Übungsblatt 08

3 Aufgaben, 20 Punkte

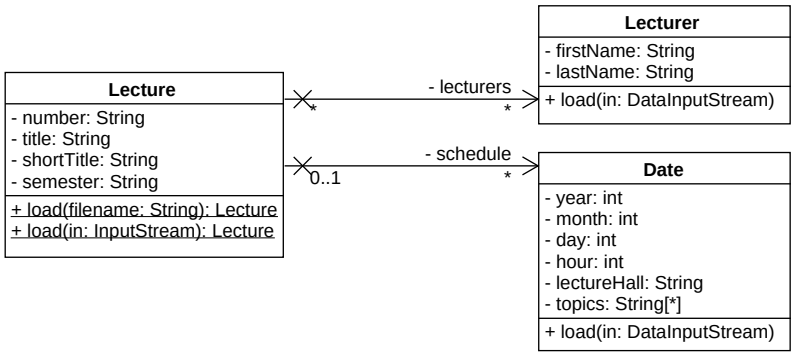
Objektorientierte Modellierung und Programmierung (inf031) Sommersemester 2020
Carl von Ossietzky Universität Oldenburg, Fakultät II, Department für Informatik

Dr. C. Schönberg

Ausgabe: 2020-06-09 14:00 **Abgabe:** 2020-06-16 12:00

Aufgabe 1: *I/O* **(5 Punkte)**

Gegeben sei folgende Modellierung einer Vorlesung (ohne getter- und setter-Methoden):



Die Java-Umsetzung steht im Stud.IP zur Verfügung. Allerdings fehlt dort die Implementierung von drei der `load`-Methoden.

Vervollständigen Sie die gegebenen Klassen, so dass die `main`-Methode der Klasse `LectureDADA` korrekt ausgeführt werden kann und mit der gegebenen Datei "`dada.dat`" die folgende Ausgabe erzeugt:

```
mag123: Defence Against the Dark Arts (DADA), 1998
Severus Snape, Dolores Umbridge, Remus Lupin
- 1998-8-23 8:00, Classroom 3C [Werewolves, Counter Jinxes, Defensive Spells]
```

Beachten Sie dabei die unterschiedlichen Signaturen der vier `load`-Methoden.

Die Datei "`dada.dat`" ist nach folgendem Schema aufgebaut:

```
String, String, String, String, int, { String, String }*,
int, { int, int, int, int, String, int, { String }* }*
```

- `Lecture.number`, `Lecture.title`, `Lecture.shortTitle` und `Lecture.semester` hintereinander als UTF-kodierte `String`s
- die Größe von `Lecture.lecturers` als **`int`**

- jeder einzelne Eintrag von `Lecture.lecturers` hintereinander: `Lecturer.firstName` und `Lecturer.lastName` hintereinander als UTF-kodierte `String`s
- die Größe von `Lecture.schedule` als **`int`**
- jeder einzelne Eintrag von `Lecture.schedule` hintereinander:
  - `Date.year`, `Date.month`, `Date.day` und `Date.hour` hintereinander als **`int`**
  - `Date.lectureHall` als UTF-kodierter `String`
  - die Größe von `Date.topics` als **`int`**
  - jeder einzelne Eintrag von `Date.topics` hintereinander als UTF-kodierter `String`

Aufgabe 2: *Textdateien* **(3 + 6 + 1 Punkte)**

Mit den Klassen `InputStream` und `OutputStream` können in erster Linie Binärdaten gelesen und geschrieben werden, d.h. Daten, die nicht direkt durch einen Menschen gelesen werden sollen, sondern die von einer Maschine ausgewertet werden.

Mit den Methoden der Klassen `Path`, `Paths` und `Files` können sehr einfach auch reine Textdateien, wie sie mit einem Texteditor direkt bearbeitet werden können, gelesen und geschrieben werden.

a) Erweitern Sie die `Lecture`-Klasse aus Aufgabe 1 um eine Methode
`public static void saveText(String filename, Lecture data)`,
die den Inhalt eines Vorlesungs-Objekts in einer menschenlesbaren Form in eine Textdatei schreibt.

b) Erweitern Sie die `Lecture`-Klasse um eine weitere Methode
`public static Lecture loadText(String filename)`,
die eine solche Textdatei wieder einlesen und in ein Vorlesungs-Objekt umwandeln kann.
Sie dürfen zur Vereinfachung verschiedene Annahmen machen, die für die Daten in der Textdatei gelten müssen. Beispielsweise dürfen Sie annehmen, dass die Textdatei im UTF-8 Format kodiert ist, oder dass einzelne Attribute bestimmte Zeichen nicht enthalten dürfen.
Sie dürfen allerdings *nicht* die Multiplizitäten im Klassendiagramm aus Aufgabe 1 einschränken, also beispielsweise dürfen Sie *nicht* annehmen, dass jede Vorlesung immer genau drei Dozenten hat.

c) Dokumentieren Sie, welche Annahmen Sie in Aufgabe 2b gemacht haben.

Hinweis: Schauen Sie sich die Methoden der Klasse `String` an, insbesondere die Funktionsweise der Methoden `String.indexOf`, `String.split`, `String.strip` und `String.substring`.

Aufgabe 3: *Webseite* **(5 Punkte)**

Betrachten Sie die Webseite `https://uol.de/en/computingscience/se/publications`.

Sie sollen auf dieser Webseite die Anzahl von Konferenzartikeln (markiert mit "[inproceedings]"), Journalartikeln (markiert mit "[article]") und Doktorarbeiten (markiert mit "phdthesis") bestimmen und ausgeben.

Natürlich sollen Sie dies nicht von Hand tun, sondern Sie sollen ein Programm schreiben, welches das Zählen für Sie erledigt.

Werfen Sie dazu einen Blick auf folgende nützliche Klassen:

- `URL`, mit dem Konstruktor `URL(url: String)` und der Methode `openConnection(): URLConnection`
- `URLConnection`, mit der Methode `getInputStream(): InputStream`
- `InputStreamReader`, mit dem Konstruktor `InputStreamReader(is: InputStream)`
- `BufferedReader`, mit dem Konstruktor `BufferedReader(r: Reader)` und der Methode `readLine(): String`, welche im Erfolgsfall einen String zurückgibt, oder **`null`**, falls keine Daten mehr gelesen werden können
- `StringBuilder`, mit der Methode `append(s: String)` und der Methode `toString(): String`

Hinweis: Wenn Sie Webseiten automatisiert auslesen, achten Sie bitte peinlichst darauf, dass sie die Seite nicht zu oft aufrufen (z.B. weil das Auslesen versehentlich in eine Schleife oder Rekursion gerutscht ist). Im Extremfall kann dies als Angriff auf die Webseite gewertet werden und zu einer Sperrung oder gar zu rechtlichen Konsequenzen führen! Ein Aufruf pro Minute ist zumeist noch unkritisch, aber ein Aufruf pro Sekunde kann bereits zu automatischen Sperren führen.