# 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 2015 |
| **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** | 2016-09-21 |
| **Duration [min]** | 120 |
| **Maximum Points [Point]** | 100 |
| **Permitted Study Aids** | Dokumentation im lokalen Netzwerkverzeichnis (Intranet);<br>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).<br>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.<br>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.<br>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.<br>Sie dürfen beliebig viele Bildschirmausgaben von Werten in den Quelltext einbauen, um ein Programm besser nachvollziehen zu können.<br>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 | 20 | 20 | 20 | 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 System [20]

Die Apache Open Source Organisation pflegt zahlreiche Projekte, unter Anderem die Java-Bibliothek "Commons IO", wobei "IO" für Input/Output steht. Ziel dieser Aufgabe ist die Benutzung eben jener externen Bibliothek, wofür es auch nötig ist, sich in das entsprechende Application Programming Interface (API) einzulesen.

a) Binden Sie die gegebene Java-Bibliothek namens "commons-io-2.4.jar" in ein neu zu erstellendes Projekt Ihrer Entwicklungsumgebung ein! Erstellen Sie eine Startklasse mit "main"-Methode! [4]

Hinweis: Als Dokumentation steht außerdem die Datei "commons-io-2.4-javadoc.jar" bereit, welche eingebunden werden oder ausgepackt und per Web Browser gelesen werden kann.

b) Nutzen Sie eine geeignete Methode der Klasse "FileUtils", um die gegebene Datei "gedicht.txt" zeilenweise einzulesen! [4]

c) Iterieren Sie unter Verwendung einer "for-each"-Schleife durch die Ergebnisliste! Bestimmen Sie die Länge einer jeden Zeile und geben Sie sie auf der Konsole aus! [4]

d) Definieren Sie eine Zeichenkette, die einen gültigen relativen Pfad auf die gegebene Datei "gedicht.txt" enthält, aber um ein nicht-existentes Unterverzeichnis gefolgt von zwei Punkten (als Verweis auf das übergeordnete Super-Verzeichnis) erweitert wurde, z. B. "filesystem/nonsense/../gedicht.txt"! Nutzen Sie nun eine geeignete Methode der Klasse "FilenameUtils", um die Pfadangabe von überflüssigen Einträgen zu befreien, so dass als Ergebnis die folgende Zeichenkette herauskommt: "filesystem/gedicht.txt"! [4]

e) Bestimmen Sie abschließend den freien Speicherplatz der aktuellen Partition der Festplatte! Nutzen Sie dazu eine geeignete Methode der Klasse "FileSystemUtils"! [4]

**Task 2: Container [20]**

Mehrere Studenten sollen in einer Liste vorgehalten werden, um später bearbeitet und ausgegeben werden zu können.

a) Erstellen Sie eine Klasse namens "Student" mit drei Attributen jeweils passenden Types zum Erfassen von Name, Fachrichtung und Matrikelnummer! [2]

b) Ergänzen Sie die Zugriffsmethoden und stellen Sie einen passenden Konstruktor zur Initialisierung bereit! [2]

c) Initialisieren Sie in einer Startklasse drei Objekte vom Typ "Student"! [2]

- Student 1: Britta, BWL, 4711
- Student 2: Max, Informatik, 4712
- Student 3: Tom, Biologie, 4713

d) Erstellen Sie eine "ArrayList" auf typisierte Weise und fügen Sie ihr die ersten ZWEI erzeugten Studentenobjekte hinzu! [2]

e) Implementieren Sie eine statische Methode namens "printList" für die Ausgabe aller in einer ArrayList enthaltenen Studenten mitsamt Eigenschaften auf der Konsole! (Vom Prinzip her vergleichbar einer "toString"-Methode.) Verwenden Sie dafür eine "for-each"-Schleife! [2]

f) Implementieren Sie eine weitere statische Methode namens "addNewStudent", welche vor dem Hinzufügen prüft, ob das an sie als Argument übergebene "Student"-Objekt in der ebenfalls als Argument übergebenen Liste bereits existiert! [2]

Hinweis: Die Klasse "ArrayList" bietet eine geeignete Methode.

