Universität Stuttgart

Institut für Visualisierung
und Interaktive Systeme

Übungsblatt 3

Datenstrukturen und Algorithmen (SS 2018)

Abgabe: Mittwoch, 16.05.2018, 23:55 Uhr — Besprechung: ab Montag, 28.05.2018

Bitte lösen Sie die Übungsaufgabe in **Gruppen von 3 Studenten** und wählen EINEN Studenten aus, welcher die Lösung im ILIAS als **PDF** als **Gruppenabgabe** (unter Angabe aller Gruppenmitglieder) einstellt. Bitte erstellen Sie dazu ein **Titelblatt**, welches die Namen der Studenten, die Matrikelnummern, und die E-Mail-Adressen enthält.

Die Aufgaben mit Implementierung sind mit Impl gekennzeichnet. Das entsprechende Eclipse-Projekt kann im ILIAS heruntergeladen werden. Bitte beachten Sie die Hinweise zu den Implementierungsaufgaben, die im ILIAS verfügbar sind.[1]

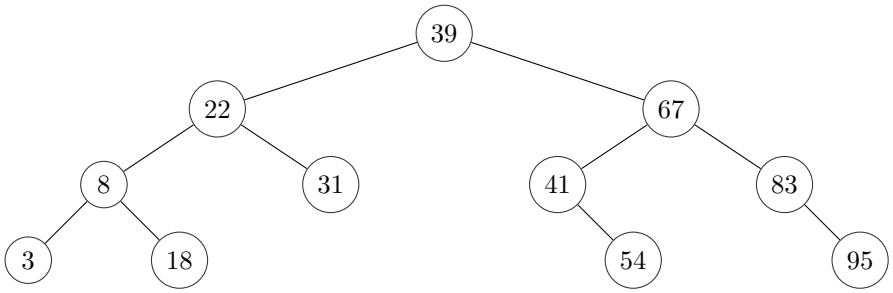
Dieses Übungsblatt beinhaltet 4 Aufgaben mit einer Gesamtzahl von 30 Punkten.

Aufgabe 1 Impl Iterator *[Punkte: 10]*

Gegeben im Eclipse-Projekt sind die Schnittstellen `ISimpleList` und `ISimpleListIterable`. Erweitern Sie die Klasse `SimpleList` (ohne die bestehenden Methoden zu modifizieren), damit sie die Schnittstelle `ISimpleListIterable` implementiert. Erstellen Sie hierzu zwei Iterator-Klassen als innere Klassen der Klasse `SimpleList`. **Die Verwendung existierender Iteratorimplementierungen (z.B. `ArrayList.iterator()` sowie Iteratoren anderer Datenstrukturen aus der Java-Klassenbibliothek) ist dabei nicht erlaubt.** Die Methode `remove` der Iteratoren wird nicht unterstützt und sollte immer `UnsupportedOperationException` werfen. Sie können davon ausgehen, dass die Liste während der Verwendung des Iterators nicht verändert wird.

Aufgabe 2 Binäre Suchbäume, AVL-Bäume *[Punkte: 6]*

Gegeben sei der binäre Suchbaum B_1:



(a) *(1 Punkte)* Erstellen Sie den Baum B_2, indem Sie in B_1 die Werte *14* und *75* einfügen.

(b) *(1 Punkte)* Erstellen Sie den Baum B_3, indem Sie aus dem ursprünglichen Baum B_1 die Werte *18* und *67* entfernen. Ersetzen Sie, wo nötig, Knoten durch ihre Inorder-**Vorgänger**.

(c) *(1 Punkte)* Geben Sie für jeden Knoten aus dem ursprünglichen Baum B_1 den Wert der AVL-Balance an.

(d) *(3 Punkte)* Erstellen Sie den Baum B_4, indem Sie in den *ursprünglichen* Baum B_1 den Wert *38* einfügen. Erstellen Sie den Baum B_5, indem Sie in den *ursprünglichen* Baum B_1 den Wert *96* einfügen. Erstellen Sie den Baum B_6, indem Sie in den *ursprünglichen* Baum B_1 den Wert *15* einfügen. Führen Sie für jeden Baum (B_4, B_5, B_6) nach dem Einfügen etwaige Schritte durch, um die AVL-Balance wiederherzustellen. Geben Sie dazu auch an, welche Schritte nötig waren, um die AVL-Balance wiederherzustellen.

[1] `https://ilias3.uni-stuttgart.de/goto_Uni_Stuttgart_fold_1442448.html`

Aufgabe 3 Impl Binäre Suchbäume *[Punkte: 10]*

Gegeben im Eclipse-Projekt sind die Schnittstellen `IBinaryTreeNode` und `IBinarySearchTree` für binäre Suchbäume.

(a) *(2 Punkte)* Implementieren Sie die Knotenklasse `BinaryTreeNode`, die die Schnittstelle `IBinaryTreeNode` implementiert. Die Knotenklasse muss einen Standard-Konstruktor (d.h. Konstruktor ohne Parameter) enthalten.

(b) *(8 Punkte)* Implementieren Sie die binäre Suchbaum-Klasse `BinarySearchTree`, die die Schnittstelle `IBinarySearchTree` implementiert. **Ignorieren Sie dabei Duplikate, d.h. wenn ein Schlüsselwert eingefügt werden soll, welcher im Baum bereits vorhanden ist, dann soll nichts passieren bzw. der Versuch des Einfügens ignoriert werden.** Die Suchbaum-Klasse muss einen Standard-Konstruktor (d.h. Konstruktor ohne Parameter) enthalten.

Aufgabe 4 Binäre Suchbäume *[Punkte: 4]*

Gegeben ist die folgende Menge von Schlüsselwerten: 47, 6, 71, 57, 12, 7, 19, 63, 11, 8, 56, 5, 28, 68, 14, 32, 99, 21, 15, 3. Zeichnen Sie den dazugehörigen *Binären Suchbaum*, so dass dessen *Preorder Traversierung* mit der Sequenz 21, 11, 6, 3, 5, 8, 7, 15, 14, 12, 19 *beginnt* und dessen *Postorder Traversierung* mit der Sequenz 28, 47, 32, 57, 68, 63, 99, 71, 56, 21 *endet*.