Examination
Data Processing (DP)

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 2016-2
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	2017-09-27
Duration [min]	120
Maximum Points [Point]	100
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	20	20	30	20	10	0	0	0	0	0	100

Die Nutzung der Java-Klassenbibliothek wird innerhalb fünfer Aufgaben geprüft. Dabei liegt der Fokus auf der Verarbeitung von Daten verschiedenster Formate.

Task 1: File Content and Logging [20]

Zweck: Eine Datei soll eingelesen werden.

a) Erstellen Sie ein Programm mit einer Startklasse namens "Launcher" und "main"-Methode! Erzeugen Sie darin eine Dateireferenz vom Typ "File" auf die gegebene Datei "scores.dat"! [2]

b) Verwenden Sie eine Instanz der "Scanner"-Klasse, um den Dateiinhalt einzulesen! Schließen Sie diese Systemressource später wieder! [2]

c) Iterieren Sie mittels "while"-Schleife durch den Dateiinhalt! [2]

d) Geben Sie die eingelesenen Daten mittels "printf"-Methode folgendermaßen formatiert auf der Konsole aus [2]:

```
| Theophanu | 60  |
| Hildegard | 20  |
| Heinrich  | 10  |
| Adelheid  | 70  |
| Wilhelm   | 30  |
| Liselotte | 50  |
| Henriette | 40  |
```

e) Fangen Sie bei der Ein-/Ausgabe eventuell auftretende Ausnahmen ab! [2]

f) Weisen Sie der Klasse eine statische Variable vom Typ "java.util.logging.Logger" zu! Initialisieren Sie sie mittels der "Logger"-Klasse "getLogger"-Methode! [2]

Hinweis: Als Logger-Name eignet sich "Logger.GLOBAL_LOGGER_NAME".

g) Schreiben Sie eine Methode namens "initLogger", in welcher dem Logger ein neuer "Handler" zugeordnet wird! Die Log-Nachrichten sollen in eine XML-Datei namens "log.xml" geschrieben werden! [2]

Hinweis: Standardmäßig ist dem Handler intern bereits ein "XMLFormatter"-Objekt zugeordnet.

h) Weisen Sie dem Logger eine Stufe ("Level") zu, die ihn Nachrichten vom Typ "INFO" oder solche höherer Priorität protokollieren lässt! [2]

i) Rufen Sie in der "main"-Methode gleich zu Beginn die "initLogger"-Methode auf! [2]

j) Protokollieren Sie den Programmablauf unter Zuhilfenahme des Loggers! Geben Sie insbesondere Fehlernachrichten im Falle einer Exception mit dem Level "SEVERE" aus! Überprüfen Sie die Funktionstüchtigkeit des Loggers anhand der generierten XML-Datei! [2]

Ergebnis: Der Dateiinhalt wird auf der Konsole angezeigt und Log-Einträge in die XML-Datei geschrieben.

Task 2: Container [20]

Zweck: Der Inhalt eines Warenkorbes soll dargestellt und berechnet werden können.

a) Erstellen Sie eine Klasse namens "Artikel" mit den drei Attributen "id" als Ganzzahl, "preis" als Gleitkommazahl und "menge" als Ganzzahl! [2]

b) Geben Sie der Klasse einen initialisierenden Konstruktor! Implementieren Sie außerdem alle Zugriffsmethoden! [2]

c) Überschreiben Sie die geerbte Methode "toString", so dass eine sinnvolle Ausgabe produziert wird! [2]

d) Erstellen Sie eine weitere Klasse namens "Warenkorb", welche intern "Vector" als Behälterdatenstruktur (Container) verwendet und kapselt! [2]

e) Erzeugen Sie das Container-Objekt im Standard-Konstruktor! [2]

f) Implementieren Sie eine Methode namens "add" zum Hinzufügen eines Artikels! [2]

