Rekursion 4AHIF

Teil 1: Binäre Bäume

Es dürfen im Baum und in der Node nur REKURSIVE Lösungen programmiert werden. KEINE iterativen! Alle Methoden müssen im Main aufgerufen werden!

Kopiere vom Angabeverzeichnis das Java-Projekt BinaryTree auf Dein T-Laufwerk und öffne es.

1. Erweitere die Node-Klasse um einen Zähler. Falls ein Wert öfters als einmal in den Baum eingefügt wird, ist der Zähler entsprechend zu erhöhen.
Teste den Baum für: *6, 4, 7, 6, 3, 5, 9, 3, 8*

2. Erstelle eine Methode `printlnDesc()`, die alle Nodes **absteigend** sortiert auf der Konsole ausgibt.
Hinweis: In jedem Node müssen zunächst die größeren Werten ausgegeben werden und am Schluß die kleineren Werte. Kontrollwerte: *9, 8, 7, 6, 6, 5, 4, 3, 3*

3. Erstelle eine Methode `printlnPreOrder(double threshold1, double threshold2)`, die alle jene Nodes pre-order auf der Konsole ausgibt, deren Wert im Zahleninterval [threshold1, threshold2] liegt. Rufe die Methode für *threshold1 = 3,5* und *threshold2 = 7,3* auf und schreib das Resultat in ein Kommentar.

6. Erstelle eine Methode, die eine absteigend sortierte `List<Integer>` all jener Werte zurückgibt, die **kleiner als ein bestimmter Threshold-Wert** sind.
z. B. tree1.getListLessThan(10) -> 6, 9, 8 in einer Liste

7. Erstelle eine rekursive Methode, die **den höchsten Wert eines Blattes** des Baumes zurückgibt.

Teil 2: Rekursion – Ackermann-Funktion

Für die zwei Ganzzahlen m und n ist die Ackermannfunktion folgendermaßen definiert:

$$A(m, n) = \begin{cases} n + 1 & \text{für } m = 0 \\ A(m - 1, 1) & \text{für } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{für } m > 0 \text{ und } n > 0 \end{cases}$$

1. Lege eine Klasse an und implementiere die Ackermannfunktion rekursiv in einer nicht statischen Methode.

2. Erweitere die Konsolenanwendung um die Möglichkeit m und n als Command Line Parameter zu übergeben. Falls der Konsolenanwendung keine Argumente übergeben werden, sollen die Werte über die Konsoleneingabe eingelesen werden.
Berechne die Werte A(m, n) und schreibe die Ergebnisse in Codekommentare neben die Aufrufe
   1) m = 1, n = 1
   2) m = 3, n = 4
   3) m = 4, n = 1

3. Sorge dafür, dass für eine bestimmte Kombination von m und n die Ackermannfunktion nur einmal berechnet wird. Implementiere also einen **Cache**.
Füge dazu in der Klasse eine passende Collection als Instanzvariable hinzu.
Speichere in dieser Collection für die Inputparameter m und n das Ergebnis der Ackermann-Funktion.
Überprüfe am Beginn der Methode ob der Wert für die Inputwerte von m und n bereits berechnet wurde und verwende diesen gespeicherten Wert.