HUMBOLDT-UNIVERSITÄT ZU BERLIN
INSTITUT FÜR INFORMATIK

Sommersemester 2010

Prof. Dr. Holger Schlingloff
Dr. Anatolij Zubow

Übungen zu Grundlagen der Programmierung

Übungsblatt Nr. 5 (30 Punkte)

Ausgabedatum: 14. 6. 2010, 17:00 – Abgabedatum: 28. 6. 2010, 8:00

Die Lösung der Aufgaben soll in Gruppen zu je zwei Studierenden erfolgen. Die Aufgaben sind elektronisch abzugeben. Auf jedem Lösungsblatt sind Name und Matrikelnummer der beiden Gruppenmitglieder anzugeben! Von jedem Blatt werden mindestens 20% der Punkte (also 6) verlangt.

Aufgabe 1 (6 Punkte)

(a) (6 Punkte) Beweisen Sie für den abstrakten Datentyp `seq<σ>` aus der Vorlesung mit den angegebenen Gesetzen folgende Gleichungen:

```
isempty(rest(prefix(a, prefix(b, empty)))) = false
isempty(rest(rest(prefix(a, prefix(b, empty))))) = true
last(lead(postfix(a, postfix(b,empty)))) = b
```

(b) (Zusatz) Beweisen oder widerlegen Sie, dass die folgende Aussage ableitbar ist:

```
prefix(a,empty) = postfix(empty,a)
```

Aufgabe 3 (8 Punkte)

Gegeben sei der abstrakte Datentyp `Rat` für rationale Zahlen als Paare ganzer Zahlen.

```
(Rat, rat, z, n, eq, +, *)

rat: Int x Int -> Rat
z, n: Rat -> Int
eq: Rat x Rat -> Bool
+, *: Rat x Rat -> Rat

z(rat(x,y)) = x
n(rat(x,y)) = y
eq(x,y) = (z(x) * n(y) = n(x) * z(y))
x + y = rat(z(x) * n(y) + z(y) * n(x), n(x) * n(y))
x * y = rat(z(x) * z(y), n(x) * n(y))
```

(a) (5 Punkte) Welche der folgenden Gleichungen lassen sich aus diesen Gesetzen (und der Arithmetik ganzer Zahlen) ableiten? Geben Sie gegebenenfalls eine Ableitung an.

```
1.   eq(rat(2,3), rat(4,6)) = true
2.   rat(2,3) + rat(4,3) = 2
3.   eq(rat(4,1) * rat(1,2), rat(2,1)) = true
4.   eq(rat(x,y), rat(x*z, y*z)) = true
5.   eq(rat(x,y), rat(x+1,y)) = false
```

(b) (3 Punkte) Geben Sie eine dem abstrakten Datentyp `Rat` entsprechende abstrakte Java-Klasse an.

Aufgabe 4 (10 Punkte)

Eine gegebene Zeichenkette soll gespiegelt werden ("Beispiel"→"leipsieB"). Hierzu ist eine geeignete Klasse (`StrMirror`) in Java zu schreiben. Folgende Punkte sind dabei zu beachten:

- Konstruktoren: der erste Konstruktor setzt den Parameter (die zu spiegelnde Zeichenkette), wobei der zweite Konstruktor den ersten Konstruktor mit der leeren Zeichenkette aufruft (→ Standard-Konstruktor).
- Die Zeichenkette ist in einer Instanzvariablen abzulegen; der Zugriff erfolgt nicht direkt, sondern über zwei Instanzmethoden (`getString()` bzw. `setString()`). Der Wert der Zeichenkette kann nachträglich über die `setString()`-Methode verändert werden.
- Die Methode `mirror()` spiegelt die Zeichenkette unter der Verwendung eines Stapels (*Stack*) und gibt diese zurück.
- Die Methode `mirrorR()` spiegelt die Zeichenkette unter der Verwendung von Rekursion und gibt diese zurück.

Ihre Klasse sollte sich wie folgt verwenden lassen können:

```
StrMirror m = new StrMirror("Beispiel");
System.out.println(m.getString() + " --> " + m.mirror());
m.setString("Rekursion");
System.out.println(m.getString() + " --> " + m.mirrorR());
```

Hinweise

- Auf der Klasse String sind eine Reihe von nützlichen Methoden definiert wie z.B. `toCharArray()` zur Umwandlung eines Strings in ein Character-Feld (`char[]`) (siehe http://java.sun.com/j2se/1.5.0/docs/api/)

Aufgabe 5 (6 Punkte)

Gegeben Sei die folgende Klasse `TestComplex`:

```
public class TestComplex {
   public static void main(String[] args) {
      // 2 komplexe Zahlen anlegen
      Complex z1 = new Complex(5, 7);
      Complex z2 = new Complex(7, 2);
      // komplexe Zahlen ausgeben
      System.out.println("Ausgabe von z1: " + z1);
      System.out.println("Ausgabe von z2: " + z2);
      // komplexe Zahl mit einem Skalar multiplizieren + ausgeben
      Complex z3 = z1.multiplyBy(5);
      System.out.println("5 * z1: " + z3);
      // 2 komplexe Zahl miteinander multiplizieren + ausgeben
      Complex z4 = z1.multiplyBy(z2);
      System.out.println("z1 * z2: " + z4);
   }
}
```

Dieses Programm verwendet die Klasse `Complex` zur Darstellung komplexer Zahlen. Es produziert die folgende Ausgabe:

```
Ausgabe von z1: 5 + i*7
Ausgabe von z2: 7 + i*2
5 * z1: 25 + i*35
z1 * z2: 21 + i*59
```

Schreiben Sie die Klasse `Complex` mit allen notwendigen Variablen und Methoden, die dazu nötig sind, um das gegebene Programm lauffähig zu machen.