Friedrich-Schiller-Universität Jena **Fakultät für Mathematik und Informatik** **Institut für Informatik** **Lehrstuhl für Programmiersprachen und Compiler**	**Höhere Programmierung** **SS 2001**	**Praktikumsblatt 7** **Ausgabe: 26.06.2001** **Termin: 28.06.2001**

Aufgabe 1: Package für rationale Zahlen

In diesem Praktikum soll ein Paket zur Manipulation von rationalen Zahlen entwickelt werden. Zu realisieren sind

- Der ZahlenTyp `BruchTyp`, der sich aus dem Zähler und dem Nenner zusammensetzt. Der Nenner soll immer positiv sein.
- Die Grundrechenarten (+, -, *, /) als Operatoren inklusive einer Funktion BruchKürzen, die einen gegebenen Bruch mit dem größten gemeinsamen Teiler von Zähler und Nenner kürzt.
- Die Typumwandlung von rationalen Zahlen in den Zahlentyp Float.
- Ein Programm, das die Funktionalität der Einheit testet.

Die Struktur eines `PACKAGE`:

Ein PACKAGE setzt sich meistens aus der Spezifikation und dem Package Body zusammen. In der Spezifikation wird das Interface zu anderen Programmen formuliert. Alles, was im öffentlichen Teil der Spezifikation steht (exportierte Größen), ist von außen sichtbar. (Der private-Teil wird hier nicht benötigt).
Die Rümpfe der in der Paketspezifikation definierten UPe müssen im zugehörigen Package Body implementiert werden.
Meistens werden die Spezifikation (spec) und der Package Body in getrennten Dateien gespeichert. Die Spezifikation wird in einer Datei mit der Endung `.ads` und der Body in einer Datei mit der Endung `.adb`
Der Aufbau des Pakets BruchArithmetik ist hier angegeben:

File `BruchArithmetic.ads` (Spezifikation):

```
PACKAGE BruchArithmetik IS

   TYPE BruchTyp IS RECORD
      Zaehler: Integer;
      Nenner: Positive;
   END RECORD;

   -- Restliche Spezifikationen von Funktionen, Prozeduren und
   -- Operatoren, die für die Umgebung des Pakets sichtbar sein sollen

END BruchArithmetik;
```

File `BruchArithmetik.adb` (Implementierung):

```
PACKAGE BODY BruchArithmetik IS

   -- Implementierung der Funktionen, Operatoren und Prozeduren, die
   -- für die Aufgabe benötigt werden.

END BruchArithmetik;
```

Eine Funktion, die nicht in der Spezifikation, aber für die Implementierung wichtig ist, ist die Funktion, die den größten gemeinsamen Teiler berechnet. Ein schnelles Verfahren beruht auf der Beobachtung, dass der ggT von zwei Zahlen a und b, für a > b, gleich dem ggT von b und dem Rest der Division von a und b ist. Ist a < b müssen a und b erst vertauscht werden. Damit kann das Problem auf ein kleineres Problem reduziert werden. Ist b=0, so ist a der ggT von a und b.
Zur Anschauung das folgende Beispiel:

```
ggT(84, 40) = ggT(40, 4) = ggT(4,0) = 4.
```

Das zugehörige Programm sieht wie folgt aus:

```
FUNCTION ggT(A , B : Integer) RETURN integer IS

  TauschVariable: Integer;

BEGIN
  IF (B=0) THEN
    RETURN A;
  IF (A < B) THEN
    RETURN ggT(B,A);
  ELSE
    RETURN ggT(B, A MOD B);
  END IF;
END ggT;
```

Das Hauptprogramm `TesteBruchArithmetik` importiert mit `WITH BruchArithmetik` das Paket und kann dann die exportierten Größen in der Form `BruchArithmetik.<Größenbezeichnung>` verwenden.
Wenn zusätzlich die Anweisung `USE BruchArithmetik` verwendet wird, dann können die exportierten Größen auch direkt mit ihrer `<Größenbezeichnung>` angesprochen.werden. Wenn importierte Operatoren in infix-Schreibweise verwendet werden sollen, dann ist eine solche `USE`-Anweisung erforderlich.

Für andere Größen ist es oft nützlich die qualifizierte Form `<Paketname>.<Größenbezeichnung>` zu verwenden, da dadurch häufig die Verstehbarkeit des Programmes erhöht werden kann. Dies ist insbesondere von Bedeutung, wenn mehrere Pakete importiert werden.