```java
package proj5;
import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.Timeout;
import static org.junit.Assert.*;

/**
 * test the functionality of BinarySearchTree.java
 * author: Son Nguyen (Kyrie)
 * version: 6/3/2020
 */
public class BSTTest {

    @Rule // a test will fail if it takes longer than 1/10 of a second to run
    public Timeout timeout = Timeout.millis(100);

    private BinarySearchTree<String> empty;

    @Before
    public void setUp() throws Exception {
        empty = new BinarySearchTree<String>();
    }

    @After
    public void tearDown() throws Exception {
        empty = null;
    }

    private BinarySearchTree<Integer> makeSampleBST() {
        BinarySearchTree<Integer> sample = new BinarySearchTree<Integer>();
        int[] sampleNodes = new int[]{270, 151, 333, 120, 250, 329, 370, 105, 234, 260, 320, 499};
        for (int sampleNode: sampleNodes) {
            sample.insert(sampleNode);
        }
        return sample;
    }

    @Test // string representation of a binary search
```

```
40  tree
41      public void testToString() {
42          assertEquals("", empty.toString());
43
44          BinarySearchTree<Integer> sample =
    makeSampleBST();
45          String expected = "
    10512015123425026027032032933370499";
46          assertEquals(expected, sample.toString());
47      }
48
49      @Test // insert a new item to a binary search
    tree
50      public void testInsert() {
51          empty.insert("Delete");
52          empty.insert("Insert");
53          empty.insert("Append");
54          String expected1 = "AppendDeleteInsert";
55          assertEquals(expected1, empty.toString());
56
57          BinarySearchTree<Integer> sample =
    makeSampleBST();
58          sample.insert(380);
59          String expected = "
    105120151234250260270320329333370380499";
60          assertEquals(expected, sample.toString());
61      }
62
63      @Test // delete an item in a binary search tree
64      public void testDelete() {
65          empty.delete("Anything");
66          assertEquals("",empty.toString());
67          empty.insert("Delete");
68          empty.insert("Insert");
69          empty.insert("Append");
70          empty.delete("Delete");
71          String expected1 = "AppendInsert";
72          assertEquals(expected1, empty.toString());
73
74          BinarySearchTree<Integer> sample =
    makeSampleBST();
75          String original = sample.toString();
76          sample.delete(380);
77          assertEquals(original, sample.toString());
```

```
 78          sample.delete(105);
 79          String expected2 = "
    120151234250260270320329333370499";
 80          assertEquals(expected2, sample.toString());
 81          sample.delete(333);
 82          String expected3 = "
    120151234250260270320329370499";
 83          assertEquals(expected3, sample.toString());
 84      }
 85
 86      @Test // search an item in a binary search tree
 87      public void testSearch() {
 88          assertNull(empty.search("Anything"));
 89          empty.insert("Delete");
 90          empty.insert("Insert");
 91          empty.insert("Append");
 92          empty.delete("Delete");
 93          assertEquals("Insert", empty.search("Insert"
    ));
 94
 95          BinarySearchTree<Integer> sample =
    makeSampleBST();
 96          assertNull(sample.search(380));
 97          assertEquals((Integer) 260, sample.search(
    260));
 98          assertEquals((Integer) 151, sample.search(
    151));
 99      }
100 }
101
```