g) Implementieren Sie eine Methode namens "berechneBestellwert", welche den Gesamtwert aller enthaltenen Artikel zurückliefert! Berücksichtigen Sie dabei auch die Menge! [2]

h) Überschreiben Sie die geerbte Methode "toString", so dass sämtliche Artikel ausgegeben werden! Verwenden Sie dafür die Klasse "StringBuilder"! [2]

i) Erzeugen Sie im Hauptprogramm ("main"-Methode) ein "Warenkorb"-Objekt! Fügen Sie ihm mindestens drei Artikel mit Werten Ihrer Wahl hinzu! [2]

j) Geben Sie den Inhalt des Warenkorbes auf der Konsole aus! Geben Sie den Gesamtbestellwert auf der Konsole aus! [2]

Ergebnis: Die Funktionstüchtigkeit des Warenkorbes wurde nachgewiesen.

Task 3: Stream [30]

Zweck: Daten sollen mittels des "Deflate"-Algorithmus' komprimiert werden.

a) Speichern Sie die Zeichenkette "ääääääääää bbbbbbbb cccccc" in einer lokalen Variablen! Geben Sie ihre Größe auf der Konsole aus! [2]

Achtung! Beim ersten Buchstaben handelt es sich nicht um ein "a", sondern den Umlaut "ä".

b) Wandeln Sie die Zeichenkette per Instanzmethode der "String"-Klasse um in ein Feld (Array) des Types "byte", welches ebenfalls lokal zu speichern ist! Geben Sie auch seine Größe auf der Konsole aus! [2]

Hinweis: Die Länge von String und "byte"-Array unterscheidet sich auf Grund der Kodierung, wobei für ASCII-Zeichen 1 Byte und für Umlaute 2 Byte stehen.

c) Erzeugen Sie für die spätere Datenausgabe unter Angabe der Länge des eben erstellten "byte"-Feldes (Arrays) ein Ausgabedatenstrom-Objekt des Types "ByteArrayOutputStream"! Vergessen Sie nicht, es am Ende zur Freigabe der Systemressource wieder zu schließen! [2]

d) Erzeugen Sie ein Objekt des Types "Deflater"! Ordnen Sie ihm das oben erstellte "byte"-Feld (Array) als Eingabedaten zu! Vergessen Sie nicht den "finish"-Aufruf! [2]

e) Komprimieren Sie die Eingabedaten und lassen Sie sie in einem neu erstellten Puffer-Feld (Array) der Größe 10 (zehn) speichern! [2]

f) Schreiben Sie die gepufferten Daten in den oben erstellten Ausgabedatenstrom! Geben Sie seine Größe auf der Konsole aus! [2]

g) Umhüllen Sie die beiden Methodenaufrufe zur Komprimierung und zum Schreiben in den Ausgabedatenstrom durch eine "while"-Schleife! Nutzen Sie als Laufkriterium die Instanzmethode "finished" der Klasse "Deflater"! [2]

Hinweis: Dies ist nötig, da der Puffer auf zehn Byte begrenzt und vorher nicht immer klar ist, wieviele Zeichen benötigt werden.

h) Wandeln Sie den Ausgabedatenstrom mittels einer seiner Instanzmethoden um in ein "byte"-Feld (Array), das als weitere lokale Variable zu speichern ist! [2]

i) Erzeugen sie ein "Inflater"-Objekt! Befüllen Sie es mit dem eben erhaltenen "byte"-Feld (Array)! [2]

j) Entpacken Sie die Daten! Nutzen Sie dafür das oben für die Kompression bereits erstellte Puffer-Feld (Array)! [2]

k) Schreiben Sie die entpackten Bytes wiederum in das Ausgabedatenstrom-Objekt! Geben Sie seine Größe auf der Konsole aus! [2]

l) Umhüllen Sie wiederum alles mit einer Schleife nach obigem Muster! [2]

m) Wandeln Sie den Ausgabedatenstrom wiederum um in ein "byte"-Feld (Array)! [2]

n) Dekodieren Sie dieses "byte"-Feld (Array) in eine Zeichenkette! Geben Sie ihre Größe auf der Konsole aus! [2]

Hinweis: Dazu bietet die Klasse "String" eine passende Konstruktormethode an.

o) Verwenden Sie an Stelle eines als Literal hart kodierten Transformationsformates wie "UTF-8" besser die Standard-Kodierung des Systems: Charset.defaultCharset().name()! [2]

Ergebnis: Die komprimierten Daten sind kleiner. Sie wurden korrekt wieder entpackt.

