Programmieren in Java

`http://proglang.informatik.uni-freiburg.de/teaching/java/2017/`

search-tree

Suchbäume

Woche 08 Aufgabe 1/3

Herausgabe: 2017-06-19
Abgabe: 2017-06-30

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project `search-tree`
Package `searchtree`
Klassen

Tree
`public Tree add(int i)`
`public boolean contains(int i)`
`public int size()`
`public String elementsAsString()`

Trees
`public static Tree makeTree(int[] elements)`

Implementieren Sie einen binären Suchbaum für Integer. Zur Erinnerung eine Definition (angepasst aus dem Wikipedia-Artikel für Suchbäume):

> In der Informatik ist ein binärer Suchbaum eine Kombination der abstrakten Datenstrukturen Suchbaum und Binärbaum. Ein binärer Suchbaum, häufig abgekürzt als BST (von englisch Binary Search Tree), ist ein binärer Baum, bei dem die Knoten „Schlüssel“ tragen, und die Schlüssel des linken Teilbaums eines Knotens **nur kleiner** und die des rechten Teilbaums **nur größer** als der Schlüssel des Knotens selbst sind.

Der Suchbaum hat das Interface *`Tree`* und unterstützt folgende Methoden:

- `add(i)`: fügt das Element `i` dem Baum hinzu
- `contains(i)`: gibt `true` zurück, genau dann wenn `i` im Baum enthalten ist
- `size()`: gibt die Größe des Suchbaums zurück, d.h. die Anzahl der in ihm enthaltenen Elemente
- `elementsAsString()`: gibt die Elemente des Baums als String aus. Die Elemente sollen durch Kommata (`", "`) getrennt werden und sortiert sein (siehe auch Beispieltests unten).

Implementieren Sie den Suchbaum durch rekursive Klassen. Die Klassen sollen **immutable** sein, das heißt, die `add` Methode soll den Baum, auf dem Sie aufgerufen wird, nicht verändern.

Achtung: bei dieser Aufgabe ist das Verwenden von Collections bei der Implementierung des Baumes verboten, ebenso wie das Verwenden von anderen Klassen oder Funktionen aus `java.util.*`. Verwenden Sie selbstgeschriebene Klassen, ansonsten gibt es keine Punkte.

Implementieren Sie weiterhin die Funktion `makeTree` in der Klasse `Trees`, die ein Array von Integern in einen Suchbaum einfügt und diesen zurück gibt.

Beispieltests

```
package searchtree;

import org.junit.Test;


import static org.junit.Assert.*;

public class TestExamples {

    @Test
    public void exampleTests() {
        Tree t = Trees.makeTree(new int[]{2, 3, 4, 4, 1});
        assertTrue(t.contains(4));
        assertFalse(t.contains(6));
        assertEquals(4, t.size());

        Tree t2 = t.add(6).add(7).add(6);
        assertFalse(t.contains(6));
        assertTrue(t2.contains(6));
        assertEquals(6, t2.size());
        assertEquals(4, t.size());

        assertEquals("1, 2, 3, 4, 6, 7", t2.elementsAsString());

        Tree empty = Trees.makeTree(new int[]{});
        assertEquals(0, empty.size());
        assertEquals("", empty.elementsAsString());
        assertFalse(empty.contains(0));
    }

}
```