g) Rufen Sie die Methode "addNewStudent" mit dem ersten und anschließend mit dem dritten Studenten als Argument auf! [2]

h) Prüfen Sie die in der Liste enthaltenen Objekte via Methode "printList" vor und nach dem Hinzufügen der neuen Objekte! Hinweis: Bei Bedarf können Sie gerne zusätzlich die Container-Größe vorher und nachher bestimmen. [2]

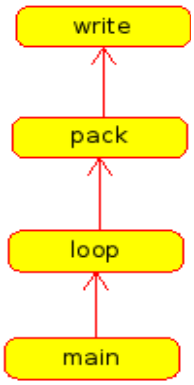
i) Löschen Sie via Listenindex den ersten Eintrag! Unternehmen Sie einen weiteren Löschversuch, indem Sie das zuerst erstellte "Student"-Objekt (Britta) als Argument an die Löschmethode übergeben! Prüfen Sie den Rückgabewert! [2]

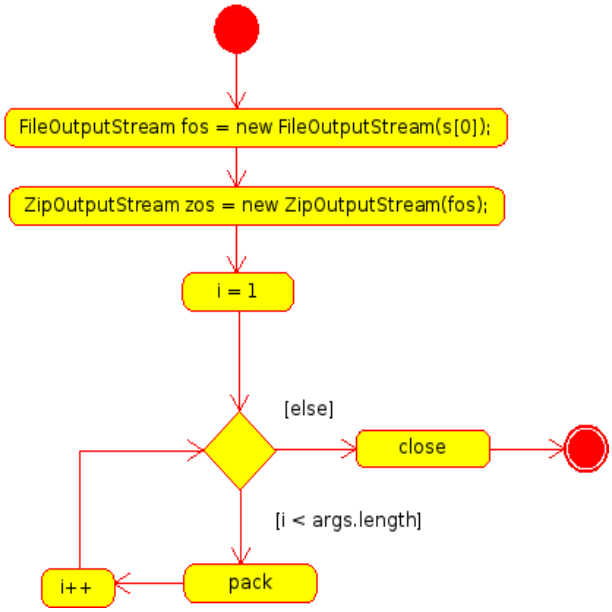
j) Entfernen Sie schließlich durch einen einzigen Methodenaufruf alle Studenten aus der Liste! Überprüfen Sie den Erfolg dieser Aktion durch Testen des Containers auf Leere! [2]

**Task 3: Stream [20]**

Hier soll der Umgang mit "Stream"-Klassen geprüft werden. Drei gegebene Dateien sind per ZIP-Algorithmus in eine komprimierte Ausgabedatei zu packen.

Zur Veranschaulichung seien dazu zwei Programmablaufpläne gegeben. Der linke zeigt die Abhängigkeiten bzw. Aufrufreihenfolge zwischen zu implementierenden Methoden; der rechte beispielhaft den Inhalt der "loop"-Methode.




a) Erstellen Sie eine Anwendung mit vier Klassenmethoden namens "main", "loop", "pack" und "write"! Ihre Argumente ergeben sich aus der weiteren Aufgabenstellung. Einen Rückgabewert hat keine von ihnen. [2]

b) Die Anwendung soll als Kommandozeilen-Argumente zunächst den Namen des zu erstellenden Archivs und nachfolgend die Namen der zu packenden Dateien entgegennehmen:

```
java Launcher zipfile files ...
```

Prüfen Sie in der "main"-Methode die Anzahl der Kommandozeilen-Argumente! Geben Sie im Fehlerfalle die durch den Nutzer zu verwendende Kommandozeile (siehe oben) als Hilfe aus! [2]

c) In der als nächstes aufgerufenen Methode "loop" sind zwei "Stream"-Objekte zu erstellen: eines für den schreibenden Dateizugriff und ein weiteres als Hülle, welches sich um das Packen per Zip-Algorithmus kümmert! [2]

d) Durchlaufen Sie die als Argument an die "loop"-Methode übergebenen Dateinamen per Zählschleife! Rufen Sie für jeden Eintrag als nächstes die "pack"-Methode auf, wobei ihr als erstes Argument der Ausgabedatenstrom und als zweites Argument der aktuelle Dateiname zu übergeben sind! [2]

e) Schließen Sie den oder die Datenströme anschließend! [2]

f) Erstellen Sie in der "pack"-Methode ein "ZipEntry"-Objekt auf Basis des mitgegebenen Dateinamens! Fügen Sie es dem Ausgabedatenstrom via Methodenaufruf hinzu! [2]

g) Rufen Sie aus der "pack"-Methode heraus schließlich die "write"-Methode auf und übergeben ihr die gleichen Parameter! [2]

h) Nutzen Sie ein "FileInputStream"-Objekt zum Lesen der als Argument gegebenen Datei! Alloziieren Sie zum Zwischenspeichern der eingelesenen Daten außerdem ein Byte-Feld (Array) der Größe 4096 als Puffer! [2]

i) Lesen Sie die Daten via "while"-Schleife ein und speichern Sie sie im Puffer! [2]

