

30.9.2024 18:01:29

## VectorTreeTest.java

Page 1/3

```

1  /*
2  * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3  * Version: Mon Sep 30 18:01:29 CEST 2024
4  */
5
6  package uebung03.as.aufgabe02;
7
8
9  public class VectorTreeTest {
10
11     public static void main(String[] args) throws NoSuchElementException {
12
13         VectorTree<Character> vt = new VectorTree<>();
14
15         vt.printVector("Empty tree:");
16
17         if (vt.size() != 0) {
18             throw new Error("Bad size: " + vt.size() + " != 0");
19         }
20         if (vt.root() != null) {
21             throw new Error("vt.root() != null");
22         }
23
24         Character a = 'A';
25         vt.setRoot(a);
26         vt.printVector("Setting root with 'A':");
27         if (vt.size() != 1) {
28             throw new Error("Bad size: " + vt.size() + " != 1");
29         }
30         if (!vt.isRoot(a)) {
31             throw new Error("!vt.root(a)");
32         }
33         if (!vt.root().equals(a)) {
34             throw new Error("!vt.root().equals(a) : " + vt.root());
35         }
36         if (!vt.isExternal(a)) {
37             throw new Error("!vt.isExternal(a)");
38         }
39         if (vt.parent(a) != null) {
40             throw new Error("vt.parent(a) != null");
41         }
42
43         Character d = 'D';
44         vt.setRightChild(vt.root(), d);
45         vt.printVector("Setting right child of 'A' with 'D':");
46         if (vt.size() != 2) {
47             throw new Error("Bad size: " + vt.size() + " != 2");
48         }
49         if (!vt.rightChild(vt.root()).equals(d)) {
50             throw new Error("!vt.rightChild(vt.root()).equals(d) : "
51                 + vt.rightChild(vt.root()));
52         }
53         if (!vt.isExternal(d)) {
54             throw new Error("!vt.isExternal(d)");
55         }
56         if (!vt.isInternal(vt.root())) {
57             throw new Error("!vt.isInternal(vt.root())");
58         }
59         if (!vt.parent(d).equals(a)) {
60             throw new Error("!vt.parent(d).equals(a)");
61         }
62
63         Character b = 'B';
64         vt.setLeftChild(vt.root(), b);
65         vt.printVector("Setting left child of 'A' with 'B':");
66         if (vt.size() != 3) {
67             throw new Error("Bad size: " + vt.size() + " != 3");
68         }
69
70

```

30.9.2024 18:01:29

## VectorTreeTest.java

Page 2/3

```

70
71     Character f = 'F';
72     vt.setRightChild(b, f);
73     vt.printVector("Setting right child of 'B' with 'F':");
74
75     Character h = 'H';
76     vt.setRightChild(f, h);
77     vt.printVector("Setting right child of 'F' with 'H':");
78
79     Character g = 'G';
80     vt.setLeftChild(f, g);
81     vt.printVector("Setting left child of 'F' with 'G':");
82     if (vt.size() != 6) {
83         throw new Error("Bad size: " + vt.size() + " != 6");
84     }
85     if (!vt.isInternal(f)) {
86         throw new Error("!vt.isInternal(f)");
87     }
88     if (!vt.isExternal(h)) {
89         throw new Error("!vt.isExternal(h)");
90     }
91     if (!vt.rightChild(vt.rightChild(vt.leftChild(vt.root()))).equals(h)) {
92         throw new Error(
93             "!vt.rightChild(vt.rightChild(vt.leftChild(vt.root()))).equals(h)");
94     }
95
96     vt.removeLeftChild(b);
97     if (vt.size() != 6) {
98         throw new Error("Bad size: " + vt.size() + " != 6");
99     }
100
101     vt.removeRightChild(b);
102     vt.printVector("Removing right child of 'B':");
103     if (vt.size() != 3) {
104         throw new Error("Bad size: " + vt.size() + " != 3");
105     }
106     if (!vt.isExternal(b)) {
107         throw new Error("!vt.isExternal(b)");
108     }
109
110     vt.setRightChild(d, 'J');
111     vt.printVector("Setting right child of 'D' with 'J':");
112
113     vt.setRightChild(a, 'X');
114     vt.printVector("Setting right child of root 'A' with 'X':");
115     if (vt.size() != 3) {
116         throw new Error("Bad size: " + vt.size() + " != 3");
117     }
118
119     vt.setRoot('Y');
120     vt.printVector("Setting root with 'Y':");
121     if (vt.size() != 1) {
122         throw new Error("Bad size: " + vt.size() + " != 1");
123     }
124
125     System.out.print("\nTesting if root is external: ");
126     if (!vt.isExternal(vt.root())) {
127         throw new Error("!vt.isExternal(vt.root())");
128     }
129     System.out.println("o.k.");
130
131     System.out.print("\nAsking for node which does not exist: ");
132     Character rightChild = vt.rightChild('Y');
133     if (rightChild != null) {
134         throw new Error("rightChild != null");
135     }
136     System.out.println("o.k.");
137
138

