Übung 06: Streams

Abgabetermin: 27. 4. 2017, 8:15

Name: _____ **Matrikelnummer**: _____

Informatik: ☐ G1 (Marr) ☐ G2 (Prähofer) ☐ G3 (Prähofer) ☐ G4 (Löberbauer)

WIN: ☐ G1 (Khalil) ☐ G2 (Hummel) ☐ G3 (Khalil)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 6	24	Java-Programm, Ausgabe des Testlaufs	Java-Programm	☐	

Übung 06: Auswertung der Übungsergebnisse

Das Auswerten der Ergebnisse aus SW2 Übung ist kompliziert, aufwendig und fehleranfällig. Ihr Programm soll die Auswertung und die Notenvergabe ermöglichen. Dazu verwenden Sie bitte die Möglichkeiten von Streams.

Input für Ihr Programm ist eine CSV-Datei mit den Resultaten, in der jede Zeile folgendes Format hat:

```
Nr;Name;Vorname;MatrNr;SKZ;1;2;3;4;5;6;7;8;9;10;11;1;2;3;4;5;6;7;8;9;10;11;Test
```

Das heißt, jede Zeile hat folgende Spalten

- eine Nummer
- den Familiennamen
- den Vornamen
- die Matrikelnummer
- die Studienkennzahl
- 11 Einträge für die Abgaben mit den Werten 0 oder 1
- 11 Einträge mit den erreichten Punkten; eventuell sind Spalten **leer**
- die beim Test erreichten Punkte; wenn kein Test geschrieben, dann **leer**

Die Spalten sind mit Strichpunkten getrennt. Die erste Zeile in der Datei ist eine Überschriftszeile und ist zu überspringen. Im Download finden Sie eine Datei `data.csv` mit Testdaten.

Im Download finden Sie auch eine Reihe von Datenklassen, die für die zu ermittelnden Resultate verwendet werden sollen. Dies sind:

- `Data` für die Kodierung der Rohdaten aus dem File,
- `Result` für die Detailresultate,
- `Grading` für die Benotung und
- die Enumerationsklasse `Grade` für die Noten.

Ihr Programm soll nun nacheinander die Resultate wie unten beschrieben berechnen und dabei die Funktionen von Streams verwenden (bei jeder Aufgabe ist angemerkt, wie man das machen kann).

Jedes Resultat ist entsprechend auf der Konsole auszugeben.

Anmerkung: Die Aufgabe ist vollständig ohne Verwendung von while- und for-Schleifen und nur unter Verwendung von Stream-Operationen zu lösen. Bei Verwendung von while und for wird der entsprechende Teil als NICHT gelöst gewertet. Für die Ausgaben auf die Konsole können Sie Schleifen verwenden, man kann aber auch die Ausgaben nur mit Streams implementieren.

1. Lesen und Kodieren der Rohdaten mit Ergebnis (`List<Data>`)

Der File soll mit der Methode `lines` der Klasse `Files` aus Package `java.nio` von einer Datei zeilenweise eingelesen werden:

```
static Stream<String> lines(Path path) throws IOException
```

Zum Beispiel kann man den Text einer Datei mit Namen `"data.csv"` wie folgt lesen:

```
java.nio.Files.lines(java.nio.Paths.get("data.csv"))
```

Dabei liefert `lines` einen Stream, mit dem man den File zeilenweise verarbeiten kann. Verwenden Sie diesen Stream, um für jede Zeile ein `Data`-Objekt zu erzeugen und dann diese in eine Liste `List<Data>` zusammenzufassen. Gehen Sie dabei folgend vor:

- Zerlegen Sie die Zeile mit `String.split(";")` in ein Array `String[]`.
- Erzeugen Sie aus den Daten im Array ein `Data`-Objekt, indem Sie aus den Elementen im Array die entsprechenden Daten extrahieren und kodieren. Für die Einträge mit den 11 Abgaben und 11 Punkten sollen `List<Integer>` bzw. `List<Double>` erzeugt werden (siehe Hinweise).
- Sortieren Sie die Daten nach dem Namen.

Hinweise:

- Mit `<T> Stream<T> stream(T[] array, int startInclusive, int endExclusive)` kann man für einen Teilbereich eines Arrays einen Stream erzeugen. Dies kann man gut für die Erzeugung der Listen für die Abgaben und erreichten Punkte verwenden.
- Mit `Integer.parseInt(String s)` bzw. `Double.parseDouble(String s)` kann man aus den Strings Integer- bzw. Double-Werte parsen.
- Beachten Sie, dass bei den Punkten die Einträge leer sein können. Diese filtert man einfach vor dem Parsen raus. Auch der Test kann leer sein und dies ist entsprechend zu behandeln.

