Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Angewandte und
Numerische Mathematik

Dr. G. Bohlender
Dipl.–Math. techn. K. Feldhoff
Dipl.–Math. techn. M. Richter

27. Juni 2007

Programmieren: Einstieg in die Informatik mit Java

SS 2007

Aufgabenblatt 11

Aufgabe 25 *Bruchrechnung*

Schreiben Sie in Java eine öffentliche Klasse namens `Bruch` für das Rechnen mit *Brüchen*.

Ein Bruch $p \in \mathbb{R}$ ist dabei definiert als der Quotient zweier ganzen Zahlen p_z, $p_n \in \mathbb{Z}$ mit $p_n \neq 0$. Man definiert nun für zwei Brüche $p = p_z/p_n$ und $q = q_z/q_n$ die folgenden Operationen:

- die *Addition* zweier Brüche:

$$(p, q) \mapsto p + q \quad := \quad \frac{p_z q_n + p_n q_z}{p_n q_n}, \tag{1}$$

- die *multiplikative Inverse* eines Bruches (auch *Kehrwert* genannt):

$$p \mapsto \mathsf{invM}(p) \quad := \quad p_n/p_z, \quad \text{falls } p_z \neq 0, \tag{2}$$

- das *Kürzen* eines Bruches:

$$p \mapsto \mathsf{kuerze}(p) \quad := \quad \frac{p_z/g}{p_n/g} \quad \text{mit } g := \mathsf{ggT}(p_z, p_n), \tag{3}$$

  $\mathsf{ggT}(a, b)$ bezeichnet dabei den größten gemeinsamen Teiler zweier Zahlen a, $b \in \mathbb{Z}$,

- und die *Auswertung eines Kettenbruches* $K_m(a_0, a_1, a_2, \ldots, a_{m-1}) \in \mathbb{R}$ zu $m \in \mathbb{N}$ gegebenen Zahlen $a_0, a_1, a_2, \ldots, a_{m-1} \in \mathbb{N}$:

$$K_m \quad := \quad a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots + \cfrac{1}{a_{m-2} + \cfrac{1}{a_{m-1}}}}}. \tag{4}$$

  Der Wert des Kettenbruches $K_m(a_0, a_1, a_2, \ldots, a_{m-1})$ kann dabei mit Hilfe des nachfolgend angegebenen Verfahrens berechnet werden:

$$\text{Initialisierung:} \quad K = a_{m-1}, \tag{5}$$

$$\text{Iteration:} \quad K = a_i + 1/K \quad \text{für } i = m-2, m-3, m-4 \ldots, 0, \tag{6}$$

$$\text{Ergebnis:} \quad K_m = K. \tag{7}$$

Verwenden Sie bei der Umsetzung die folgenden Elemente:

- Zwei private Instanzvariablen `zaehler` und `nenner` vom Typ `int` zur Darstellung des Zählers und des Nenners eines Bruches.
- Einen öffentlichen Konstruktor mit zwei formalen Parametern vom Typ `int`, in dem der Zähler und der Nenner eines Bruches mit den übergebenen Werten initialisiert wird.
  Prüfen Sie vor der Initialisierung, ob der Nenner von Null verschieden ist. Geben Sie andernfalls eine Fehlermeldung auf der Konsole aus, und beenden Sie das Programm mit Hilfe der Methode `System.exit`.
- Eine öffentliche Instanzmethode `toString` ohne formale Parameter, in der der Wert eines Bruches p in der Form p_z/p_n in Form einer Variablen vom Typ `String` an die aufrufende Methode zurückgeliefert wird.
- Eine öffentliche Instanzmethode `add` mit einem formalen Parameter vom Typ `Bruch`, in der die Summe zweier Brüche p und q gemäß Gleichung (1) berechnet, das Ergebnis mittels der Methode `kuerze` gekürzt und in Form einer Variablen vom Typ `Bruch` an die aufrufende Methode zurückgegeben wird.
- Eine öffentliche Instanzmethode `invM` ohne formale Parameter, in der die multiplikative Inverse eines Bruches p gemäß Gleichung (2) berechnet und das Ergebnis in Form einer Variablen vom Typ `Bruch` an die aufrufende Methode zurückgegeben wird.
  Prüfen Sie vor der Berechnung, ob der Zähler p_z von Null verschieden ist. Geben Sie andernfalls eine Fehlermeldung auf der Konsole aus, und beenden Sie das Programm.
- Eine öffentliche Instanzmethode `kuerze` ohne formale Parameter, in der ein Bruch p gemäß Gleichung (3) gekürzt und das Ergebnis in Form einer Variablen vom Typ `Bruch` an die aufrufende Methode zurückgegeben wird.
  Bestimmen Sie dabei den größten gemeinsamen Teiler des Zählers p_z und des Nenners p_n mit Hilfe einer nicht von Ihnen zu implementierenden Klassenmethode `berechneGGT`. Die Methode besitzt zwei formale Parameter vom Typ `int` sowie einen Rückgabewert vom Typ `int` für den größten gemeinsamen Teiler.
- Eine öffentliche Klassenmethode `bestKettenbruch` mit einem formalen Parameter vom Typ `int[]` zur Darstellung $m \in \mathbb{N}$ natürlicher Zahlen a_0, a_1, a_2, …, a_{m-1}, in der der Kettenbruch $K_m(a_0, a_1, a_2, \ldots, a_{m-1})$ mit Hilfe des in den Gleichungen (5) bis (7) angegebenen Verfahrens ausgewertet und das Ergebnis in Form einer Variablen vom Typ `Bruch` an die aufrufende Methode zurückgegeben wird.
  Erzeugen Sie zunächst zwei Referenzen `K` und `ai` vom Typ `Bruch` zur Darstellung des Wertes K sowie der Zahlen a_i. Bilden Sie anschließend eine Schleife über i, in der Sie den Wert K gemäß Gleichung (6) unter Verwendung der Methoden `add` und `invM` aktualisieren.
- Ein Hauptprogramm, in dem der Wert des Kettenbruches $K_5(1, 1, 1, 1, 1)$ $(= 8/5)$ mit Hilfe der Methode `bestKettenbruch` berechnet und anschließend unter Verwendung der Methode `toString` auf der Konsole ausgegeben wird.

