HSLU Hochschule Luzern

Objektorientierte Programmierung (OOP)

Übung SW12: Input/Output-Datenströme

Themen: Byte- und Zeichenströme, Lesen und Schreiben in Dateien (O13).
Zeitbedarf: ca. 180min.

Roland Gisler, Version 1.6.5 (FS25)

1 Byte-Datenströme

1.1 Lernziele

- Sie verstehen den Unterschied zwischen Byte- und Zeichenströmen.
- Sie können die in Java vorhandenen I/O-Streamklassen anwenden.
- Strukturiertes Lesen und Schreiben von Daten in Dateien.

1.2 Grundlagen

Als Grundlage dienen die Codebespiele aus dem Input `O13_IP_Datenströme`.

1.3 Aufgaben

a.) Studieren Sie die Codebeispiele aus dem Input `O13_IP_Datenströme` für die Klassen `DataInputStream` und `DataOutputStream`. Werfen Sie auch einen Blick in die JavaDoc.

b.) Verwenden Sie einen Byte-Datenstrom und schreiben Sie mit Hilfe von `DataOutputStream` einen einfachen `int`-Wert in eine Datei. Öffnen Sie die Datei mit einem Editor und schauen Sie sich den Inhalt an.

c.) Lesen Sie den Inhalt der Datei von b) mit Hilfe von `DataInputStream` wieder ein. Was passiert, wenn Sie versuchen, den int-Wert als einen anderen Typ einzulesen (z.B. `byte` oder `float`)?

d.) Sie können verschiedenste elementare Datentypen (`float`, `double`, `char` etc.) in beliebiger Reihenfolge in einen Byte-Datenstrom (und somit auch in eine Datei) schreiben. Was aber gewährleistet sein muss ist, dass der Empfänger beim Einlesen der Daten die Typen in **exakt derselben Reihenfolge** einliest. Probieren Sie es aus!

e.) Sie können auch dynamische Binärformate entwerfen: Nehmen Sie dazu das Programm der letzten Woche (Eingabe von Temperaturwerten von der Konsole) als Basis. Nach Abschluss der Eingabe schreiben Sie alle Werte in eine Binärdatei. Die Idee dabei: Schreiben Sie als erstes die Anzahl der Werte als `int`, und danach die entsprechenden Temperaturwerte (`double` oder `float`) der Reihe nach in die Datei!

f.) Lesen Sie die Daten von e) nun auch wieder ein. Zuerst kommt also die Anzahl der Werte als `int`-Wert, und danach können Sie mit einer Iteration die dynamische Anzahl von Temperaturwerten einlesen. Klappt es? Sie haben soeben Ihr erstes eigenes Binärformat implementiert.

2 Einlesen von Temperaturwerten aus einer `csv`-Datei

2.1 Lernziele

- Implementation einer einfachen Anwendung zum Einlesen von Temperaturwerten.
- Parsen von textbasierten Daten aus einer Textdatei.
- Objektorientiertes Design und Refactoring.

2.2 Grundlagen

Als Grundlage dient der aktuelle Entwicklungsstand der Klassen `Temperatur` und `TemperaturVerlauf` von letzter Woche.

Vom Dozent wird Ihnen auf dem ILIAS eine Datei `netatmo-export-202301-202304.csv` zur Verfügung gestellt, welche Messdaten einer Wetterstation enthält. Diese sind in einem **csv**-Format (**c**haracter-**s**eparated **v**alues) gespeichert.

WICHTIG: Wenn Sie sich die Daten einfach mal ansehen wollen (was eine durchaus vernünftige Idee ist), verwenden Sie dafür einen einfachen Test-Viewer oder Editor (Notepad etc.), aber **nicht** Excel. Dank dieses Hinweises werden Sie es wohl trotzdem ausprobieren, aber dank kritisch-vorgewarntem Blick einen wirklich «fiesen» Effekt feststellen.

Konkret sind die einzelnen Datenwerte mit Semikolons (`;`) getrennt, und Strings in Anführungszeichen (`"`) eingefasst. Hier ein Ausschnitt, in jeder Zeile ist je ein einzelner Datenwert **fett** markiert:

```
1473517749;"2023/01/10 16:29:09";30.4;60
1473518057;"2023/01/10 16:34:17";30.9;59
1473518364;"2023/01/10 16:39:24";31.5;58
1473518672;"2023/01/10 16:44:32";32.1;57
```

Der erste Wert hat (für uns) keine sinnvolle Bedeutung. Der zweite Wert ist ein Zeitpunkt, bestehend aus Datum und Zeit (ein sogenannter `Timestamp`). Der dritte Werte ist die Temperatur (mit maximal einer Dezimalstelle) und der vierte Wert ist die Luftfeuchtigkeit in (ganzen) Prozent.

2.3 Aufgaben

a.) Kopieren Sie die Datei an einen von Ihnen gewählten Ort und prüfen Sie dann mit einem kleinen Java-Programm (oder Unit-Test!) ob Sie mittels `File.exists(…)` Zugriff auf die Datei haben und Ihre Pfadangabe somit korrekt ist.

