Institut für Informatik
der Universität München
Prof. Dr. M. Hofmann
Dr. J. Johannsen, Dr. R. Matthes

SS 2004

8.7.2004

Übungen zur Vorlesung
Informatik II

Blatt 11

Abgabe der Hausaufgaben bis spätestens am St. Nimmerleinstag, 14:00 Uhr über
`/dev/null`,
Bearbeitung in Gruppen zu max. 3 Personen ist zulässig.

Bitte halten Sie sich an folgende Grundsätze zur Bearbeitung der Aufgaben. Die Bewertung Ihrer Abgaben kann andernfalls deutlich herabgesetzt werden.

Verwenden Sie genau die Signatur der Aufgabenstellung, insbesondere die Namen der Klassen und Methoden in der vorgegebenen Klein- und Großschreibung.

Verwenden Sie keine „magischen" Größen in Ihren Programmen, d. h., weisen Sie Konstanten (außer den Zahlen 0 und 1 und dem leeren String) immer einer mit `final` deklarierten und in Großbuchstaben geschriebenen „Variablen" zu und verwenden Sie anschließend nur diese. Wer gute Gründe hat für eine Abweichung von dieser Regel, der kommentiere seinen Code entsprechend.

Jede Klasse und `public`-Methode muß angemessen mit `javadoc`-Kommentaren dokumentiert werden.

Programmieraufgabe P-25 (`AVLBaum.java, AVLBaumTest.java`): **0 Punkte**
Auf der Vorlesungshomepage finden Sie in der Datei `AVLBaum.java` eine unvollständige Implementierung der Einfüge- und Löschoperation auf AVL-Bäumen. Genau genommen, fehlt in der Klasse `AVLBaum` ausschließlich die Implementierung von

```
public AVLBaum insert_avl(Comparable x)
```

Diese Methode soll leisten, was in der Vorlesung dazu beschrieben ist. Dies soll allerdings so imperativ wie möglich mithilfe einer passenden Implementierung des Interfaces `AVLBaumVisitorI` geschehen. Die berechnete Höhendifferenz soll nicht Rückgabewert sein, sondern braucht nur in einer passenden Instanzvariablen des Visitors gehalten zu werden. Halten Sie sich ansonsten genau an den algorithmischen Ablauf, der (funktional) auf den Vorlesungsfolien beschrieben ist.

Schreiben Sie sinnvolle Tests, die Suchbäume mit mindestens 100000 Elementen zufällig und/oder deterministisch erzeugen und dann deren Suchbaumeigenschaft oder Balanciertheit mit den `AVLBaum`-Methoden `istSuchbaum` bzw. `istBalanciert` überprüfen. Derart große Bäume kann man nur schlecht ausgeben. Entfernen Sie daher mittels `remove_max_avl` oder `delete_avl` alle Elemente und überzeugen Sie sich, daß dann der leere Baum verbleibt.

– bitte wenden –

Hinweis: Der Baum soll nicht jedesmal neu aufgebaut werden, also soweit wie möglich auf die Verwendung der Konstruktoren `Empty` und `Build` verzichtet werden. Beachten Sie, daß `insert_avl` einen Baum zurückliefert, während der Baum, für den die Methode aufgerufen wurde, nicht mehr zu verwenden ist. Um sich selbst vor Irrtümern zu schützen, empfiehlt es sich zumeist, gleich Anweisungen z. B. der Form

```
t = t.insert_avl(new Integer(0));
```

zu verwenden. Das gilt sinngemäß ebenso für die anderen Operationen auf AVL-Bäumen, nicht aber für `void rotate()`, das wirklich an Ort und Stelle umdreht.

Programmieraufgabe P-26 (`museum.jar`): **0 Punkte**

Schreiben Sie in einer Klasse `Museum` eine Simulation des Verleihs für Audioguides in einem Museum. In zufällig gewählten Abständen von bis zu 10 Sekunden erscheinen Besucher, die Audioguides ausleihen wollen. Es stehen 7 Audioguides zur Verfügung. Wenn ein Besucher kommt und kein Guide mehr da ist, muss er warten, bis wieder einer zur Verfügung steht. Erhält ein Besucher einen Audioguide, bleibt er für einen zufälligen Zeitraum von bis zu 60 Sekunden im Museum, anschliessend gibt er seinen Guide zurück.

Modellieren Sie jeden Besucher durch einen eigenen Thread der Klasse `Besucher`. Das Ausleihen und Zurückgeben von Audioguides ist durch synchronisierte Methoden eines Objektes der Klasse `Verleih` zu realisieren. Beim Ausleihen und Zurückgeben sollten entsprechende Meldungen ausgegeben werden, so dass eine Ausgabe ähnlich der auf der Vorlesungshomepage zu findenden Datei `ausgabe.txt` erzeugt wird.