BITTE WENDEN

Aufgabe 26 *Etappenverlauf eines Radrennens*

Schreiben Sie in Java ein Applet, in dem der Etappenverlauf eines Radrennens graphisch dargestellt wird und auf Knopfdruck Städtenamen und Entfernungen zwischen den Städten der Etappe dem Streckenverlauf hinzugefügt bzw. entfernt werden.

Die Entfernung $d_k \in \mathbb{R}_{\geq 0}$ zwischen der $(k-1)$-ten und der k-ten Stadt berechnet sich dabei für $k \in \{1, 2, 3, \ldots, N-1\}$ wie folgt:

$$d_k \quad := \quad \left((x_k - x_{k-1})^2 + (y_k - y_{k-1})^2\right)^{1/2}. \tag{8}$$

Das Paar (x_k, y_k) gibt die Koordinaten der k-ten Stadt an, $N \in \mathbb{N}$ die Anzahl der Städte.
Mathematische Koordinaten $(x, y) \in \mathbb{R}^2$ sollen mit Hilfe der beiden Transformationen S und T in den Gleichungen (9) und (10) in Bildschirmkoordinaten (p, q) mit $p \in \{0, 1, 2, \ldots, p_{\mathsf{max}}\}$, $q \in \{0, 1, 2, \ldots, q_{\mathsf{max}}\}$ umgerechnet werden. $p_{\mathsf{max}} \in \mathbb{N}$ und $q_{\mathsf{max}} \in \mathbb{N}$ sind dabei die Abmessungen des Applets.

$$S : [a; b] \longrightarrow \{0, 1, 2, \ldots, p_{\mathsf{max}}\}, \quad x \mapsto S(x) := \left\lfloor \frac{x-a}{b-a} p_{\mathsf{max}} \right\rfloor, \tag{9}$$

$$T : [c; d] \longrightarrow \{0, 1, 2, \ldots, q_{\mathsf{max}}\}, \quad y \mapsto T(y) := \left\lfloor \frac{y-d}{c-d} q_{\mathsf{max}} \right\rfloor. \tag{10}$$

$\lfloor \cdot \rfloor$ bestimmt dabei zu einer reellen Zahl die größte ganze Zahl, die kleiner oder gleich der reellen Zahl ist.

a, b, c und $d \in \mathbb{R}$ mit $a < b$ und $c < d$ geben die Grenzen des Gebietes $\Omega := [a; b] \times [c; d]$ an, auf dem der Streckenverlauf gezeichnet werden soll.

Definieren Sie eine öffentliche Klasse `RadEtappe`, leiten Sie diese von der Klasse `Applet` ab, und verwenden Sie die folgenden Elemente:

- Eine geschützte Instanzvariable `btInfo` vom Typ `Button`, drei private Instanzvariablen `pMax`, `qMax` und `N` vom Typ `int` zur Speicherung der Abmessungen p_{max} und q_{max} des Applets sowie der Anzahl N an Städten, zwei private Instanzvariablen `x` und `y` vom Typ `double[]`, in denen die Koordinaten der Städte gespeichert werden, eine private Instanzvariable `Stadtname` vom Typ `String[]` zur Speicherung der Städtenamen und eine geschützte Instanzvariable `zeigeInfo` vom Typ `boolean` zur Kontrolle der Ausgabe auf der graphischen Oberfläche. Desweiteren vier private Instanzvariablen `a`, `b`, `c` und `d` vom Typ `double` für das Gebiet Ω, auf dem der Streckenverlauf gezeichnet werden soll.

- Zwei private Instanzmethoden `transformX` und `transformY`, jeweils mit einem formalen Parameter vom Typ `double` sowie einem Rückgabewert vom Typ `int`, in denen die mathematische x-Koordinate bzw. die mathematische y-Koordinate in die entsprechende Bildschirmkoordinate $p = S(x)$ bzw. $q = T(y)$ gemäß der Transformation S bzw. T in den Gleichungen (9) und (10) umgerechnet wird.

