14.Übung zu Programmierung 2 (binäre Suchbäume)

Hintergrund:

In vielen realen Anwendungsfällen besteht für eine Menge von Items irgendeines Typs (z.B. Personen) der Bedarf, sie sowohl mit Operationen zu bearbeiten, die eine sortierte Datenhaltung nahe legen, als auch Operationen, die dies nicht erfordern. Deshalb kann es sinnvoll sein, die Items in einem unsortierten Array zu halten und nur die Sortierreihenfolge in einem Baum mit Verweisen auf das jeweilige Arrayelement.

Weitere Details und Beispiele finden Sie in der Powerpointpräsentation.

Bsp.: Eine Eingabefolge der Personen mit Namen "guettler", "folz", "kretschmer", "denzer", "grabowski", "schmidt" würde zu folgender Konstellation führen:

Nr	Name	Vorlesung	Gehalt
0	guettler	Prog.Spr. 2	4000,00
1	folz	C++	3950,00
2	kretschmer	XML	3900,00
3	denzer	Computergraphik	4100,00
4	schmidt	ADV	7420,00
5	grabowski	Mathematik	3850,00
•	grabowski	Mathematik	3850,00

Um bei "folz" die Vorlesung zu ändern, erfolgt der Zugriff auf das Item effizient über eine Suche im Baum.

Zur Ermittlung des Durchschnittsgehalts erfolgt der Zugriff sequentiell über alle Items des Arrays.

1. Aufgabe

Implementieren Sie diese Datenstruktur zunächst unter der Voraussetzung, dass keine Löschoperationen vorkommen.

Folge: Alle eingefügten Knoten werden jeweils hinter den letzten im Array abgelegt. Im Baum wird die richtige Stelle gesucht und "normal" eingefügt. Überlegen Sie sich geeignete Definitionen für die Klassen.

Bemerkung: Unter diesen Voraussetzungen wäre die "contigous sequential representation" aus der Vorlesung natürlich besonders effizient. Aufgabe 1 soll aber auf Aufgabe 2 vorbereiten.

Eingabe siehe → Allgemeine Hinweise zur Programmierung

2. Aufgabe

Erweitern Sie ihre Lösung von Aufgabe 1 um die "delete"-Operation. Unter der Voraussetzung, dass man mit einem möglichst kleinen Array auskommen möchte, erfordert dies eine geeignete Freispeicherverwaltung im Array, d.h. bei einem "insert" sollen jeweils durch "delete" entstandene Lücken im Array wieder aufgefüllt werden. Dies ist möglich, da die Einträge im Array ja nicht sortiert zu sein brauchen. Es erfordert aber, dass man zu jedem Zeitpunkt einen Überblick über alle Lücken hat (von denen aber nur die erste gebraucht wird).

Eingabe siehe → Allgemeine Hinweise zur Programmierung

Allgemeine Hinweise zur Programmierung

- 1) Die Eingabedaten (Strings) stehen in einer Eingabedatei (Name der Datei ist einzulesen!) jeweils mit einem vorangestellten "+" für *insert* oder einem "-" für *delete*. Beschränken Sie die Eingabe auf "Name", die weiteren Angaben "Vorlesung" und "Gehalt" sind zum Testen nicht verlangt!
- 2) Die Eingabe-Datei ist auf "existiert", ist "einfache Datei" und ist "lesbar" mittels selbst geschriebener Exceptions abzuprüfen. Im Fehlerfall ist adäquat zu reagieren.
- 3) Als Eingabedatei-Reader ist der LineNumberReader zu nutzen.
- 4) Es soll nach Lesen einer Eingabezeile deren Nummer und Inhalt auf Stdout ausgegeben werden, und nach deren Verarbeitung der momentane Zustand der entsprechenden Datenstrukturen (Array, binärer Baum).
- 5) Fehlerhafte Eingabezeilen oder zu fehlerhaften bzw. verbotenen Aktionen führende Eingabezeilen sind mittels eigens geschriebener Exceptions dem Testprogramm zu melden und auf der Stdout mit Zeilennummer zu dokumentieren.
- 6) Benutzen Sie bei den Methodenschnittstellen als Übergabeparameter, wie bei den Datenstrukturen im Skript, Object-Referenzen.
- 7) Bedenken Sie, dass auch in diesem Falle nur vergleichbare Objekte in die Datenstruktur aufgenommen werden dürfen.

<u>Abgabe</u>

Folgen Sie dem Link "*Praktikum Programmierung 2*" auf der Homepage des Systemtechniklabors. Dort ist der Abgabetermin für diese Übung angegeben.