Proseminar Programmiermethodik

Universität Innsbruck - Institut für Informatik

A. Blaas, M. Mayerl, B. Murauer, M. Tschuggnall

26.04.2018

1st Midterm Test - Programmieraufgaben (Gruppe 4)

Programmierteil

Verwenden Sie für diese Aufgaben Eclispe. Sie finden die Installation unter `/usr/site/eclipse`.

In diesem Teil des Tests sollen Sie selbst Java programmieren. Sehen Sie sich dazu die bereitgestellten Aufgaben an.

Hinweis

Sie können das Eclipse-Projekt importieren, indem Sie auf File > Import > General > Existing Projects Into Workspace klicken und für das Feld Select archive file die Datei `midterm1.zip` (OLAT) auswählen.

Hinweis

Exportieren Sie Ihre Lösung am Ende des Tests ebenfalls mit Eclipse. Wählen Sie dazu File > Export > General > Archive File und geben Sie Ihrem Archiv **midterm1_<Ihre C-Kennung>.zip** als Namen. Laden Sie diese Datei anschließend als Abgabe auf OLAT hoch.

Aufgabe 1 (Programmablauf) [6 Punkte]

Sehen Sie sich die Klassen an, die im Paket `exercise01` bereitgestellt wurden. Für diese Aufgabe müssen Sie **nur die** `Order`**-Klasse ändern**. In dieser Klasse sollen Bestellungen verwaltet werden. Dazu gibt es ein Feld für einen Kunden, dem die Bestellung zugewiesen wurde. Überlegen Sie sich eine passende Art, um einer Bestellung zusätzlich noch Elemente hinzuzufügen. Beachten Sie dabei folgendes:

- Sehen Sie sich die vier Methoden an, die mit einem `// TODO`-Kommentar versehen sind. Lesen Sie die jeweiligen JavaDoc-Kommentare durch und implementieren Sie das gewünschte Verhalten für jede dieser Methoden. Wenn Sie sich über die Verwendung einer dieser Methoden nicht sicher sind, sehen Sie in der `main`-Methode (in der Datei `OrderManager.java` (OLAT)) nach, wie sie aufgerufen wird.
- Stellen Sie außerdem sicher, dass die Ausgabe der `main`-Methode leserlicher erscheint. Ändern Sie dabei aber nichts an dieser Klasse, sondern beschränken Sie ihre Modifikationen auf die Klasse `Order`. Jedes Order-Objekt soll dabei wie folgt ausgegeben werden:

```
Order by John Smith, including items:
total: 63.90$
average: 21.30$
```

Aufgabe 2 (Kontoverwaltung) [8 Punkte]

Sehen Sie sich die Klasse `Account` im Paket `exercise02` an. Lesen Sie die Aufgabenbeschreibung vollständig durch, bevor Sie mit dem Programmieren beginnen.
Schreiben Sie 2 Klassen `SavingsAccount` und `CurrentAccount`. Beide sollen das Interface `Account` implementieren (zu finden in `Account.java` OLAT). Folgendes Verhalten soll in beiden Klassen auftreten:

- der aktuelle Kontostand wird in einer Variable gespeichert.
- nur positive Beträge sind für den Parameter `amount` erlaubt.
- die `transfer(Account other, float amount)`-Methode hebt von einem Konto ab und überträgt den Betrag auf das Konto `other`, das der Methode übergeben wird.
- falls eine Methode mit falschen Parametern aufgerufen wurde (z.B. ein negativer Betrag), geben Sie eine schlüssige Fehlermeldung auf der Konsole aus und brechen Sie die weitere Abhandlung der Methode ab.

Zusätzlich sollen aber noch folgende Eigenschaften für die einzelnen Klassen gelten:

SavingsAccount

- In der Klasse `SavingsAccount` regelt eine Variable `interest` den Zinsatz des Kontos. Erstellen Sie einen Konstruktor, der diese Variable setzt.
- Stellen Sie eine Methode `increaseByInterest(double percentage)` zur Verfügung, die den Kontostand um die jeweiligen Zinsen erhöht.
- Vor dem Abheben von einem Konto soll kontrolliert werden, ob der Kontostand nach der Transaktion noch positiv ist. Ein Überziehen ist hier nicht erlaubt.

CurrentAccount

- Die Klasse `CurrentAccount` erlaubt das Überziehen des Kontos um einen Betrag, der in einer eigenen Variable `overdraft` gespeichert wird. Erstellen Sie einen Konstruktor, der diese Variable setzt.
- Wenn der Kontostand durch eine Überweisung negativ (aber noch im erlaubten Rahmen) wird, soll eine Warnung auf der Konsole ausgegeben werden.
- Kontostände unter dem gesetzten Rahmen sind nicht erlaubt.

Überlegen Sie während des Implementierens, wie Sie vermeiden können, Funktionalitäten der beiden Klassen `SavingsAccount` und `CurrentAccount` doppelt zu implentieren. Sollten Sie den Code für diese Methoden redundant schreiben (z.B. den gleichen Code für die `deposit`-Methode in beiden Klassen), werden Ihnen Punkte für die Lösung abgezogen.
Sie können Ihren Code mit der gegebenen `Main.java` OLAT -Datei testen.