Task 4: Cloning [20]

Zweck: Ein Datenstapel (Stack) soll als Datenstruktur implementiert und genutzt werden.

a) Definieren Sie eine Klasse "IntegerStack", welche ganze Zahlen in einem Feld (Array) vorgegebener Länge speichern kann! Geben Sie ihr ein zweites Instanzattribut namens "top", welches auf den obersten bzw. höchsten Index zeigt! [2]

b) Implementieren Sie eine Konstruktormethode zur Initialisierung! Setzen Sie den Anfangswert von "top" auf "-1"! [2]

c) Geben Sie der Klasse eine Methode folgender Signatur, um einen Wert oben auf den Stapel legen zu können [2]:

```
public void push(int value)
```

d) Geben Sie der Klasse desweiteren eine Methode, um das oberste Element des Stapels zurückzuliefern und zugleich vom Stapel zu entfernen [2]:

```
public int pop()
```

e) Lassen Sie "IntegerStack" von der Schnittstelle "Cloneable" erben! [2]

f) Überschreiben Sie die "clone"-Methode so, dass tiefe Kopien erstellt werden! [2]

g) Erzeugen Sie im Hauptprogramm ("main"-Methode) ein Objekt des Types "IntegerStack" mit der Größe 10! Befüllen Sie es mit den Werten 0...9! [2]

h) Klonen Sie das "IntegerStack"-Objekt anschließend! [2]

i) Ersetzen Sie das oberste Element des ersten Stapel-Objektes (die Ziffer 9) durch die Zahl 100! [2]

j) Geben Sie das jeweils oberste Element des ersten und des zweiten Stapels auf der Konsole aus! [2]

Ergebnis: Die Funktionstüchtigkeit der Klasse "IntegerStack" ist nachgewiesen, wenn für den ersten Stapel 100 und für den zweiten 9 ausgegeben wird. Bei einer flachen Kopie würden beide Stacks dasselbe Array referenzieren, so dass statt 9 ebenfalls 100 ausgegeben würde.

Task 5: Formatting [10]

Zweck: Der Umgang mit Währungen wird geübt.

a) Bestimmen Sie unter Nutzung einer passenden Instanzmethode der Klasse "Currency" alle in der Java Runtime Environment (JRE) bzw. im System vorhandenen Währungen! Speichern Sie sie in einer lokalen Variablen des Types "Set", wobei dieser zu parametrisieren ist! [2]

b) Iterieren Sie mittels "for-each"-Schleife durch die Ergebnismenge (Set)! [2]

c) Geben Sie der Währung Namen und numerischen ISO 4217 Kode sowie ihr Symbol auf der Konsole aus! [2] Beispiel:

```
Russischer Rubel (neu) - 643 - RUB
Pakistanische Rupie - 586 - PKR
Philippinischer Peso - 608 - PHP
Burundi-Franc - 108 - BIF
```

d) Erzeugen Sie unter Angabe der Lokalisierung für in Deutschland gesprochenes Deutsch ein Währungsformat-Objekt! [2]

Hinweis: Methode der Klasse "DecimalFormat".

e) Nutzen Sie eine Instanzmethode dieses Objektes, um die Zahl 12345.6789 auf der Konsole auszugeben! [2]

Ergebnis: Alle dem System bekannten Währungen sowie 12.345,68 € wurden ausgegeben.

Viel Erfolg!