Allgemeines Programmierpraktikum **SS 2008**
Dr. Henrik Brosenne Georg-August Universität Göttingen
Institut für Informatik

Übung 5

Testat nach Mittwoch, 28.05., 14.00 Uhr in der Gruppenübung.

Vorbereitung

Java

Informieren Sie sich in der *Java 2 Platform Standard Edition 5.0 API Specification* `http://java.sun.com/j2se/1.5.0/docs/api/` über die Collection `java.util.Stack` und die Schnittstelle `java.lang.Comparable`. Betrachten Sie jeweils die nicht typsichere Version.

Aufgabe 1 – 50 Punkte

Normierte rationalen Zahlen

1. Implementieren Sie eine Klasse `NormRational`, die von der Klasse `Rational` von Übung 4, Aufgabe 1 erbt.

   Die Klasse `NormRational` speichert rationale Zahlen in einer Normalform für die Folgendes gilt.

   - Der Zähler und Nenner sind teilerfremd, das wird erreicht indem beide durch ihren größten gemeinsamen Teiler (ggt) geteilt werden. Der ggt kann z.B. mit dem Euklidischen Algorithmus ermittelt werden.
   - Der Nenner ist immer positive, d.h. negative Brüche besitzen einen negativen Zähler.

   Auch nach einer beliebigen arithmetischen Operation liegt der Bruch in Normalform vor.

   Implementieren Sie alle Methoden, die im Aufgabenteil 2. benötigt werden.

2. Schreiben Sie ein ausführbare Klasse, die Ausdrücke in umgekehrter polnischer Notation auf der Kommandozeile übergeben bekommt und das Ergebnis ausgibt. Gerechnet wird mit Objekten der Klasse `NormRational`.

   Zulässige Ausdrücke sind wie folgt aufgebaut.

   - Zahlen sind ganzzahlig und nicht-negativ, d.h. sie bestehen nur aus Ziffern. Das schließt nicht aus, dass das Resultat eines Ausdrucks negativ oder rational ist.

- Operatoren sind `+` (Addition), `-` (Subtraktion), `*` (Multiplikation) und `/` (Division).

Der Stack, auf dem die Operanden abgelegt werden, ist ein Objekt der Klasse `java.util.Stack`.

Hinweis

Ein *unchecked warning* z.B. die Folgende kann ignoriert werden.

```
Note: xxx.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Allgemeine Anforderungen

Kommentieren Sie Ihren Programmtext **ausführlich**. Spärliche und/oder schlechte Kommentierung führt zu Punktabzug.

Aufgabe 2 – 50 Punkte

Vergleichbare rationale Zahlen

1. Implementieren Sie eine Klasse `CompRational`, die von der Klasse `Rational` von Übung 4, Aufgabe 1 erbt (oder von `NormRational`) und die Schnittstelle `java.lang.Comparable` implementiert.

   Die Methode `compareTo` der Schnittstelle `Comparable` wird wie folgt implementiert.

   - `compareTo` gibt einen Wert kleiner 0 zurück, wenn das aktuelle Element vor dem zu vergleichenden liegt.
   - `compareTo` gibt einen Wert größer 0 zurück, wenn das aktuelle Element hinter dem zu vergleichenden liegt.
   - `compareTo` gibt 0 zurück, wenn das aktuelle Element und das zu vergleichende gleich sind.

2. Programmieren Sie eine Klasse `BinarySearchTree`, einen binären Suchbaum für Referenzen auf Objekte von Klassen die `java.lang.Comparable` implementieren.

   Programmieren Sie mindestens folgende Methoden.

   - `insert` fügt ein Element in den Baum ein. Enthält der Baum schon ein äquivalentes Element (`compareTo == 0`), wird eine Exception ausgelöst.
   - `contains` gibt Auskunft, ob ein äquivalentes Element in Baum enthalten ist oder nicht.
   - `getMaximum` liefert das maximale Element im Baum zurück.
   - `getMinimum` liefert das minimale Element im Baum zurück.

3. Schreiben Sie ein ausführbare Klasse, die rationale Zahlen auf der Kommandozeile übergeben bekommt und die größte und kleinste der Zahlen ausgibt.

   Die Kommandozeile enthält eine gerade Anzahl von ganzen Zahlen. Zwei aufeinanderfolgende Zahlen werden als Zähler und Nenner einer rationalen Zahl (`CompRational`) interpretiert.

   Speichern Sie die übergebenen Zahlen in einem binären Suchbaum und lassen Sie das größte und kleinste Element ausgeben.

Hinweis

Ein *unchecked warning* z.B. die Folgende kann ignoriert werden.

```
Note: xxx.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Allgemeine Anforderungen

Kommentieren Sie Ihren Programmtext **ausführlich**. Spärliche und/oder schlechte Kommentierung führt zu Punktabzug.

Umgekehrte polnische Notation

Eine Modifikation der polnischen Notation ist die umgekehrte polnische Notation (UPN, Postfix-Notation), bei der der Operator hinter die verbundenen Operaden geschrieben wird.

- Zahlen sind Operanden
- Operatoren beschreiben Verknüpfungen zwischen 2 Operanden. Ein Operator wirkt auf die beiden Operanden direkt vor ihm. Die Anwendung eines Operators auf seine Operanden wird als Einheit betrachtet und ist wieder ein Operand.

Diese Forderungen macht Klammern und Bindungsregeln für Operatoren überflüssig.

Beispiel

```
  1 2 + = 3                          UPN
  1 2 + 3 * = 9
  1 2 3 * + = 7

  1 2 + 3 / 4 * 5 6 - +              UPN
= ((((1 2 +) 3 /) 4 *) (5 6 -) +)    Klammerung durch Operatoren erzungen
= ((((1 + 2) / 3) * 4) + (5 - 6))    Infix-Notation
= 3

  5 4 3 2 1 + * - /                  UPN
= (5 (4 (3 (2 1 +) *) -) /)          Klammerung durch Operatoren erzungen
= (5 / (4 - (3 * (2 + 1))))          Infix-Notation
= -1
```

Binäre Suchbäume

In einem binären Suchbaum gelten folgende Eigenschaften.

- Jeder Knoten enthält ein Element der gespeicherten Menge und hat maximal zwei Unterbäume.
- Der linke Unterbaum enthält nur Elemente, die kleiner sind als das im Knoten gespeicherte.
- Der rechte Unterbaum enthält nur Elemente, die größer sind als das im Knoten gespeicherte.

Diese Eigenschaft gilt rekursiv für die Unterbäume und muss z.B. bei Einfügeoperationen erhalten bleiben, wobei eine Balancierung allerdings nicht notwendig ist.

Beispiel

Einfügeoperation in einem binären Suchbaum.