Examination
Data Processing (DP)

Prof. Dr.-Ing. Christian Heller <christian.heller@ba-leipzig.de>

Personal Data	
First and Last Name	Ihre Daten müssen von der Klausuraufsichtsperson schriftlich auf der Anwesenheitsliste festgehalten werden!
Matriculation Number	
Subject and Year	CS 2017-2
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	2018-09-
Duration [min]	120
Permitted Study Aids	Erlaubt ist die Nutzung von Dokumentationen im lokalen Netzwerk (Intranet) sowie Recherche im Internet. NICHT gestattet sind: * 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.
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. 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.

Evaluation						
Task	1	2	3	4	5	Summe
Desired Value [Credit Point]	20	10	10	20	40	100

Im Rahmen der Klausur sind fünf verschiedene Aufgaben zu lösen.

Task 1: Container [20]

Zweck: Eine verkettete Liste (Linked List) soll selbst gebaut und verwendet werden.

a) Erstellen Sie eine Klasse namens "ListNode" mit den zwei Attributen "data" vom Typ "Object" und "next" vom Typ "ListNode"! [2]

b) Geben Sie der Klasse einen parameterbehafteten Konstruktor zur Initialisierung! [2]

c) Implementieren Sie eine Suchmethode mit der folgenden Signatur [2]:

```
public static boolean contains(ListNode n, Object o);
```

d) Nutzen Sie das Prinzip der Iteration per “while”, um die Listenelemente zu durchlaufen! [2]

e) Als Abbruchkriterium soll dienen, dass es kein weiteres Listenelement gibt! [2]

f) Lassen Sie die “contains”-Methode “true” zurückliefern, wenn das übergebene Listenelement mit dem ebenfalls übergebenen Objekt inhaltlich übereinstimmt! [2]

Hinweis: Vergleich der Inhalte, NICHT Zeiger!

g) Erzeugen Sie in der "main"-Methode mindestens fünf Listenelemente! Weisen Sie Ihnen die Zeichenketten "n1" ... "n5" als Namen bzw. Daten-Objekt zu! [2]

h) Verketten Sie sie sinnvoll! [2]

Hinweis: Das letzte Element bzw. der letzte Knoten zeigt auf null.

i) Nutzen Sie die statische "contains"-Methode, um die Liste ausgehend vom ersten Knoten nach der Zeichenkette "n4" zu durchsuchen! [2]

j) Geben Sie das Ergebnis auf der Konsole aus! [2]

Ergebnis: Die Suchmethode der verketteten Liste wurde erfolgreich angewandt.

Task 2: Stream [10]

Zweck: Arbeiten mit Datenströmen.

a) Instanziieren Sie die Klasse "FileWriter" zur Ausgabe von Text in eine Datei namens "history.txt"! [2]

b) Kapseln Sie die Instanz in einem neu erzeugten Objekt des Types "PrintWriter"! [2]

c) Schreiben Sie folgenden Text in die Datei hinein [2]:

```
Call History:
```

d) Fügen Sie der Datei drei folgendermaßen formatierte Einträge hinzu [2]:

```
Call by Max Mustermann at local time: 14:21:59
```

Bestimmen Sie dabei die aktuelle Zeit per Calendar-Objekt!

e) Übergeben Sie dem Programm an der Kommandozeile sechs Zeichenketten-Argumente, die an Stelle von "Max Mustermann" als Vor- und Nachname einzusetzen sind! Vergessen Sie nicht, die “Stream”-Objekte zu schließen! [2]

Ergebnis: Die Inhalte wurden korrekt geschrieben, was per Dateimanager überprüfbar ist.

