

Dozenten

Prof. Dr. Thomas Vetter
Dr. Marcel Lüthi
Departement
Mathematik und Informatik
Spiegelgasse 1
CH – 4051 Basel

Assistent

Dr. Marcel Lüthi
Departement
Mathematik und Informatik
Spiegelgasse 1
CH – 4051 Basel

Tutoren / Tutorinnen

Claudia Grundke
Viktor Gsteiger
Simon Dold
Timo Steinebrunner
Alexander Rovner
Nikodem Kernbach
Lukas Stöckli

Erweiterte Grundlagen der Programmierung (45398-01)**Blatt 9****[10 Punkte]**

Vorbesprechung 18. November - 22. November

Abgabe 29. November

Allgemeine Hinweise

- Wir empfehlen Ihnen, dass Sie im Buch “Sprechen Sie Java” bis und mit Kapitel 13, sowie Kapitel 18 gelesen haben.

Voraussetzung

- Es gelten dieselben Voraussetzungen wie für Übungsblatt 1. Wenn Sie sich betreffend der Umgebung oder der automatisierten Tests noch unsicher sind, lesen Sie bitte nochmals sorgfältig das Infoblatt oder fragen Sie die Tutoren.
- Die Zip-Datei, die auch dieses Übungsblatt enthält, muss entpackt werden. Es enthält die gesamte Übungsumgebung inklusive der automatisierten Tests. Schreiben Sie Ihre Lösungen in die dafür vorgesehenen Dateien, wie in der jeweiligen Übungsaufgabe angegeben.

Empfohlenes Vorgehen

- Wechseln Sie in den Ordner `src/main/java`. Dort finden Sie die Dateien, in welche Sie Ihren Java Code schreiben.
- Schreiben Sie Ihr Programm, kompilieren Sie dieses mit dem Java Compiler `javac` und führen Sie es mit `java` aus, wie es in der Vorlesung gezeigt wurde.
- Wenn Sie denken, dass alles in Ordnung ist, wechseln Sie zurück ins Übungsverzeichnis `uebung9` und führen da `gradlew test` aus, um zu überprüfen ob Ihre Lösung die automatisierten Tests besteht. Überprüfen Sie auch Ihren Codestil mit `gradlew checkstyleMain`. Falls Ihr Code den Vorgaben entspricht, erhalten Sie einen Bonuspunkt.

Abgabe

Ergänzen Sie die Datei `email.txt` mit Ihrer Unibas E-Mail Adresse. Erstellen Sie eine Zip-Datei der gesamten Übungsumgebung (also des Verzeichnisses `uebung9`) und laden Sie dieses auf Courses (<https://courses.cs.unibas.ch/>) hoch.

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 1 - Fraction

[3 Punkte]

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. 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 2 - Sortierte Liste

[3 Punkte]

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`.

Implementieren Sie 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.

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.

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.

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/sortedlist/SortedListTests.java` entfernen und dann wie üblich mit `gradlew test` 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 3 - Kassenbon

[4 Punkte]

In dieser Aufgabe werden Sie ein Programm schreiben, welches für einen Einkauf einen Kassenzettel erstellt. Sie finden im Verzeichnis `src/main/java/kassenbon` die Hauptklasse `Kasse`. An dieser Datei sollten Sie nichts ändern.

Ziel ist es nun, die benötigten 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
=====
```

- 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 hinzu, die Artikel halten kann: `ArrayList<Artikel> artikelliste = new ArrayList()` (die Bedeutung der eckigen Klammern lernen Sie später in der Vorlesung kennen, Sie erhalten eine Liste die mit Objekten des Typs `Artikel` gefüllt werden 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..