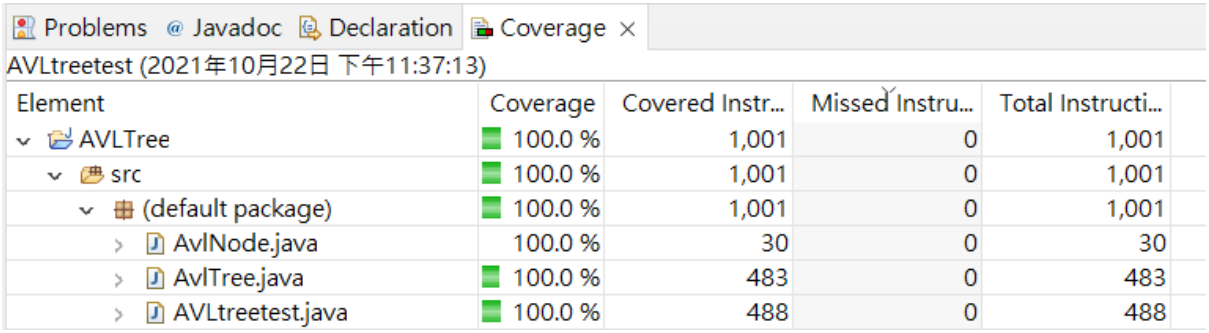
Homework5

Code coverage



Problems Javadoc Declaration Coverage ×

AVLtreetest (2021年10月22日 下午11:37:13)

Element	Coverage	Covered Instr...	Missed Instru...	Total Instructi...
AVLTree	100.0 %	1,001	0	1,001
src	100.0 %	1,001	0	1,001
(default package)	100.0 %	1,001	0	1,001
AvlNode.java	100.0 %	30	0	30
AvlTree.java	100.0 %	483	0	483
AVLtreetest.java	100.0 %	488	0	488

insert()

```
40      /* Function to insert data recursively */
41      private AvlNode insert(int x, AvlNode t) {
42          if (t == null)
43              t = new AvlNode(x);
44          else if (x < t.data) {
45              t.left = insert(x, t.left);
46              if (height(t.left) - height(t.right) == 2)
47                  if (x < t.left.data)
48                      t = rotateWithLeftChild(t);
49                  else
50                      t = doubleWithLeftChild(t);
51          } else if (x > t.data) {
52              t.right = insert(x, t.right);
53              if (height(t.right) - height(t.left) == 2)
54                  if (x > t.right.data)
55                      t = rotateWithRightChild(t);
56                  else
57                      t = doubleWithRightChild(t);
58          } else
59              ; // Duplicate; do nothing
60          t.height = max(height(t.left), height(t.right)) + 1;
61          return t;
62      }
```

1. 42行的branch由 `Insert_ToNullTree_BalanceIs0_byCoverage()` cover
2. 44~47行由 `Insert_LLrotation_BalanceIs0_byCoverageAndPartition()` cover

```
    10
    /
   4
  /
 2
```

3. 44~46、49~50行由 `Insert_LRrotation_BalanceIs0_byCoverageAndPartition()` cover

```
    10
    /
   2
    \
     4
```

4. 51~54行由 `Insert_RLrotation_BalanceIs0_byCoverageAndPartition()` cover

```
    10
     \
     15
     /
    4
```

5. 51~53、56~57由 `Insert_RRrotation_BalanceIs0_byCoverageAndPartition()` cover

```
    10
     \
     15
      \
       16
```

inorder()、preorder()、postorder()

```
145    private String inorder(AvlNode r) {
146        if (r != null) {
147            String topRes = inorder(r.left);
148            String bottomRes = inorder(r.right);
149            String retRes = (topRes.isEmpty() ? "" : (topRes + " ")) +
150                    r.data +
151                    (bottomRes.isEmpty() ? "" : (" " + bottomRes));
152
153            return retRes;
154        }
155        return "";
156    }
```

```
158     /* Function for preorder traversal */
159     public String preorder() {
160         return preorder(root);
161     }
162
163     private String preorder(AvlNode r) {
164         if (r != null) {
165             String topRes = preorder(r.left);
166             String bottomRes = preorder(r.right);
167             String retRes = r.data +
168                     (topRes.isEmpty() ? "" : (" " + topRes)) +
169                     (bottomRes.isEmpty() ? "" : (" " + bottomRes));
170
171             return retRes;
172         }
173         return "";
174     }

176     /* Function for postorder traversal */
177     public String postorder() {
178         return postorder(root);
179     }
180
181     private String postorder(AvlNode r) {
182         if (r != null) {
183             String topRes = postorder(r.left);
184             String bottomRes = postorder(r.right);
185             String retRes = (topRes.isEmpty() ? "" : (topRes + " ")) +
186                     (bottomRes.isEmpty() ? "" : (bottomRes + " ")) +
187                     r.data;
188
189             return retRes;
190         }
191         return "";
192     }
193 }
```

三個方法都是分別排序空樹和非空樹完成 coverage

search()

```
123     private boolean search(AvlNode r, int val) {
124         boolean found = false;
125         while ((r != null) && !found) {
126             int rval = r.data;
127             if (val < rval)
128                 r = r.left;
129             else if (val > rval)
130                 r = r.right;
131             else {
132                 found = true;
133                 break;
134             }
135             found = search(r, val);
136         }
137         return found;
138     }
```

分別搜尋比root大、比root小、等於root完成 coverage

countNodes()

```
107     private int countNodes(AvlNode r) {
108         if (r == null)
109             return 0;
110         else {
111             int l = 1;
112             l += countNodes(r.left);
113             l += countNodes(r.right);
114             return l;
115         }
116     }
```

計算空樹及非空樹的節點數量，完成 coverage