Aufgabenblatt 10

Bearbeitungsende: 18.12.2022

Eine der bewerteten Aufgaben gilt als Bonusaufgabe: der erreichbare Punkt wird zwar angerechnet, aber nicht in der Menge der erreichbaren Punkte mitgezählt.

Aufgabe 1 [Programmierung – nicht bewertet]

Hinweise: Erstellen Sie zuerst ein UML-Klassendiagramm zu dieser Aufgabe, bevor Sie eine Lösung implementieren.

Ergänzen Sie die Klasse `Datum` aus B08A1 in folgender Weise:

- Die Methode `monatFuerName` nimmt einen ganzzahligen Wert im Bereich $[1;12]$ an. Sie gibt den Namen des entsprechenden Monats (gezählt ab Januar) ohne Umlaute als Zeichenkette zurück.

  Z.B. wird für das Argument 3 der Wert `"Maerz"` geliefert.

- Die Methode `nameFuerMonat` nimmt umgekehrt einen Monatsnamen als Zeichenkette an und gibt den Zahlwert des Monats zurück.

  Z.B. wird für das Argument `"April"` der Wert 4 geliefert.

- Wenn Methode `setFormatTZ` als Argument das Zeichen `'0'` erhält (unsinnig als Trennzeichen), sollen Aufrufe von `toString` für alle `Datum`-Objekte fortan das Datum in ausgeschriebener Form als Zeichenkette liefern.

  Z.B. soll `toString` das Ergebnis `"15. August 1980"` liefern, wenn das Trennzeichen als `'0'` und die Reihenfolge als `"tmj"` gewählt worden sind (und natürlich das Datum den entsprechenden Wert hat).

  Hat die Reihenfolge den Wert `"mtj"`, liefert `toString` das Ergebnis `"August 15. 1980"`.

  Hinweis: Bestimmen Sie in `toString` erst, wie die einzelnen Darstellungen von Jahr, Monat, Tag und Trennzeichen als Zeichenkette aussehen. Legen Sie dann fest, in welcher Reihenfolge sie anzuordnen sind.

Sofern möglich, sollten Methoden Klassenmethoden sein.

Aufgabe 2 [Programmierung – nicht bewertet]

Ergänzen Sie die Klasse `Datum` aus B08A1 und Aufgabe 1 um die Methoden:

- `istSchaltjahr` aus B03A4
- `tageInMonat` nimmt als Argumente eine Jahresangabe (positiv, ganzzahlig) sowie eine Monatsangabe (ganzzahlig im Bereich $[1;12]$) an. Sie gibt zurück, wieviele Tage dieser Monat in diesem Jahr hat (i.W. in Vorlesung gezeigt).

  Dabei ist darauf zu achten, dass der Februar in Schaltjahren 29 statt 28 Tage hat.

Wenn eines ihrer Argumente ungültig ist, sollen Methoden und Konstruktoren eine `IllegalArgumentException` werfen.

Die Exception soll eine Fehlerbeschreibung tragen, die in der Regel folgenden Inhalt hat:

`ungueltiger Wert fuer` *`JMT`*`: ` *`Wert`*

Darin steht *`JMT`* entsprechend für `Jahr`, `Monat` oder `Tag` und *`Wert`* für den ungültigen Wert.

Für die Methode `setFormatRF` soll die Fehlerbeschreibung folgenden Inhalt haben:

`ungueltiger Wert fuer Format-Reihenfolge: ` *`Wert`*

Allgemein soll dabei gelten:

Ist das Argument eine `null`-Referenz, wird eine `NullPointerException` geworfen.

Ist das ungültige Argument eine Zeichenkette `"`*`zeichenkette`*`"`, erscheint ihr *`Wert`* auch als `"`*`zeichenkette`*`"` in der Meldung.

Beispiel: `ungueltiger Wert fuer Format-Reihenfolge: "jtm"`

Hinweis: Das Zeichenliteral `'\"'` stellt das (doppelte) Anführungszeichen dar.

Aufgabe 3 [Programmierung]

Hinweise: Erstellen Sie zuerst ein UML-Klassendiagramm zu dieser Aufgabe, bevor Sie eine Lösung implementieren.

Ergänzen Sie die Klasse `Rational` aus B08A2 so, dass eine `ArithmeticException` geworfen wird, wenn ein ungültiger Bruch mit 0 im Nenner entstünde.

Ergänzen Sie die Klasse dann um eine Klassenmethode `parse`. Die Methode nimmt eine Zeichenkette an und liefert eine Referenz auf ein neues `Rational`-Objekt.

Die Methode liest die Daten für dieses Objekt aus der Zeichenkette, die die Form `"`*`GZ/GZ`*`"` haben sollte. Darin steht *`GZ`* jeweils für eine Ganzzahldarstellung ohne vorangehende oder folgende White Spaces, ggf. mit Vorzeichen `'+'` oder `'-'`. Die erste Zahl gibt den Dividend, die zweite Zahl den Divisor (nicht unbedingt in normalisierter Form!) für das neue Objekt an.

Beispiele: `"-2/3"` `"24/-8"` `"-20/-20"`

Die Methode soll eine `IllegalArgumentException` werfen, wenn die vorliegende Eingabe nicht das erwartete Format hat. Besteht aber der Fehler darin, dass der Divisor 0 ist, soll eine `ArithmeticException` geworfen werden.

Hinweise: Verwenden Sie die Methoden `indexOf` und `substring` aus `String` sowie `parseInt` aus `Integer`.

Werfen Sie Exceptions nur dort selbst, wo es nötig ist. Wenn eine verwendete Methode / Operation sowieso eine entsprechende Exception wirft, wenn ein bestimmter Fehler vorliegt, müssen Sie diesen Fall nicht selbst explizit (per Fallunterscheidung und `throw`-Anweisung) behandeln.