- Eine öffentliche Instanzmethode `init` ohne Rückgabewert und ohne formale Parameter, in der das Applet initialisiert wird.
  Verwenden Sie ein freies Layout, und fügen Sie den Knopf `btInfo` der Pixelgröße 210×30 mit der Beschriftung „Mehr Informationen” der graphischen Oberfläche des Applets an der

Stelle $(10, 10)$ hinzu. Verknüpfen Sie den Knopf `btInfo` mit einer Instanz der inneren Klasse `AktionInfo` (siehe unten).
Bestimmen Sie danach mit Hilfe der Methode `getSize` die Abmessungen $p_{\max}$ und $q_{\max}$ des Applets, und speichern Sie diese in den Variablen `pMax` und `qMax` ab.
Geben Sie das Gebiet Ω in Form der Gebietsgrenzen a, b, c und d explizit vor, und lesen Sie innerhalb eines `try-catch`–Blockes die Koordinaten und Städtenamen mittels der Methode `readLine` der Klasse `BufferedReader` aus der unten angegebenen Beispieldatei ein. Beachten Sie dabei, daß der erste Eintrag in der Datei die Anzahl N an Städten angibt und jeweils zwei Datensätze, bestehend aus Koordinaten und Stadtnamen, durch eine Leerzeile voneinander getrennt sind. Erzeugen Sie in Abhängigkeit von N die Felder `x`, `y` und `Stadtname`, lesen Sie anschließend in einer Schleife über i für $i \in \{0, 1, 2, \ldots, N-1\}$ die Koordinaten (x_i, y_i) sowie den Namen der i-ten Stadt ein, und speichern Sie diese in der i-ten Komponente des zugehörigen Feldes. Wandeln Sie dabei gegebenenfalls erzeugte Zeichenketten mit Hilfe der Methode `parseDouble` der Hüllklasse `Double` in Variablen vom Typ `double` um. Schließen Sie die Datei wieder, und fangen Sie die Ausnahme vom Typ `IOException`, die von der Methode `readLine` ausgelöst werden kann, im `catch`–Teil des `try-catch`–Blockes durch Ausgabe einer Warnmeldung auf der Konsole ab.

- Eine öffentliche Instanzmethode `paint` ohne Rückgabewert und mit einem formalen Parameter vom Typ `Graphics`, in der der Streckenverlauf der Etappe mittels Geradenstücken gezeichnet und in Abhängigkeit von der Variablen `zeigeInfo` die Städtenamen und die Entfernungen zwischen den Städten ein- bzw. ausgeblendet werden.
  Rechnen Sie in einer Schleife über i alle Koordinaten (x_i, y_i) mit Hilfe der beiden Methoden `transformX` und `transformY` in Bildschirmkoordinaten um, und zeichnen Sie den Streckenabschnitt zwischen der $(i-1)$-ten und der i-ten Stadt durch Aufruf der Methode `drawLine` der Klasse `Graphics`.
  Geben Sie zusätzlich in einer Schleife über i den Namen der i-ten Stadt an der Stelle (x_i, y_i) sowie in einer weiteren Schleife über k die Entfernung d_k zwischen der $(k-1)$-ten und k-ten Stadt an der Stelle $((x_{k-1} + x_k)/2, (y_{k-1} + y_k)/2)$ ohne Nachkommastellen gemäß Gleichung (8) auf der graphischen Oberfläche des Applets aus, falls `zeigeInfo` den Wert `true` besitzt. Transformieren Sie dabei wieder die Koordinaten mit Hilfe der Methoden `transformX` und `transformY` in Bildschirmkoordinaten.

Erstellen Sie eine innere Klasse `AktionInfo` innerhalb der Klasse `RadEtappe`, die die Schnittstelle `ActionListener` implementiert. Verwenden Sie dabei die folgenden Elemente:

- Eine öffentliche Instanzmethode `actionPerformed` ohne Rückgabewert und mit einem formalen Parameter vom Typ `ActionEvent`, in der in Abhängigkeit von der Variablen `zeigeInfo` die Beschriftung des Knopfes `btInfo` geändert und das Applet neu gezeichnet wird.
  Negieren Sie zunächst den Wert von `zeigeInfo`. Ändern Sie anschließend die Beschriftung des Knopfes `btInfo` mittels der Methode `setLabel` der Klasse `Button` zu „Mehr Informationen”, falls `zeigeInfo` den Wert `false` besitzt, bzw. auf „Weniger Informationen”, falls `zeigeInfo` den Wert `true` besitzt. Zeichnen Sie zum Schluß das Applet durch Aufruf der Methode `repaint` neu.

Testen Sie Ihr Programm für $a = 0$, $b = 30$, $c = 0$, $d = 30$ und der von den Webseiten zur Vorlesung herunterladbaren Beispieldatei `NancyKa.dat`.