Hinweis: Auch der Eingabedatenstrom ist zu schließen, aber erst nach der Schleife.

j) Schreiben Sie die gepufferten Daten über einen passenden Methodenaufruf in den Ausgabedatenstrom! [2]

Hinweis: Das erzeugte Archiv kann zum Testen aus Eclipse heraus oder mit Anwendungen wie z. B. "KDE Ark", "jar", "winzip" oder "pkunzip" ausgepackt werden.

**Task 4: Formatting [20]**

In dieser Aufgabe sollen Daten lokalisiert und formatiert ausgegeben werden.

a) Schreiben Sie ein Programm, das als Kommandozeilen-Argument die Gleitkommazahl 1234567,89 übergeben bekommt! Konvertieren und speichern Sie sie in einer lokalen Variablen des "Wrapper"-Types "Double"! [2]

b) Erstellen Sie eine statische Methode namens "displayCurrency", welche die Gleitkommazahl sowie ein Objekt vom Typ "Locale" übergeben bekommt und keinen Rückgabewert besitzt! [2]

c) Instanziieren Sie über eine passende Klassenmethode einer geeigneten Klasse ein Formatierungsobjekt! [2]

d) Berücksichtigen Sie dabei das landesspezifische Argument! [2]

e) Geben Sie den übergebenen Zahlenwert schließlich als Währungsbetrag mit dem entsprechenden Währungssymbol auf der Konsole aus! [2]

f) Erstellen Sie eine weitere statische Methode namens "displayDate", die ebenfalls ein Lokalisierungsargument entgegen nimmt! [2]

g) Instanziieren Sie auch hierfür über eine passende Klassenmethode einer geeigneten Klasse ein Formatierungsobjekt! [2]

h) Lassen Sie das Datum vollständig und in landestypischer Sprache / Formatierung stehen! [2]

i) Bestimmen Sie das tagesaktuelle Datum und geben Sie es auf der Konsole aus! [2]

j) Rufen Sie beide Methoden jeweils für Deutschland, die Vereinigten Staaten von Amerika und Frankreich auf! [2]

**Task 5: Reflexion [20]**

Im Rahmen dieser Aufgabe wird der Umgang mit Mitteln der Meta-Programmierung geprüft.

a) Initialisieren Sie eine lokale Variable mit einem "java.awt.Point"-Objekt! Bestimmen Sie Ihren Typ (bzw. Klassenobjekt) auf dreierlei Weise: per Schlüsselwort an Klasse, per Aufruf einer Instanzmethode und per Klassenmethode! Geben Sie die Ergebnisse jeweils auf der Konsole aus! [5]

b) Finden Sie über eine Methode eines der drei oben bestimmten Typen (bzw. Klassen) heraus, ob es sich um eine Schnittstelle handelt! Bestimmen Sie die Superklasse sowie sämtliche Schnittstellen! Berichten Sie jeweils auf der Konsole! [5]

c) Instanziieren Sie einen der drei oben bestimmten Typen (bzw. Klassen) durch Aufrufen einer passenden Methode und geben Sie die x- und y-Werte des Punktes auf der Konsole aus! Wiederholen Sie das Prozedere unter Vermeidung einer Typumwandlung (Downcast) auf "Point"! Instanziieren Sie ein drittes Mal, nun allerdings über den Umweg der Bestimmung des Konstruktors aus dem Typen (bzw. der Klasse)! [5]

d) Holen Sie sich über eines der eben erzeugten Objekte dessen "x"-Eigenschaft (bzw. Attribut/ Feld)! Manipulieren Sie den Wert dieser Variablen beliebig! Geben Sie ihn zur Kontrolle aus! [5]

**Viel Erfolg!**