Testat-Übung 3: Input/Output

Semesterwoche 12

Abgabe

Testat-Bedingung: 2 von 3 Testat-Übungsserien bestanden.

Spätestens bis und mit Dienstag, 11. Dezember 2018

Gruppenarbeit bitte nur max. 2 Leute. Abgabe via Moodle. Bitte Teilnehmer einer Gruppenarbeit im Kommentar in Moodle vermerken. Bei allfälligen Problemen bitte umgehend ein Email an den Übungsbetreuer senden.

Lernziele

- Das I/O Stream-Konzept von Java vertiefen.
- Byte-Streams und Reader/Writer einsetzen.
- Streams für File- und auch Netz-Zugriff benutzen.
- Objekt-Serialisierung anwenden.
- Regular Expressions üben.

Aufgabe 1: File-Copy

Entwickeln Sie ein Programm, das eine Datei (Text oder binär) eins zu eins in eine neue Datei kopiert. Der Name und Pfad der Quell- und Zieldatei soll über die Konsole eingegeben werden.

Aufgabe 2: Story-Rekonstruktion

In der Vorlage finden Sie die Datei `story-input.txt`. In dieser Datei ist eine kleine Geschichte enthalten, deren Wörter allerdings im Voraus durchnummeriert und dann vermischt wurden. Konkret hat jede Zeile folgendes Format:

```
<zahl>=<wort>
```

Die Zahl der Zeile gibt die Position des Wortes innerhalb der Geschichte an. Die Zahl ist stets dreistellig.

Ihre Aufgabe ist es nun, diese Geschichte aus der Datei wiederherzustellen. Die rekonstruierte Geschichte können Sie anschliessend in eine Ausgabedatei speichern.

Hinweis:

- Zur Lösung eignet sich beispielsweise eine (sortierte) `TreeMap`.

Aufgabe 3: Kontaktverwaltung

Es soll eine einfache Kontaktverwaltung entwickelt werden, welche die Daten in einer Datei dauerhaft speichern und von dort wieder laden kann.

In der Vorlage ist bereits das Gerüst für das Datenmodell (Abbildung 1) und die Bedienung per Konsole enthalten. Ihre Aufgabe ist es nun, die Datenobjekte bei dem Benutzerbefehl „`save`“ zu speichern und bei „`load`“ neu zu laden.

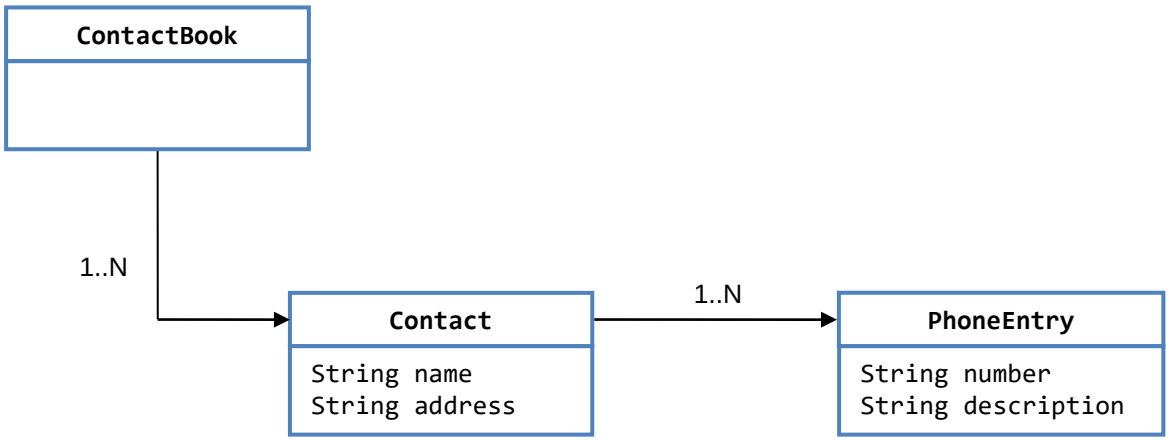

Abbildung 1: Klassendiagramm der Kontaktverwaltung

Die Anwendung (Gerüst der Vorlage) lässt sich beispielsweise wie folgt bedienen

- `load`
- `add contact "Hans Meier" "Bahnhofstrasse 123, 8001 Zürich"`
- `add number "Hans Meier" "043 222 11 33" "office"`
- `add number "Hans Meier" "079 123 45 67" "mobile"`

…

- `find "Hans Meier"`
- `save`

Hinweise:

- Implementieren Sie die TODO-Kommentare in der Klasse `ContactBook` und ergänzen Sie `Contact`.
- Sie können hierzu Objekt-Serialisierung verwenden.

Aufgabe 4: Datums-Konversion

Mit einem kleinen Tool sollen Datums- und Zeitangaben vom US-Format in das deutsche Format übersetzt werden.

Beispiele der gewünschten Konversion:

Eingabe (US Format)	Ausgabe (deutsches Format)
Thu 12/7/2017 10:15 AM	Do 7.12.2017 10:15
Sat 12/24/2017 8:00 PM	Sa 24.12.2017 20:00
Sun 1/1/2018 12:00 AM	So 1.1.2018 0:00

Wochentage: Mon, Tue, Wed, Thu, Fri, Sat, Sun

Schreiben Sie dazu ein kleines Programm:

- Die Eingabe im US-Format soll per Regular Expression zerlegt werden.

Aufgabe 5: Weblink-Analyzer (fakultativ)

Für eine Webseite möchte man gerne wissen, welche anderen Webseiten davon per Hyperlink referenziert werden. Dies soll nun mit einem selbstgemachten Tool analysiert werden. Das Vorgehen ist:

1. Die zu untersuchende Web-URL (z.B. https://www.hsr.ch) wird eingegeben.
2. Mit Hilfe der Klasse `URL` (`import java.net.URL`) kann der `InputStream` der Webseite zugegriffen werden. Beispiel:

```
URL url = new URL("https://www.hsr.ch");
try (InputStream stream = url.openStream()) {
   …
}
```

3. Der Stream wird als Character-Stream ausgelesen und es werden alle Vorkommnisse von Links in der Webseite gesucht. Ein Link kann wie folgt innerhalb der Anführungszeichen vorkommen (Regular Expressions sind hierfür geeignet):

```
<a href="link">
<img src="link">
```

4. Der gefundene Link wird auf der Konsole ausgegeben.

Fakultative Zusatzaufgabe: Die referenzierte Seite unter dem Link können wiederum analysiert werden, solange sie auf dem gleichen Server sind (gleicher Domain-Name, z.B. hsr.ch). Weiter können Sie auch die Seiten abrufen und schauen, ob es nicht erreichbare Web-Links gibt (`FileNotFoundException` bei `URL.openStream()`).