Grundlagen der Programmierung

Marcel Lüthi
Andreas Morel-Forster
HS 22

Universität Basel
Fachbereich Informatik

Übung 10

Voraussetzung

- Ein JDK ist installiert.
- Installierte IDE, Visual Studio Code sowie die Plugins für Java und Gradle
- Wenn Sie die Vorlesung verpasst haben, dann empfehlen wir Ihnen die Unterlagen anzuschauen.
- Die Zip-Datei, die auch dieses Übungsblatt enthält, muss entpackt werden. Es enthält die gesamte Übungsumgebung. Schreiben Sie ihre Lösungen in die dafür vorgesehenen Dateien, wie in der jeweiligen Übungsaufgabe angegeben.

Wichtiger Hinweis

- *Achten Sie auf guten Programmierstil.*

Hinweise zum Kompilieren und Ausführen der Programme

In dieser Übung ist der Code das erste mal in Paketen organisiert. Dies müssen Sie beim Kompilieren und Ausführen berücksichtigen. Um beispielsweise die Klasse `Fraction` im Paket `fraction` zu kompilieren, wechseln Sie wie gewohnt in das Verzeichnis `src/main/java`, geben dann aber beim Kompilieren den Pfad zur Datei mit an:

```
> javac fraction/Fraction.java
```

Beim Ausführen müssen Sie dann entsprechend den gesamten Namen, inklusive Paket angeben:

```
> java fraction.Fraction
```

Aufgabe 10.1 (Fraction)

Sie finden im Verzeichnis `src/main/java/fraction` die Klassen `Fraction` und `ReducedFraction`.

Implementieren Sie die Methode `reduce` in der Klasse `Fraction`, die einen Bruch kürzt. Um die Brüche zu kürzen, können Sie den Euklidschen Algorithmus verwenden. Mit dessen Hilfe können Sie den *grössten gemeinsamen Teiler* (GGT) des Nenners und Zählers bestimmen.

Beim modernen euklidischen Algorithmus wird in aufeinanderfolgenden Schritten jeweils eine Division mit Rest durchgeführt, wobei im nächsten Schritt der Divisor zum neuen Dividenden und der Rest zum neuen Divisor wird. Der Divisor, bei dem sich Rest 0 ergibt, ist der größte gemeinsame Teiler der Ausgangszahlen. Beispiel für die Ausgangszahlen 3780 und 3528: [1]

$$\begin{aligned} 3780 : 3528 &= 1 \quad \text{Rest } 252 \\ 3528 : 252 &= 14 \quad \text{Rest } 0 \end{aligned}$$

[1] von `https://de.wikipedia.org/w/index.php?title=Gr\%C3\%B6\%C3\%9Fter_gemeinsamer_Teiler&oldid=218004391`

Somit ist 252 der größte gemeinsame Teiler von 3780 und 3528.

Die Klasse `ReducedFraction` soll nun alle Methoden von `Fraction` anbieten, aber es soll immer ein gekürzter Bruch resultieren. Nutzen Sie dabei die bereits vorhandene Funktionalität in der Klasse `Fraction`, aber ohne diese zu kopieren.

Aufgabe 10.2 (Sortierte Liste)

Sie finden im Verzeichnis `src/main/java/sortedlist` die Klasse `SortedList`, welche eine verkettete Liste implementiert, deren Elemente aufsteigend sortiert sind. Die Liste verwaltet Knoten vom Typ `ListNode`.

Passen Sie die Klassen `StringNode` und `IntNode` so an, dass diese von `ListNode` erben. Damit können Sie dann auch in der Liste verwaltet werden. Implementieren Sie alle benötigten Methoden dieser Klassen.

Implementieren Sie dann die `insert` Methode der Klasse `SortedList` welche ein neues Element an der richtigen Stelle in die Sortierte Liste einfügt, sowie die Methode `delete`, welche ein Element aus der Liste löscht.

