sie ihr außerdem die zwei Konstanten MIN_AGE (mit dem Wert 0) und MAX_AGE (mit dem Wert 150)! [2]

b) Definieren Sie einen Konstruktor, der für beide Instanzattribute Initialwerte entgegennimmt! [2]

c) Überschreiben Sie die geerbte Methode *toString*, so dass sie den Namen und das Alter der Person ausgibt! Machen Sie mittels Annotation kenntlich, dass es sich um eine überschriebene Methode handelt! [2]

d) Definieren Sie eine zweite Klasse namens *OutOfRangeException*, welche von *RuntimeException* erbt! [2]

e) Deaktivieren Sie mittels Annotation Warnungen bezüglich einer fehlenden *serialVersionUID*! [2]

f) Definieren Sie einen Konstruktor, der die Argumente *value, min, max* vom Typ *long* entgegennimmt! [2]

g) Rufen Sie darin den Konstruktor der *Super*-Klasse auf, welchem -- unter Verwendung der übergebenen Argumente -- eine Fehlernachricht mitzugeben ist! [2]

Beispiel einer Fehlerausgabe:

```
Wert -1 liegt nicht im Bereich [0..150]
```

h) Ergänzen Sie nun den Konstruktor der Klasse *Person* durch eine Bedingung: Werfen Sie eine *OutOfRangeException*, falls sich das als Argument übergebene Alter außerhalb des gültigen Bereiches befindet! [2]

i) Erstellen Sie eine dritte Klasse mit *main*-Methode! Erzeugen Sie darin drei *Person*-Objekte, welche jeweils auf der Konsole auszugeben sind! Das erste Objekt soll dabei ein gültiges Alter erhalten, das zweite ein zu kleines, das dritte ein zu großes. [2]

j) Fangen Sie schließlich nacheinander möglicherweise auftretende Ausnahmen ab und geben Sie die Fehlernachricht der Ausnahme auf der Konsole aus! [2]

Viel Erfolg!