

```
1 import static org.junit.Assert.assertEquals;
2
3 /**
4  * JUnit test fixture for {@code BinarySearchTreeMethods}'s static methods
5  * isInTree (and removeSmallest).
6  */
7
8 public final class BinarySearchTreeMethodsTest {
9
10     /**
11      * Constructs and return a BST created by inserting the given {@code args}
12      * into an empty tree in the order in which they are provided.
13      *
14      * @param args
15      *     the {@code String}s to be inserted in the tree
16      * @return the BST with the given {@code String}s
17      * @requires [the Strings in args are all distinct]
18      * @ensures createBSTFromArgs = [the BST with the given Strings]
19      */
20     private static BinaryTree<String> createBSTFromArgs(String... args) {
21         BinaryTree<String> t = new BinaryTree<String>();
22         for (String s : args) {
23             BinaryTreeUtility.insertInTree(t, s);
24         }
25         return t;
26     }
27
28     @Test
29     public void inTreeTest1() {
30         /*
31          * Set up variables
32          */
33         BinaryTree<String> t1 = createBSTFromArgs("b", "a", "c");
34         BinaryTree<String> t2 = createBSTFromArgs("b", "a", "c");
35         /*
36          * Call method under test
37          */
38         boolean inTree = BinarySearchTreeMethods.isInTree(t1, "a");
39         /*
40          * Assert that values of variables match expectations
41          */
42         assertEquals(true, inTree);
43         assertEquals(t2, t1);
44     }
45
46     @Test
47     public void inTreeTest2() {
48         /*
49          * Set up variables
50          */
51         BinaryTree<String> t1 = createBSTFromArgs("b", "a", "c");
52         BinaryTree<String> t2 = createBSTFromArgs("b", "a", "c");
53         /*
54          * Call method under test
55          */
56         boolean inTree = BinarySearchTreeMethods.isInTree(t1, "b");
57         /*
58          * Assert that values of variables match expectations
59          */
60         assertEquals(true, inTree);
61     }
62 }
```

```
65         assertEquals(t2, t1);
66     }
67
68     @Test
69     public void inTreeTest3() {
70         /*
71          * Set up variables
72          */
73         BinaryTree<String> t1 = createBSTFromArgs("b", "a", "c");
74         BinaryTree<String> t2 = createBSTFromArgs("b", "a", "c");
75         /*
76          * Call method under test
77          */
78         boolean inTree = BinarySearchTreeMethods.isInTree(t1, "c");
79         /*
80          * Assert that values of variables match expectations
81          */
82         assertEquals(true, inTree);
83         assertEquals(t2, t1);
84     }
85
86     @Test
87     public void inTreeTest4() {
88         /*
89          * Set up variables
90          */
91         BinaryTree<String> t1 = createBSTFromArgs("b", "a", "c");
92         BinaryTree<String> t2 = createBSTFromArgs("b", "a", "c");
93         /*
94          * Call method under test
95          */
96         boolean inTree = BinarySearchTreeMethods.isInTree(t1, "d");
97         /*
98          * Assert that values of variables match expectations
99          */
100        assertEquals(false, inTree);
101        assertEquals(t2, t1);
102    }
103
104    @Test
105    public void inTreeTest5() {
106        /*
107         * Set up variables
108         */
109        BinaryTree<String> t1 = createBSTFromArgs("b");
110        BinaryTree<String> t2 = createBSTFromArgs("b");
111        /*
112         * Call method under test
113         */
114        boolean inTree = BinarySearchTreeMethods.isInTree(t1, "b");
115        /*
116         * Assert that values of variables match expectations
117         */
118        assertEquals(true, inTree);
119        assertEquals(t2, t1);
120    }
121
122    @Test
123    public void inTreeTest6() {
```

```
124      /*
125       * Set up variables
126       */
127      BinaryTree<String> t1 = createBSTFromArgs("b");
128      BinaryTree<String> t2 = createBSTFromArgs("b");
129      /*
130       * Call method under test
131       */
132      boolean inTree = BinarySearchTreeMethods.isInTree(t1, "a");
133      /*
134       * Assert that values of variables match expectations
135       */
136      assertEquals(false, inTree);
137      assertEquals(t2, t1);
138    }
139
140    // TODO: add here other test cases for BinarySearchTreeMethods.isInTree
141    // (and for BinarySearchTreeMethods.removeSmallest)
142
143 }
144
```