Die Klasse `Mathe` mit der Methode `ggT`, die von `Rational` benötigt wird, liegt im Praktomat bereit und muss/soll nicht mit hochgeladen werden.

Aufgabe 4 [Programmierung]

Hinweise: Erstellen Sie zuerst ein UML-Klassendiagramm zu dieser Aufgabe, bevor Sie eine Lösung implementieren.

Die Klasse `RationalAusdruck` soll ermöglichen, Ausdrücke aus rationalen Zahlen interaktiv zu berechnen. Dazu soll sie Klasse `Rational` aus Aufgabe 3 verwenden und folgende Klassenmethoden bereitstellen:

- `auswertung` nimmt, in dieser Reihenfolge,
  1. eine `Rational`-Referenz `a`
  2. ein Zeichen `op`
  3. eine `Rational`-Referenz `b`

  an und unterscheidet zwischen den folgenden Werten für `op`:

  `'+'` Addition
  `'-'` Subtraktion
  `'*'` Multiplikation
  `'/'` Division

  Die Methode ruft die entsprechende `Rational`-Methode für `a` und `b` auf und gibt das Resultat (ebenfalls eine `Rational`-Referenz) zurück.

  Für einen ungültigen Wert von `op` soll eine `IllegalArgumentException` geworfen werden, bei Division durch 0 eine `ArithmeticException`.

- `next` nimmt eine `Scanner`-Referenz als Argument an.

  Die Methode liest mit Hilfe der Methode `parse` aus Aufgabe 3 eine Eingabe der Form

  a op b

  mit mindestens je einem White Space vor und hinter *op*, ansonsten beliebig vielen weiteren White Spaces vor und nach jedem Element. *a* und *b* sind darin `Rational`-Werte im in Aufgabe 3 beschriebenen Format. *op* ist ein Zeichen. Die Methode übergibt diese Werte an die Methode `auswertung` und gibt dann deren Ergebnis zurück.

  Für eine ungültige Eingabe soll die Methode eine `java.util.InputMismatchException` werfen, für eine Division durch 0 eine `ArithmeticException`.

- `dialog` nimmt keine Argumente an und liefert kein Resultat. Sie soll einen `Scanner` für die Tastatur öffnen und bis Eingabeende die Methode `next` aufrufen, um Eingaben einzulesen.

  Für jede Eingabe soll das Resultat *r* als

  = *r*

  gefolgt von einem Zeilenumbruch auf dem Bildschirm ausgegeben werden.

  Beispiel: Ein Aufruf von `dialog()` könnte folgende Ausgabe auf dem Bildschirm produzieren. Zeilen ohne `'='` sind Tastatureingaben, in denen man beliebig White Spaces ergänzen kann:

```
8/12 + 36/27
= 2/1
18/21 * -35/8
= -15/4
  14/4  -  24/15
= 19/10
```

Im Fall eines Fehlers (auch Division durch 0) wirft die Methode eine (geprüfte) `java.io.IOException`.

Prüfen Sie an möglichst wenig Stellen selbst auf Fehler und werfen Sie nur dort, wo nötig, selbst Exceptions.

Aufgabe 5 [Programmierung]

Die Summe $\sum_{k=0}^{n} \frac{(-1)^k}{2 \cdot k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots$ konvergiert für $n \to \infty$ gegen $\frac{\pi}{4}$.

Schreiben Sie eine Klasse `Mathe` mit folgenden Methoden zur Berechnung und Rückgabe von π:

(a) `pi1(int n)`: Berechne die Summe bis zum Wert `n` $\geqslant 0$.

(b) `pi2(double eps)`: Berechne und addiere Summanden so lange, bis für den nächsten Summanden s gilt, dass $|s| <$ `eps`, wobei $0 <$ `eps` $\leqslant 1$.

Wird kein Argument übergeben, sei `eps` $= 0.00000001 = 10^{-8}$ (Überladen der Methode).

(c) `pi3(double eps)`: Berechne und addiere für `eps` > 0 Summanden so lange, bis der berechnete Wert für π weniger als `eps` vom Java-internen Wert `Math.PI` abweicht.

Wird kein Argument übergeben, sei `eps` $= 0.00000001 = 10^{-8}$ (Überladen der Methode).

Für ungültige Argumentwerte soll eine `IllegalArgumentException` geworfen werden.

Verwenden Sie jeweils eine Schleife der geeignetsten Form.

Hinweise:

- Wenn in einer Operation ein ein `double`- und ein `int`-Wert verwendet werden, wird letzterer automatisch in einen `double`-Wert umgewandelt.
- Beachten Sie, dass die Formel gegen $\frac{\pi}{4}$ konvergiert, Sie aber π annäherungsweise berechnen und zurückgeben sollen.
- Der Ausdruck $(-1)^i$ bewirkt nur einen Wechsel des Vorzeichens in jeder Iteration. Berechnen Sie dafür keine Potenz; verwenden Sie insbesondere nicht `Math.pow`. Überlegen Sie, wie Sie den Vorzeichenwechsel einfacher realisieren können.

Lösungen zu mit [**Programmierung**] markierten Aufgaben sind im **Praktomat** einzureichen.

Lösungen zu mit [**Programmierung – nicht bewertet**] markierten Aufgaben können ebenfalls im **Praktomat** eingereicht werden, werden jedoch nicht bewertet.

Allgemeine **Fragen** zu den Aufgaben können Sie im **LEA-Forum „Übungsaufgaben“** stellen.

Hilfe bei der Lösung der Aufgaben erhalten Sie in den **Übungen** und in der **Studierwerkstatt** .