

## Aufgabe 11 *Einführung in die Programmierung mit Java*

### Hinweise

- Die Abgabe dieser Übungsaufgaben muss bis spätestens Sonntag, den 7. Februar 2021 um 23:59 Uhr im ISIS-Kurs erfolgt sein. Es gelten die Ihnen bekannten Übungsbedingungen.
- Lösungen zu diesen Aufgaben sind als gezippter Projektordner abzugeben. Eine Anleitung zum Zippen von Projekten finden Sie auf der Seite des ISIS-Kurses. *Bitte benutzen Sie einen Dateinamen der Form VornameNachname.zip.*
- Bitte beachten Sie, dass Abgaben im Rahmen der Übungsleistung für die Zulassung zur Klausur relevant sind. Durch Plagieren verirken Sie sich die Möglichkeit zur Zulassung zur Klausur in diesem Semester.

**Accounting** Auf diesem Blatt werden Sie ein einfaches Buchungssystem implementieren. Für einen kurzen Überblick über die benutzten Grundbegriffe des Rechnungswesens konsultieren Sie z.B. folgende Seite:

### Soll und Haben

Importieren Sie dazu zunächst das Projekt **Aufgabe 11 - Accounting** in Ihren Workspace und betrachten Sie die Klasse **Account** (für Konten): Ein Konto besitzt die Attribute **Name**, **Offen**, **Wert** und **Delta** (Veränderung). Eine Buchung auf einem Konto soll wie folgt ablaufen:

1. Das Konto wird geöffnet (**open**)
2. Eine Buchung wird vorbereitet (**credit** oder **debit**)
3. Eine Buchung wird durchgeführt (**commit**)
4. Das Konto wird geschlossen (**abort** oder **close**)

### Aufgabe 11.1    *Klassen Account, Asset & Liability (2 Punkte)*

Implementieren Sie die Methoden **credit(int)** und **debit(int)** in den Klassen **Asset** (Aktivkonto) und **Liability** (Passivkonto). Achten Sie darauf, dass Sie nur den Deltawert und nicht den Wert des Kontos selbst verändern. Erst bei einem späteren Aufruf von **commit()** (siehe Klasse **Account**) soll der Wert des Kontos angepasst werden. Werfen Sie passende Ausnahmen (siehe die Klassen im Package **exceptions**), wenn ein Konto nicht geöffnet ist oder der Betrag auf dem Konto nicht ausreicht, um den gegebenen Wert auf dem Konto zu buchen (negativer Wert auf dem Konto).

### Aufgabe 11.2     *Klasse AccountManager (2 Punkte)*

Implementieren Sie die Methoden `getAccount`, `getAsset` und `getLiability` in der Klasse `AccountManager`. Während die Methoden `getAsset(name)` und `getLiability(name)` nur in der Liste von Aktivkonten, bzw. Passivkonten, nach gegebenen Konto suchen soll, soll die Methode `getAccount(name)` *alle* Konten durchsuchen. Werfen Sie eine passende Ausnahme, wenn kein Konto mit dem gegebenen Namen gefunden wurde (und liefern Sie insbesondere nicht den `null`-Wert an den Aufrufer zurück).

### Aufgabe 11.3     *Klasse Accountant (2 Zusatzpunkte)*

Implementieren Sie die Methode `postEntry()` in der Klasse `Accountant`, welche einen Buchungssatz als Parameter übergeben bekommt. Ein korrekter Buchungssatz besteht aus einer Reihe an Konten, die im Soll angesprochen werden, gefolgt von einem Semikolon (als dem Zeichen `;`), gefolgt von einer Reihe an Konten, die im Haben angesprochen werden. Die einzelnen Konto-Wert-Paare sind durch Kommas getrennt, Kontoname und Wert werden durch ein Leerzeichen getrennt. Ein beispielhafter Buchungssatz lautet also wie folgt:

Equity 27, Bonds\_Payable 385; Land 137, Other\_Revenues 275

Überprüfen Sie in Ihrer Implementierung zunächst die Syntax des übergebenen Buchungssatzes und stellen Sie sicher, dass der Betrag auf der Soll-Seite dem Betrag auf der Haben-Seite entspricht. Werfen Sie im Fehlerfall geeignete und aussagekräftige Ausnahmen. Für die anschließende Buchung sollen folgende Punkte abgearbeitet werden:

1. Öffnen Sie zunächst alle beteiligten Konten (`open`)
2. Bereiten Sie die Buchung auf jedem Konto vor (`credit` oder `debit`)
3. Tritt bei keinem Konto ein Fehler auf, dann führen Sie die Buchung durch (`commit`)
4. Tritt irgendein Fehler auf (z.B. wenn der Betrag nicht ausreichend ist), sollen alle bisherigen Änderungen rückgängig gemacht werden (`abort`)
5. Abschließend sollen alle Konten in jedem Fall geschlossen werden (`close`)