Institut für Informatik
Prof. Dr.-Ing. Elke Pulvermüller
Lars Huning, M.Sc.

Universität Osnabrück, 26.04.2021
Testat bis 05.05.2020, 18:00 Uhr

Übungen zu Einführung in die Software-Entwicklung

Sommersemester 2021

Blatt 3

Aufgabe 3.1: Bruchrechner (35 Punkte)

Erweitern Sie die Klasse `Fraction` aus der mitgelieferten zip-Datei um die Methoden `add(Fraction addend)` und `substract(Fraction subtrahend)`, die die übergebene `Fraction` addieren bzw. subtrahieren und das Ergebnis als neue Fraction zurückgeben.

Implementieren Sie zusätzlich die *Klassenmethode* `parseFraction`, die eine `Fraction` wie von der `toString`-Methode ausgegeben übergeben bekommt und die passende Instanz vom Typ `Fraction` zurückliefert. Um zu überprüfen, ob der übergebene `String` einen korrekten Bruch darstellt, sollen Sie die Methode `matches(String regex)` der Klasse `String` benutzen und für `regex` einen passenden *regulären Ausdruck* einsetzen. Erklären Sie Ihrem Tutor, welche Funktion die einzelnen Komponenten Ihres regulären Ausdrucks haben. In der Dokumentation der Klasse `Pattern` aus der Java-API finden sie alles Wissenswerte über die Generierung eines regulären Ausdrucks in Java. Nutzen Sie zur Verarbeitung des `String` seine Methode `split` und die Methode `Integer.parseInt`.

Verwenden Sie die erweiterte `Fraction` anschließend für ein einfaches Rechenprogramm, das über die Kommandozeile zwei Brüche und einen Operator erhält, die so definierte Rechnung ausführt und das Ergebnis auf der Standard-Konsole ausgibt. Als Operatoren sind `+`, `-`, `*` und `/` zulässig. Achten Sie auf Fehlerbehandlung und eine geeignete Ausgabe von Fehlermeldungen auf `System.err`. Geben Sie bei Fehleingaben auch immer eine Anleitung zur Bedienung des Programms auf der Standard-Konsole aus. Sie müssen in dieser Aufgabe **keine** separate Testklasse schreiben.

Hinweis: Das Symbol `*` hat auf der Konsole eine besondere Bedeutung, deswegen geben Sie dieses beim Testen immer in `"` an. (Beispiel: `java Calculator 1/2 "*" -1/2`).

Hinweis: Sie sollen für diese Aufgabe explizit **nicht(!)** Ihre eigene Lösung der Klasse `Fraction` von Aufgabenblatt 1 verwenden. Nutzen Sie stattdessen die Klasse `Fraction` aus der **mitgelieferten zip-Datei!**

Aufgabe 3.2: EBNF (30 Punkte)

Definieren Sie (schriftlich!) eine Grammatik in EBNF-Syntax um Rechenoperationen mit den Grundrechenarten auf Brüchen darstellen zu können. Berücksichtigen Sie dabei folgendes: Es können beliebig viele Brüche durch Operatoren miteinander verknüpft werden. Eine jede solche Operation und jeder Bruch wiederum kann beliebig tief mit Klammern geschachtelt werden. Ein Bruch besteht immer aus Zähler, Bruchstrich und Nenner und darf keinen Nullteiler haben, der Zähler darf aber sehr

wohl Null sein. Als Operatoren sind `+`, `-`, `*` und `/` erlaubt. Brüche, Klammern und Operatoren sollten immer durch ein Leerzeichen voneinander getrennt sein. Richten Sie sich auch nach folgenden Beispielwörtern der Grammatik:

```
4/3 * 1/2 + -2/2 * ( 3/1 + -3/2 )
( -1/2 ) + 3/4 * 2/1
```

Aufgabe 3.3: `equals` und `hashCode` (20 Punkte)

Lesen Sie die `javadoc`-Dokumentation zu den Methoden `equals` und `hashCode` der Klasse `Object`. Betrachten Sie die Klassen `Student` und `Person` **in der mitgelieferten zip-Datei** und beurteilen Sie anhand einer eigenen separaten Testklasse mit aussagekräftiger Ausgabe, ob die Methoden `equals` und `hashCode` in diesen beiden Klassen korrekt implementiert wurden. Achten Sie dabei auch auf die Beziehungen zwischen Unter- und Oberklasse. Geben Sie für jeden Fehler, den Sie entdecken, mindestens einen Lösungsvorschlag an.

Aufgabe 3.4: Dynamisches Binden (15 Punkte)

Laden Sie die Dateien `Bird.java`, `Parrot.java`, `Dodo.java` und `Aviary.java` aus der **mitegelieferten zip-Datei** herunter, kompilieren Sie sie und führen Sie die Klasse `Aviary` aus. Erklären Sie Ihrem Tutor schriftlich mit Hilfe der Fachbegriffe aus dem Skript jede Zeile der Ausgabe anhand des Quellcodes.