Task 3: Date and Time [10]

Zweck: Berechnen der Zeit in Tagen, Stunden, Minuten und Sekunden, die seit dem 2000-01-01T00:00:00 Uhr vergangen ist.

a) Erstellen Sie eine Instanz von "GregorianCalendar"! Initialisieren Sie sie auf 2000-01-01! [2]

b) Bestimmen Sie aus dem Kalenderobjekt die Zeit im Format "Date"! Bestimmen Sie aus dem "Date"-Objekt die Zeit in Millisekunden! [2]

c) Bestimmen Sie mit Hilfe der Klasse "System" die aktuelle Zeit in Millisekunden! Bilden Sie die Differenz aus aktueller Zeit und oben (in Teilaufgabe b) bestimmter Zeit! [2]

d) Berechnen Sie ausgehend von der Differenz in Millisekunden die Sekunden und Minuten! [2]

Hinweis: Die Operatoren "/" für Division und "%" (Modulo) für Restbestimmung sind geeignet.

e) Berechnen Sie weiterhin die Stunden und Tage! Geben Sie alle Werte zur Kontrolle auf der Konsole aus! [2]

Ergebnis: Die seit dem 2000-01-01 vergangene Zeit wurde bestimmt und ausgegeben.

Task 4: Bubble Sort [20]

Zweck: Anwendung des "Bubble Sort" Algorithmus'.

Kurzbeschreibung (Wikipedia): Bubblesort (Sortieren durch Aufsteigen oder Austauschsortieren) ist ein Algorithmus, der vergleichsbasiert eine Liste von Elementen sortiert. Pseudocode:

```
typedef int values[MAX];
void bubble_sort(values t) {
    int i, j, tmp;
    for (i = 1; i < MAX; i++) {
        for (j = 0; j < MAX - i; j++) {
            if (t[j] > t[j + 1]) {
                tmp = t[j+1];
                t[j+1] = t[j];
                t[j] = tmp;
            }
        }
    }
}
```

a) Erstellen Sie eine Startklasse, welche neben der "main"-Methode eine weitere statische Methode namens "sortBubble" enthält! [2]

b) Übergeben Sie ihr als Parameter ein Feld (Array) vom Typ "int"! [2]

c) Durchlaufen Sie das Feld (Array) per klassischer "for"-Schleife! [2]

d) Vertauschen Sie benachbarte Werte, so dass sich eine aufsteigende Reihenfolge ergibt! [2]

e) Umrahmen Sie diese innere durch eine äußere "for"-Schleife, welche beginnend mit dem Index 1 ebenfalls alle Elemente des Feldes (Arrays) durchläuft! [2]

f) Führen Sie eine Bool'sche Variable namens "swapped" ein, welche zu Beginn eines jeden Zyklus' der äußeren Schleife auf "false" zurückgesetzt wird! Setzen Sie sie im Falle des Vertauschens zweier Werte auf "true"! Brechen Sie die äußere Schleife ab, sobald die Variable "swapped" den Wert "false" hat! [2]

Hinweis: Dies ist eine erste mögliche Optimierung des Algorithmus'.

g) Passen Sie das Laufkriterium der inneren Schleife so an, dass sie nicht mehr immer bis zum letzten Element durchläuft, sondern nur den noch nicht sortierten Teil des Feldes berücksichtigt! [2]

Hinweis: Dies ist eine zweite mögliche Optimierung des Algorithmus'.

h) Definieren Sie in der "main"-Methode ein Feld (Array) mit den folgenden Ganzzahlen [2]:

`{34, 65, 43, -23, 8, 454, 34, 2, -9, 7, 6, 4, 12, 234, 54, 23, 76, 8, 98, 32}`

i) Sortieren Sie es durch Aufrufen der "sortBubble"-Methode! [2]

j) Geben Sie die Elemente des sortierten Feldes (Arrays) auf der Konsole aus! [2]

Ergebnis: Das gegebene Ganzzahlen-Feld (Array) wurde erfolgreich sortiert.

Task 5: XML Processing [40]

Zweck: Einlesen und Verarbeiten einer XML-Datei mittels Streaming API for XML (StAX).

Gegebene Datei: "repertoire.xml".

