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 2017-1
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	2018-06-
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]	10	20	30	30	10	100

Im Rahmen der Klausur sind fünf verschiedene Aufgaben zu lösen.

Task 1: File System [10]

Zweck: Arbeiten mit dem Dateisystem.

a) Erstellen Sie eine Java-Anwendung, in welcher auf rekursive Weise ein Verzeichnis namens "blu/bla/testdir/" erstellt wird! [2]

b) Überprüfen Sie die Existenz des Verzeichnisses und geben Sie eine Fehler-Nachricht auf der Konsole aus, falls es nicht existiert! [2]

c) Geben Sie den absoluten Pfadnamen des Verzeichnisses aus! [2]

d) Erstellen Sie in dem neuen Verzeichnis eine leere Textdatei namens "test.txt"! [2]

e) Erzeugen Sie eine weitere Dateireferenz mit *neuem* Namen! Ändern Sie den Namen der ursprünglich erstellten, leeren Textdatei “test.txt” auf den neuen Namen! [2]

Ergebnis: Das Verzeichnis und die Datei können im Datei-Manager überprüft werden.

Task 2: Collection Framework [20]

Zweck: Ein Behälter (Container) soll manipuliert werden.

a) Erstellen Sie eine Hauptklasse beliebigen Namens, in welcher die zwei Nachrichten "Startup" und "Shutdown" auf der Konsole ausgegeben werden! [2]

b) Ergänzen Sie zwischen beiden Nachrichten den Quelltext zum Erzeugen eines Containers vom Typ "Hashtable"! Weisen Sie die Instanz einer lokalen Variablen vom Typ "Map" zu! [2]

c) Passen Sie den Konstruktor-Aufruf des oben erzeugten Hashtables so an, dass eine typsichere Klasse angenommen wird (Schlüssel vom Typ "String"; Wert vom Typ "Double")! [2]

d) Füllen Sie den Container per Zufallszahlen-Generator mit zehn beliebigen Zahlen-Werten! Verwenden Sie dazu eine Schleife! [2]

e) Benennen Sie die Schlüssel dabei nach dem Muster "entry0", "entry1" etc.! [2]

f) Lassen Sie sich die Schlüssel des Hashtables als Set zurückgeben! [2]

g) Durchsuchen Sie die Schlüssel unter Verwendung des passenden Iterators! [2]

h) Geben Sie die Nachricht "Found entry 7." auf der Konsole aus, wenn der Schlüssel "entry7" gefunden wurde! [2]

i) Entfernen Sie den Eintrag mit dem Schlüssel "entry8" aus dem Hashtable! [2]

j) Geben Sie die Größe sowie den Inhalt der Map auf der Konsole aus! (Schleife nicht nötig!) [2]

Ergebnis: Die Map wurde befüllt und bearbeitet sowie ihr Inhalt ausgegeben.

Task 3: Stream [30]

Zweck: Einlesen von Text aus einer Datei. Ausgeben auf Konsole. Schreiben in eine Datei.

a) Erstellen Sie auf Dateisystem-Ebene eine Datei namens "in.txt" mit folgendem Inhalt [2]:

```
Once upon a time, there were two little robots.
They flew through space, from planet to planet.
Then they got old and tired and stopped flying around.
```

b) Erzeugen Sie ein geeignetes “Stream”-Objekt zum Einlesen von Zeichen aus der oben erstellten Datei! Hinweis: Beachten Sie die korrekte Schreibung von Pfadnamen! [2]

c) Kapseln Sie es in einem gepufferten Eingabedatenstrom! [2]

d) Lesen Sie eine Zeile aus der Datei ein! [2]

e) Verwenden Sie eine “while”-Endlosschleife, um sämtliche Zeilen einzulesen! [2]

f) Bauen Sie ein sinnvolles Abbruchkriterium in den Schleifenkörper ein! [2]

g) Fügen Sie der Schleife eine ganzzahlige Zählvariable hinzu, die in jedem Durchlauf um eins inkrementiert wird! [2]

h) Geben Sie in jedem Durchlauf den jeweils eingelesenen Text sowie die zugehörige Zeilennummer (Zählvariable) auf der Konsole aus! [2]

i) Erzeugen Sie ein geeignetes “Stream”-Objekt zum Ausgeben von Zeichen in eine Datei namens “out.txt”! [2]

j) Kapseln Sie es in einem gepufferten Ausgabedatenstrom! [2]

k) Schreiben Sie die eingelesenen Zeilen nun zusätzlich zur Ausgabe auf der Konsole zugleich in die Ausgabedatei! [2]

l) Fügen Sie hinter jeder Zeile unter Verwendung einer Methode des gepufferten Ausgabedatenstromes einen Zeilenumbruch ein! [2]

m) Fügen Sie der Ausgabedatei am Ende folgenden Text inklusive Zeilenumbruch hinzu [2]:

```
And they all lived happily ever after.
```

n) Vergessen Sie nicht, geöffnete Datenströme zu schließen! [2]

