
Programmieren in Java<http://proglang.informatik.uni-freiburg.de/teaching/java/2017/>

generic-tree*Generische Suchbäume*

Woche 09 Aufgabe 3/3

Herausgabe: 2017-06-26

Abgabe: 2017-07-07

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

Project **generic-tree**
Package **generictree**
Klassen (siehe Beispiele in Testfällen unten)

In dieser Aufgabe sollen Sie die Klassen und das Interface für Suchbäume für `ints` aus `w08/search-tree` erweitern, so dass Sie *generische* Suchbäume implementieren: Anstatt sich auf den Typ `int` zu beschränken, soll die generische Implementierung das Erstellen und Manipulieren von Suchbäume für allen Typen `T`, die das Interface `Comparable<T>` implementieren, erlauben.

Die folgenden Testfälle, welche auch im Skelett dieser Aufgabe enthalten sind, zeigen, wie die generischen Suchbaumklassen für `Integer`, `String`, und `Character` Elemente funktionieren sollen. (Die Klassen `Integer`, `String`, und `Character` aus der Java-API implementieren alle ihr entsprechendes `Comparable` Interface.)

```
package generictree;

import org.junit.Test;

import static org.junit.Assert.*;

public class TestExamples {

    @Test
    public void testInts1() {
        Tree<Integer> t = Trees.makeTree(new Integer[]{2, 3, 4, 4, 1});
        assertTrue(t.contains(4));
        assertFalse(t.contains(6));
        assertEquals(4, t.size());
    }

    @Test
    public void testStrings() {
        Tree<String> t = Trees.makeTree(new String[]{"abc", "abcd",
                                                    "xy", "xy", "z"});
        Tree<String> t2 = t.add("hi").add("world").add("hi");
        assertFalse(t.contains("hi"));
        assertTrue(t2.contains("hi"));
        assertEquals(6, t2.size());
        assertEquals(4, t.size());
        assertEquals("abc, abcd, hi, world, xy, z",
                     t2.elementsAsString());
    }
}
```

```

    }

    @Test
    public void testChars() {
        Tree<Character> t = Trees.makeTree(new Character[]{'2', '3',
                                                         '4', '4', '1'});

        Tree<Character> 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());
    }
}

```

Hinweise: Zur Lösung der Aufgabe müssen Sie *generische Klassen und Methoden* mit *unteren Typschranken* verwenden. Näheres dazu erfahren Sie in der Vorlesung und in den folgenden Tutorials:

<https://docs.oracle.com/javase/tutorial/java/generics/types.html>

<https://docs.oracle.com/javase/tutorial/java/generics/methods.html>

<https://docs.oracle.com/javase/tutorial/java/generics/bounded.html>

<https://docs.oracle.com/javase/tutorial/java/generics/boundedTypeParams.html>

Das Comparable Interface ist in der Java-API Dokumentation beschrieben:

<https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html>