Übung 1

Mit dieser ersten Übung sollen Sie sich noch einmal an Ihre Java-Kenntnisse erinnern und Kontrollstrukturen üben. Sie sind Grundvoraussetzung für die weiteren Vorlesungen/Übungen. Wiederholen Sie, falls nötig, für sich selbst alles!

Grundsätzliches für alle Übungen:

Denken Sie daran, dass Dokumentation, korrekte Einrückung, Einhaltung von Namenskonventionen und sinnvolle Aufteilung des Codes in Klassen/Methoden immer dazugehören.

In den meisten Übungsaufgaben wird kein ausführlicher JUnit-Test verlangt. Ich gehe allerdings grundsätzlich davon aus, dass Sie Ihren Code mindestens ein paar Mal in einer main-Methode aufrufen und prüfen, ob die Ergebnisse stimmen. Code, der nie gelaufen ist, sollte man einfach nicht aus der Hand geben…

a) Brüche (*)

Erstellen Sie eine Klasse zur Speicherung eines mathematischen Bruchs – er besteht aus einem Zähler (oben) und einem Nenner (unten).

- Erstellen Sie einen Konstruktor in der Klasse `Bruch`, der einen Zähler und einen Nenner als Parameter erwartet.
- Erstellen Sie einen Konstruktor in der Klasse `Bruch`, der nur eine ganze Zahl als Parameter erwartet.
- Machen Sie alle Attribute privat und schreiben Sie get-Methoden dazu. Halten Sie set-Methoden für sinnvoll? Wenn ja, ergänzen Sie auch diese.
- Die Klasse soll das Interface `Comparable<Bruch>` implementieren.

Schreiben Sie dann folgende Methoden, die mit Brüchen rechnen:

- `public String toString()`
  liefert eine String-Darstellung des Bruches
- `public Bruch multiplizieren(Bruch b)`
  rechnet `this` * `b`
- `public double ausrechnen()`
  rechnet den Bruch in eine Kommazahl um
- `public void kuerzen()`
  kürzt `this`, d.h. Zähler und Nenner werden durch deren größten gemeinsamen Teiler geteilt. (Es gibt in der Klasse `BigInteger` die Methode `gcd()` oder Sie benutzen den euklidischen Algorithmus, um den größten gemeinsamen Teiler zu bestimmen.)
- `public Bruch kehrwert()`
  liefert den Kehrwert von `this` zurück.
- `public Bruch dividieren(Bruch b)`
  teilt `this` durch `b` und gibt das Ergebnis zurück. Rufen Sie die bisher geschriebenen Funktionen `multiplizieren()` und `kehrwert()` auf, denn Dividieren heißt mathematisch multiplizieren mit dem Kehrwert.

Denken Sie in allen Methoden/Konstruktoren an sinnvolle Prüfungen der Parameter bzw. der Eigenschaften von `this`, der Nenner darf in einem Bruch ja nie 0 sein. Wie gehen Sie mit dieser Fehlersituation um?

Dorothea Hubrich
d_hubrich@web.de

Schreiben Sie ein Hauptprogramm, in dem ein Array von Brüchen angelegt wird (eine Benutzereingabe ist nicht zwingend erforderlich). Sortieren Sie das Array mit Hilfe der Methode `Arrays.sort` zweimal – einmal aufsteigend nach dem Wert des Bruches, einmal absteigend nach der Differenz aus Zähler und Nenner. (Ja, diese Sortierung ist mathematischer Blödsinn… Tun Sie es trotzdem, um die Technik zu erlernen.)

Tipp: Sehen Sie sich die Überladungen von `Arrays.sort` an. Das Interface `Comparator<Bruch>` entspricht in etwa dem Vergleicher aus der Vorlesung.

b) Ratespiel

Schreiben Sie ein Programm, das mit dem Benutzer ein Ratespiel spielt: Das Programm erzeugt eine ganze Zufallszahl zwischen 1 und 100. Der Benutzer soll eine Zahl eingeben, woraufhin das Programm ausgibt, ob diese Zahl kleiner oder größer als die Zufallszahl ist. Das wird solange wiederholt, bis der Benutzer die Zufallszahl erraten hat. Das Programm soll dann noch ausgeben, wie viele Versuche der Benutzer gebraucht hat.

c) Zensurenspiegel

Ein Dozent möchte den Notenspiegel einer Klausur erstellen. Er soll in Ihr Programm nacheinander die Noten (von 1 bis 5, keine Zwischennoten) aller Studenten eintragen. Die Angabe einer 6 (also die höchste Note +1) beendet diese Eingabe. Das Programm gibt dann aus, wie oft jede Note vorkam.

Tipp: Deklarieren Sie ein Array: int anzahlNoten[5];

Erweiterung: In der Schule hat man die Noten 1-6 oder in der Oberstufe 1-15, in den USA werden teilweise Noten von 1 bis 100 verwendet und wer weiß, welche Notensysteme noch benutzt werden.... Lassen Sie den Benutzer die höchste mögliche Note zu Beginn des Programmes eingeben, bevor der eigentliche Notenspiegel erzeugt wird.

d) Uhrzeit

Erstellen Sie eine Klasse namens `Zeit` für die Speicherung einer Uhrzeit (Stunden, Minuten, Sekunden). Schreiben Sie drei Methoden dazu:

- `public void ausgebendeutsch()`
  gibt die Uhrzeit aus, wie es in Deutschland üblich ist (Stunde:Minute:Sekunde)
- `public void ausgebenEnglisch()`
  gibt die Uhrzeit aus, wie es in der englischen Sprache (also z.B. in England, USA oder Neuseeland) üblich ist (Stunde:Minute:Sekunde am bzw. pm). Achtung: Informieren Sie sich über die Darstellung von 12 Uhr Mittags bzw. Mitternacht.
- `public int differenz(Zeit t2)`
  liefert die Anzahl der Sekunden zwischen den beiden Uhrzeiten this und t2.
- Sie können einen Konstruktor schreiben, der Parameter für die drei zu speichernden Werte bekommt, oder einen Standardkonstruktor, der die Uhrzeit vom Betriebssystem abfragt (Klasse Date bzw. Calendar).

Schreiben Sie ein kleines Testprogramm, das Ihre Funktionen aufruft. Die Uhrzeiten dafür können Sie beliebig wählen. Zum Testen sollten Sie alle Funktionen mehrfach aufrufen. Denken Sie darüber nach, wo Schwierigkeiten liegen könnten, was Sie also unbedingt ausprobieren müssen.