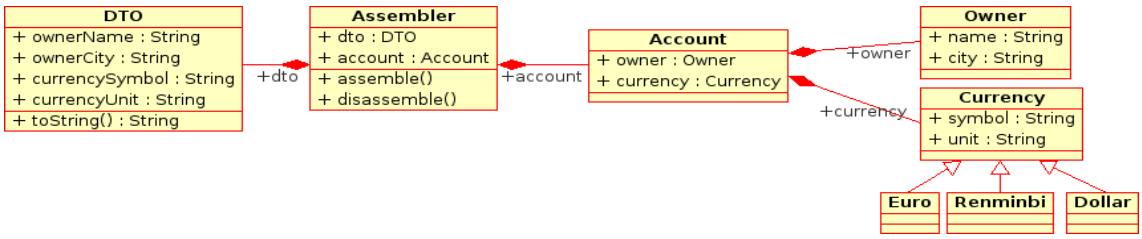
o) Fangen Sie auftretende Ausnahmen ab! [2]

Ergebnis: Der Dateiinhalt ist in einem Datei-Manager überprüfbar.

Task 4: Mapping [30]

Zweck: Für die Datenübertragung in einem Netzwerk ist es oft hilfreich, einzelne Daten eines komplexen Domänenmodells in eine flache Datenstruktur zu überführen, welche auch bekannt ist unter dem Namen “Data Transfer Object” (DTO).

Gegeben sei dazu ein Domänenmodell namens “Account” mit zugehörigen Klassen “Owner”, “Currency”, “Euro”, “Renminbi”, “Dollar”. Durch einen “Assembler” sollen dessen Daten in das DTO geschrieben bzw. aus ihm gelesen werden. Siehe auch folgendes UML Klassendiagramm:



a) Erstellen Sie die Klassen gemäß oben gegebenem Diagramm! [2]

b) Berücksichtigen Sie Vererbungsbeziehungen! [2]

c) Geben Sie den Klassen die zugehörigen Attribute! [2]

d) Weisen Sie ihnen die korrekten Typen zu! [2]

e) Fügen Sie die Methoden ein! [2]

f) Füllen Sie den Rumpf der "assemble"-Methode der "Assembler"-Klasse mit den für die Übersetzung notwendigen Zuweisungen! Hinweis: Holen der Werte aus dem Domänenmodell per “get”-Methoden und zuweisen zum DTO per “set”-Methoden. [2]

g) Tun Sie Gleiches in umgekehrter Weise innerhalb der “disassemble”-Methode! [2]

h) Ergänzen Sie die Klassen "Account", "Owner" und "Currency" um Standard-Konstruktoren, auch wenn diese im Diagramm nicht gezeigt sind! [2]

i) Implementieren Sie die Standard-Konstruktoren so, dass alle Attribute der jeweiligen Klasse automatisch erzeugt werden, so dass kein Attribut den Wert "null" besitzt! [2]

j) Verwenden Sie dabei standardmäßig die "Euro"-Währung für das "currency"-Attribut der "Account"-Klasse! [2]

k) Schreiben Sie eine "toString"-Methode für die "DTO"-Klasse, welche die von "java.lang.Object" geerbte "toString"-Methode überschreibt! Machen Sie dies durch eine Annotation kenntlich! [2]

l) Erstellen Sie schließlich eine Hauptklasse namens "Launcher.java" mit "main"-Methode! Erzeugen Sie darin je eine Instanz der Klassen "Account", "DTO" und "Assembler"! [2]

m) Weisen Sie die Instanzen von "Account" und "DTO" den entsprechenden Attributen der "Assembler"-Instanz zu! [2]

n) Rufen Sie der "Assembler"-Instanz' "assemble"-Methode auf! [2]

o) Fügen Sie vor und nach dem Aufruf der "assemble"-Methode eine Zeile ein, in welcher die "DTO"-Instanz auf der Konsole ausgegeben wird! [2]

Ergebnis: Aus dem Domänenmodell wurde erfolgreich ein DTO gebastelt.

Task 5: Reflexion [10]

Zweck: Eine Methode soll dynamisch, d. h. zur Laufzeit, mittels Reflexion aufgerufen werden.

a) Erstellen Sie eine Klasse namens "Test" mit zwei Instanz-Methoden, die jeweils eine beliebige Zeichenkette zurückliefern! Erstellen Sie auch die übliche Startklasse mit "main"-Methode! [2]

b) Bestimmen Sie in der “main”-Methode mittels "forName"-Methode der Klasse "Class" das Klassenobjekt der "Test"-Klasse! Hinweis: Beachten Sie evtl. anzugebende Paketnamen! [2]

c) Instanziieren Sie die "Test"-Klasse! Nutzen Sie dazu NICHT den "new"-Operator, sondern eine Instanz-Methode des Klassenobjektes! [2]

d) Bestimmen Sie anschließend, wiederum über das Klassenobjekt, eine der beiden Methoden der "Test"-Klasse! Speichern Sie die entsprechende Referenz in einer lokalen Variablen des Types "Method"! [2]

e) Rufen Sie diese Methode am zuvor erzeugten Objekt des Types "Test" auf! [2]

Ergebnis: Die zurückgelieferte Zeichenkette wurde auf der Konsole ausgegeben.

Viel Erfolg!