Hinweis zum Exception Handling: Werfen Sie einfach sämtliche Ausnahmen nach außen!

a) Erstellen Sie eine Klasse "Item" mit vier "String"-Attributen: year, artist, title, genre! [2]

b) Kapseln Sie die Klasse durch private Attribute und öffentliche Zugriffsmethoden! [2]

c) Implementieren Sie die “toString”-Methode, so dass alle Attribute ausgegeben werden! [2]

d) Erstellen Sie eine weitere Klasse namens “Parser”! Geben Sie ihr zwei Attribute: “item” vom Typ “Item” und “tagName” vom Typ “String”! [2]

e) Definieren Sie darin eine Methode “parseFile”, welche einen Dateinamen als Zeichenkette übergeben bekommt! Erzeugen Sie darin eine “ArrayList”, die Elemente des Typs “Item” speichert! Geben Sie sie am Ende der Methode als Rückgabewert zurück! [2]

f) Erzeugen Sie desweiteren eine “XMLInputFactory” sowie einen “FileInputStream”, wofür der übergebene Dateiname nötig ist! [2]

g) Verwenden Sie die obige Fabrik, um einen “XMLEventReader” zu erzeugen! [2]

h) Iterieren Sie mittels “while”-Schleife durch den “XMLEventReader”! Holen Sie sich das jeweils nächste “XMLEvent”! [2]

i) Übergeben Sie Event und Liste als Argumente an die aufzurufende Methode “processEvent”, welche zu implementieren ist! [2]

j) Prüfen Sie in der “processEvent”-Methode mittels der “is”-Methoden des “XMLEvent”-Parameters, ob es sich um ein Startelement, Endelement oder Zeichen handelt! Rufen Sie passend dazu je eine der folgenden Methoden auf: [2]

```
void processTagStart(StartElement e);
void processTagEnd(EndElement e, List<Item> l);
void processTagContent(Characters c);
```

k) Filtern Sie in der “processTagStart”-Methode nach den Tag-Namen: “item”, “artist”, “title”, “genre”! [2]

`Hinweis: startElement.getName().getLocalPart().equals()`

l) Erzeugen Sie ein neues “Item”-Objekt, wenn es sich um das “item”-Tag handelt! Weisen Sie das Objekt dem Instanzattribut “item” zu! [2]

m) Lesen Sie das XML-Attribut “year” aus und weisen Sie es dem Instanzattribut “item” zu! [2]

n) Weisen Sie in allen anderen Fällen dem Instanzattribut “tagName” die passende Zeichenkette “artist” oder “title” oder “genre” zu! [2]

o) Prüfen Sie in der “processTagEnd”-Methode, ob es sich um ein “item”-Tag handelt! Fügen Sie in diesem Falle das Instanzattribut “item” der als Parameter übergebenen Liste hinzu! [2]

p) Verwenden Sie in der “processTagContent”-Methode die Methode “isWhiteSpace” der “Characters”-Klasse, um überflüssige Leerzeichen herauszufiltern! [2]

q) Vergleichen Sie anschließend mittels Instanzattribut “tagName”, ob es sich beim aktuellen Element um “artist”, “title” oder “genre” handelt! Holen Sie sich dann mittels der “getData”-Methode den eigentlichen Textinhalt! Weisen Sie ihn via öffentliche Zuweisungsmethoden dem Instanzattribut “item” zu! [2]

r) Definieren Sie alle “String”-Literale als Konstanten der “Parser”-Klasse! [2]

s) Erzeugen Sie in der “main”-Methode der Startklasse eine “Parser”-Instanz! Lesen Sie damit die gegebene Datei “repertoire.xml” ein! Speichern Sie die Liste in einer lokalen Variablen! [2]

t) Geben Sie sämtliche Elemente der Liste per “for-each”-Schleife auf der Konsole aus! [2]

Ergebnis: Alle Musiktitel wurden auf der Konsole ausgegeben.

Viel Erfolg!