Tipp: Denken Sie (je nach Betriebssystem) an die Escapezeichen bei den Pfadangaben!

b.) Nehmen Sie den Input zu Hilfe und lesen Sie die Datei in einem ersten Entwicklungsschritt ganz einfach zeilenweise mit `readline()` ein. Geben Sie die den Inhalt gleich wieder auf der Konsole (oder per Logging) aus – ohne weitere Verarbeitung!

Tipp: Da wir hier (zum Glück) keine Encoding-Probleme zu erwarten haben (alle enthaltenen Zeichen gehören zum ASCII-7bit-Standard), können Sie zur Vereinfachung eine Verkettung von `BufferedReader` mit `FileReader` verwenden. Aber es gibt eigentlich keinen wirklich guten Grund für diese Vereinfachung!

c.) Extrahieren Sie nun aus den per Semikolon unterteilten Zeilen als erstes den Temperaturwert und geben Sie diesen ebenfalls gleich wieder auf der Konsole aus. Durch dieses Schrittweise vorgehen finden Sie allfällige Fehler sehr viel schneller!

Tipp: Verwenden Sie die Methode `split(…)` der Klasse `String` um die einzelnen Werte zu extrahieren, und `Double.valueOf(…)` um die Temperatur zu einer `double` zu konvertieren (für andere Datentypen wie `Float` und `Integer` analog).

d.) Als letztes wollen wir auch das Datum und die Zeit (bilden zusammen einen sogenannten **`Timestamp`**) auslesen. Leider ist dieser nicht in einem Standardformat gespeichert, wohl aber in einem für uns gut interpretierbaren Format. Sie können den Wert mit Hilfe des folgenden Statements in ein **`DateTime`** einlesen:

```
LocalDateTime timestamp =
        LocalDateTime.parse(timestampAsString,
        DateTimeFormatter.ofPattern("\"yyyy/MM/dd HH:mm:ss\""));
```

Hinweis: Seit Java 8 kennt Java eine neue DateTime API. Verwenden Sie also **nicht** mehr die veraltete Klasse **`java.util.Date`**, sondern die Klassen aus dem neuen Package **`java.datetime.*`**.

e.) Wenn Sie bei der Entwicklung professionell vorgegangen sind, haben Sie diese Konvertierungen in eine eigene Klasse ausgelagert, die weder weiss, woher die Daten (der **`String`**) kommen, noch wohin (in **`TemperaturVerlauf`**) sie gehen! Der Vorteil dieses SRP-Designs ist, dass sich diese Klasse extrem gut und einfach (Unit-)testen lässt!

f.) Damit haben wir alles zusammen, damit «der Spass» beginnen kann: Erweitern Sie das Programm, so dass Sie folgende Fragen beantworten können:

- Was war im gesamten Zeitraum die tiefste und die höchste Temperatur?
- Was war die Durchschnittstemperatur?
- Wann wurde die höchste Temperatur erreicht? Wann die tiefste?

Vergessen Sie nicht, dafür Ihre bereits vorhandenen Klassen aus allen früheren Übungen zu verwenden!

g.) Für einige der Fragenstellungen bietet sich bereits wieder ein Refactoring Ihrer vorhandenen Lösung an. Haben Sie beispielsweise daran gedacht, dass es sinnvoll sein könnte eine neue Klasse, z.B. mit dem Namen «**`Messpunkt`**» zu entwerfen, welche mindestens den **`Timestamp`** und die **`Temperatur`** zusammenfasst? Welchen Einfluss hätte diese Klasse auf **`TemperaturVerlauf`**?

h.) Setzen Sie Ihre Überlegungen praktisch um: Viel Spass beim Programmieren!

3 Optionale Repetitionsaufgabe: Raumverwaltung – Teil 3 von 5

3.1 Lernziele

- Repetition bereit behandelter Themen zur Festigung und Vertiefung.

3.2 Grundlagen

Diese Zusatzaufgaben sind unabhängig von den restlichen Aufgaben dieser Woche. Sie baut aber auf der optionalen Repetitionsaufgabe der letzten Woche auf.

Optionale Aufgaben sind **nicht** Bestandteil der Besprechungen.

3.3 Aufgaben

a.) Testen Sie die `equals()`-Implementation auf der Klasse `Raum` mit **EqualsVerifier**.

b.) Fügen Sie der Klasse `Raum` eine öffentliche Methode hinzu, welche als `boolean` zurückliefert, ob ein Raum frei ist. Testen Sie diese Methode mit Unit-Tests.

c.) Implementieren Sie auf der Klasse `RaumVerwaltung` eine Methode, über welche für eine bestimmte Anzahl Personen automatisch ein passender Raum ausgewählt und reserviert wird. Als Rückgabetyp verwenden Sie die Klasse `Raum`. In einem ersten Schritt darf (zur Vereinfachung!) der erste genügend grosse (und natürlich freie) Raum gewählt werden.

d.) Ergänzen Sie auf der Klasse `RaumVerwaltung` eine Methode, mit welcher ein reservierter Raum wieder freigegeben wird. Wählen Sie einen geeigneten Rückgabetyp, um den Erfolg der Operation mitteilen zu können.