```

30.9.2024 18:01:29

**VectorTreeTest.java**

Page 3/3

```

138
139     System.out.print("\nUsing node which does not exist: ");
140     NoSuchNodeException noSuchNodeException = null;
141     try {
142         vt.setRightChild('A', 'B');
143     } catch (NoSuchNodeException e) {
144         noSuchNodeException = e;
145     }
146     if (noSuchNodeException == null) {
147         throw new Error("NoSuchNodeException missing!");
148     }
149     System.out.println("o.k.");
150
151 }
152
153 }
154
155
156 /* Session-Log:
157
158 Empty tree:
159 [null, null]
160
161 Setting root with 'A':
162 [null, A]
163
164 Setting right child of 'A' with 'D':
165 [null, A, null, D]
166
167 Setting left child of 'A' with 'B':
168 [null, A, B, D]
169
170 Setting right child of 'B' with 'F':
171 [null, A, B, D, null, F, null, null]
172
173 Setting right child of 'F' with 'H':
174 [null, A, B, D, null, F, null, null, null, null, null, H, null, null, null, null]
175
176 Setting left child of 'F' with 'G':
177 [null, A, B, D, null, F, null, null, null, null, G, H, null, null, null, null]
178
179 Removing right child of 'B':
180 [null, A, B, D, null, null, null, null, null, null, null, null, null, null, null, null]
181
182 Setting right child of 'D' with 'J':
183 [null, A, B, D, null, null, null, J, null, null, null, null, null, null, null, null]
184
185 Setting right child of root 'A' with 'X':
186 [null, A, B, X, null, null, null, null, null, null, null, null, null, null, null, null]
187
188 Setting root with 'Y':
189 [null, Y, null, null, null, null, null, null, null, null, null, null, null, null, null, null]
190
191 Testing if root is external: o.k.
192
193 Asking for node which does not exist: o.k.
194
195 Using node which does not exist: o.k.
196
197 */

```

30.9.2024 18:01:29

**TreeInterface.java**

Page 1/1

```

1  /*
2  * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3  * Version: Mon Sep 30 18:01:29 CEST 2024
4  */
5
6  package uebung03.as.aufgabe02;
7
8  public interface TreeInterface<T> {
9
10     T root();
11
12     void setRoot(T root);
13
14     T parent(T child) throws NoSuchNodeException;
15
16     T leftChild(T parent) throws NoSuchNodeException;
17
18     T rightChild(T parent) throws NoSuchNodeException;
19
20     boolean isInternal(T node) throws NoSuchNodeException;
21
22     boolean isExternal(T node) throws NoSuchNodeException;
23
24     boolean isRoot(T node);
25
26     void setRightChild(T parent, T child) throws NoSuchNodeException;
27
28     void setLeftChild(T parent, T child) throws NoSuchNodeException;
29
30     void removeRightChild(T parent) throws NoSuchNodeException;
31
32     void removeLeftChild(T parent) throws NoSuchNodeException;
33
34     int size();
35
36 }
37

```

30.9.2024 18:01:29

## VectorTree.java

Page 1/2

```

1  /*
2  * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3  * Version: Mon Sep 30 18:01:29 CEST 2024
4  */
5
6  package uebung03.as.aufgabe02;
7
8  import java.util.ArrayList;
9
10 public class VectorTree<T> implements TreeInterface<T> {
11
12     private final static int ROOT_INDEX = 1;
13
14     protected ArrayList<T> binaryTree;
15     protected int size;
16
17     public VectorTree() {
18         binaryTree = new ArrayList<>();
19         binaryTree.add(0, null);
20         binaryTree.add(ROOT_INDEX, null);
21     }
22
23     @Override
24     public T root() {
25         // TODO: Implement here...
26         return null;
27     }
28
29     @Override
30     public void setRoot(T root) {
31         // TODO: Implement here...
32     }
33
34     @Override
35     public T parent(T child) throws NoSuchNodeException {
36         // TODO: Implement here...
37         return null;
38     }
39
40     @Override
41     public T leftChild(T parent) throws NoSuchNodeException {
42         // TODO: Implement here...
43         return null;
44     }
45
46     @Override
47     public T rightChild(T parent) throws NoSuchNodeException {
48         // TODO: Implement here...
49         return null;
50     }
51
52     @Override
53     public boolean isInternal(T node) throws NoSuchNodeException {
54         // TODO: Implement here...
55         return false;
56     }
57
58     @Override
59     public boolean isExternal(T node) throws NoSuchNodeException {
60         // TODO: Implement here...
61         return false;
62     }
63
64     @Override
65     public boolean isRoot(T node) {
66         // TODO: Implement here...
67         return false;
68     }
69
70

```

30.9.2024 18:01:29

## VectorTree.java

Page 2/2

```

70
71     @Override
72     public void setRightChild(T parent, T child) throws NoSuchNodeException {
73         // TODO: Implement here...
74     }
75
76     @Override
77     public void setLeftChild(T parent, T child) throws NoSuchNodeException {
78         // TODO: Implement here...
79     }
80
81     @Override
82     public void removeRightChild(T parent) throws NoSuchNodeException {
83         // TODO: Implement here...
84     }
85
86     @Override
87     public void removeLeftChild(T parent) throws NoSuchNodeException {
88         // TODO: Implement here...
89     }
90
91     @Override
92     public int size() {
93         // TODO: Implement here...
94         return -1;
95     }
96
97     public void printVector(String message) {
98         System.out.println("\n" + message);
99         System.out.println(binaryTree);
100    }
101
102 }

```

30.9.2024 18:01:29

**NoSuchNodeException.java**

Page 1/1

```
1  /*
2   * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3   * Version: Mon Sep 30 18:01:29 CEST 2024
4   */
5
6  package uebung03.as.aufgabe02;
7
8  /**
9   * Thrown if a reference to a node (parent or child) is given which is not part
10   * of the tree.
11   */
12  public class NoSuchNodeException extends Exception {
13
14      private static final long serialVersionUID = 1L;
15
16  }
17
```