```
1 import components.binarytree.BinaryTree;
 2 import components.binarytree.BinaryTree1;
 4 /**
 5 * {@code Queue} represented as a {@code Sequence} of entries, with
 6 * implementations of primary methods.
 7 *
8 * @param <T>
9 *
                 type of {@code Queue} entries
10 * @correspondence this = $this.entries
11 */
12 public class HelloWorld {
13
14
      /**
15
       * Returns the size of the given {@code BinaryTree<T>}.
16
17
       * @param <T>
18
                     the type of the {@code BinaryTree} node labels
19
       * @param t
                     the {@code BinaryTree} whose size to return
20
21
       * @return the size of the given {@code BinaryTree}
22
       * @ensures size = |t|
23
24
      public static <T> int size(BinaryTree<T> t) {
25
          int count = 1;
26
27
          if (t.height() > 0) {
28
               BinaryTree<T> left = new BinaryTree1<>();
29
               BinaryTree<T> right = new BinaryTree1<>();
              T root = t.disassemble(left, right);
30
               int leftCount = size(left);
31
               int rightCount = size(right);
32
33
               count += leftCount + rightCount;
34
               t.assemble(root, left, right);
35
          }
36
37
          return count:
38
      }
39
40
41
       * Returns the size of the given {@code BinaryTree<T>}.
42
43
       * @param <T>
44
                     the type of the {@code BinaryTree} node labels
```

```
Tuesday, February 8, 2022, 9:39 AM
HelloWorld.java
45
       * @param t
                    the {@code BinaryTree} whose size to return
46
       * @return the size of the given {@code BinaryTree}
47
48
       * @ensures size = |t|
49
      public static <T> int size(BinaryTree<T> t) {
50
51
          int count = 1;
52
          for (T x : t) {
53
54
              count++;
          }
55
56
57
          return count;
      }
58
59
60 }
```