Testen Sie, dass Sie nun Listen von unterschiedlichen Typen erstellen können, indem Sie einmal eine Liste mit den Strings “first“, “third“’, “second“ erstellen und einmal eine Liste mit den Zahlen $1, 3, 2$. *Bemerkung: Sie können Listen mit unterschiedlichen Typen erstellen, die Typen aber nicht mischen.*

Sie können nun Ihre Implementation auch mit den mitgelieferten Tests testen. Dazu müssen Sie den Kommentar um die Klasse `SortedListTests` in der Datei
`src/test/java/SortedListTests.java` entfernen und dann wie üblich die Tests laufen lassen.

Hinweis 1: Sie brauchen hier explizite Casts um einen Knoten von Typ `ListNode` in den gewünschten Subtyp zu casten, also z.B.
`StringNode sn = (StringNode) aListNode`.

Hinweis 2: Nutzen Sie die Methode `compareTo` der Klasse `String` um zwei Strings zu vergleichen
.

Aufgabe 10.3 (Kassenbon, 4 Punkte)

In dieser Aufgabe werden Sie ein Programm schreiben, welches für einen Einkauf einen Kassenzettel erstellt. Ziel ist es nun, die benötigen Klassen `Kassenbon`, `Artikel` und `Adresse` zu erstellen, um folgende Ausgabe zu erhalten:

```
|=========================|
|    Herbstmesse Basel    |
|        Uni Basel        |
|      Petersplatz 1      |
|        4001 Basel       |
|=========================|

Marroni       2 x  5.40
                       10.80
Magebrot      5 x  1.10
                        5.50
Glühwein      2 x  6.00
                       12.00

----------------------------
Total                  28.30
============================
```

Sie finden im Verzeichnis `src/main/java/kassenbon` die Hauptklasse `Kasse`. An dieser Datei sollten Sie nichts ändern.

- Erstellen Sie die Dateien für die Klassen `Kassenbon`, `Artikel` und `Adresse`.
- Leiten Sie aus der Hauptklasse ab, welche Felder Sie in den jeweiligen Klassen benötigen.
- Schreiben Sie in jeder Klasse einen Konstruktor, der diese mit den übergebenen Werten füllt.
- Fügen Sie der Klasse `Kassenbon` eine Liste `artikelliste` vom generischen Typen `ArrayList` hinzu, die Artikel halten kann. Sie müssen dazu folgenden Teil der API importieren: `import java.util.ArrayList;`.
- Fügen Sie der Klasse `Kassenbon` eine Methode `add` hinzu, um der `artikelliste` einen zusätzlichen `Artikel` hinzuzufügen. Suchen Sie die benötigte Methode, um diesen `Artikel` in der `ArrayList` hinzuzufügen, in der API-Dokumentation.
- Fügen Sie den Klassen `Kassenbon`, `Artikel` und `Adresse` je eine Methode `print` hinzu, diese darf noch leer sein.
- Ihr Gesamt-Programm sollte nun bereits kompilieren, sie müssen dazu nur das Kompilieren der Hauptklasse ausführen.
- Fügen Sie der Klasse `Artikel` eine Methode `getPrice` hinzu, welche den Preis dieses Postens zurückgibt.
- Schreiben Sie nun die `print`-Methoden für `Artikel` und `Adresse` - achten Sie in einem ersten Schritt nur auf den Inhalt, die Formatierung erfolgt später.
- Schreiben Sie nun auch die `print`-Methode für `Kassenbon` diese soll die `print`-Methoden für `Artikel` und `Adresse` aufrufen und das Total berechnen.
- Die Ausgabe sollte nun inhaltlich gleich sein wie die obige Vorlage.

- Versuchen Sie nun die Formatierung der Vorlage anzupassen. Hilfreich ist dazu die Funktion `String.format`.

Hinweis: Für diese Aufgabe testen die automatisierten Tests nur, dass alle Methoden vorhanden sind, und nicht ob diese richtig implementiert sind. Dies sollten Sie an der Ausgabe sehen.

Abgabe Erstellen Sie eine Zip-Datei der gesamten Übungsumgebung (also des Verzeichnisses `uebung010`) und laden Sie dieses auf Adam hoch.