2. Liste mit Resultaten (Ergebnis `List<Result>`)

Aus der Liste der `Data`-Objekte soll nun eine Liste von `Result`-Objekten mit Matrikelnummer, Studienkennzahl, Anzahl der Abgaben, Punktedurchschnitt und Testergebnis erzeugt werden.

Hinweise:

- Verwenden Sie `sum()` von `IntStream`, um aus der Liste mit den Abgaben die Anzahl der abgegeben Übungen zu ermitteln.
- Verwenden Sie `collect(Collectors. averagingDouble())` für den Durchschnitt der Punkte.

3. Liste der Benotungen (Ergebnis `List<Grading>`)

Erzeugen Sie aus der Liste der Resultate eine Liste mit Benotungen (`Grading`) wie folgt:

- Filtern Sie zuerst die Liste der `Result`s so, dass die Einträge, die nicht benotet werden, entfernt sind. Keine Benotung bekommen Teilnehmer, die weniger als 9 Übungen abgegeben oder die keinen Test geschrieben haben. Für die Restlichen wird eine Benotung wie folgt erstellt:
- Wurde ein Test geschrieben und hat man weniger als 12 Punkte erreicht, ist die Note `NotPassed` (*Nicht genügend*).
- Sonst wird das Mittel vom Durchschnitt der erreichten Punkte und dem Test berechnet und daraus die Noten folgend ermittelt. Diese wird noch mit Anzahl der Abgaben / 11 Übungen gewichtet. Die Note ergibt sich dann wie folgt:

  < 12 : NotPassed, < 15 Sufficient, < 18 Satisfactory, < 21 Good, >= 21 Excellent

4. Statistik der Benotungen (Ergebnis `Map<Grade, List<Grading>>`)

Gruppieren Sie die Benotungen nach der Note. Geben Sie damit eine Statistik mit Note und Anzahl der Noten aus.

Hinweise:

- Verwenden Sie `collect(Collectors.groupingBy(…))`

5. Generierung einer CSV-Datei mit den Benotungen (Ergebnis ein `String` mit mehreren CSV-Zeilen)

Generieren einen Text für eine CSV-Datei mit

- Matrikelnummer
- Studienkennzahl
- entsprechende Notentext `"1"`, `"2"`, `"3"`, `"4"`, oder `"5"`

jeweils mit `";"` getrennt.

Schreiben Sie diesen Text in eine Ausgabedatei `grades.csv`.

Hinweise:

- Verwenden Sie `String.format`, um eine einzelne Zeile zu erstellen, und `collect(Collectors.joining("\n"))`, um die einzelnen Zeilen zum Gesamttext zusammenzufügen.
- Mit `Out.open(filename)` kann man eine Datei öffnen, dann mit `Out.print` schreiben und mit `Out.close()` wieder schließen.

Anmerkung: Eine Datei mit diesem Format kann man im KUSSS importieren!

Beispielausgabe:

Die Ergebnisausgabe sollte in etwa folgend aussehen:

```
1. List of Data objects
-----------------------

Data [name=Berger, firstName=Anna, stdtId=1356333, skz=521, submitted=[1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 0], points=[17.0, 23.0, 22.0, 21.0, 17.0, 17.0, 18.0, 19.0, 20.0, 21.0], test=22.0]
Data [name=Fuereder, firstName=Herbert, stdtId=1234567, skz=521, submitted=[1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0], points=[19.0, 23.0, 22.0, 21.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0], test=9.0]
…


2. List of detailed results
---------------------------
Result [name=Berger, firstName=Anna, stdtId=1356333, skz=521, submitted=9, avrgPoints=19.5,
test=22.0]
Result [name=Fuereder, firstName=Herbert, stdtId=1234567, skz=521, submitted=9, avrgPoints=19.6,
test=9.0]

3. Gradings
------------
Grading [stdtId=1356333, skz=521, grade=Satisfying]
Grading [stdtId=1234567, skz=521, grade=NotPassed]
…

4.Statistics
------------
Grade 1: 1
Grade 2: 3
Grade 3: 2
Grade 4: 1
Grade 5: 1

5. CSV File
-----------
1356333;521;3
1234567;521;5
1455765;521;2
…
```