```
AVLTree.java
1.10.2018 20:09:33
                                                                                  Page 1/3
    * HSR - Uebungen 'Algorithmen & Datenstrukturen 2'
    * Version: Mon Oct 1 20:09:33 CEST 2018
3
   package uebung03.as.aufgabe03;
   import java.util.Collection;
   import uebung02.ml.aufgabe01.BinarySearchTree.Entry;
12
13
   public class AVLTree <K extends Comparable<? super K>, V> {
     private AVLTreeImpl<K, V> avlTreeImpl = new AVLTreeImpl<K, V>();
17
     // Start the GVS-Server first: Double-Click 'GVS Server v1.4.jar'
     //private AVLTreeImpl<K, V> avlTreeImpl = new AVLTreeImplGVS<K, V>();
18
     public V put(K key, V value) {
20
21
       return avlTreeImpl.put(key, value);
22
23
     public V get(K key)
24
25
       return avlTreeImpl.get(key);
26
27
     public int getHeight() {
28
       return avlTreeImpl.getHeight();
29
30
31
     public int size() {
       return avlTreeImpl.size();
33
34
35
     public boolean isEmpty()
       return avlTreeImpl.isEmpty();
37
38
39
     public void clear()
       avlTreeImpl.clear();
42
43
     public Collection<Entry<K, V>> inorder()
       return avlTreeImpl.inorder();
45
46
47
     public void printInorder()
48
49
       avlTreeImpl.printInorder();
50
51
     public void print() {
52
53
       avlTreeImpl.print();
54
55
     protected AVLTreeImpl<K, V> getImpl() {
56
57
       return avlTreeImpl;
58
```

```
AVLTree.java
1.10.2018 20:09:33
                                                                         Page 2/3
     public static void main(String[] args) {
61
62
      AVLTree<Integer, String> avlTree = new AVLTree<Integer, String>();
63
       System.out.println("Inserting 2:");
64
       avlTree.put(2, "Str2");
65
66
       avlTree.print();
      System.out.println("=======");
67
       System.out.println("Inserting 1:");
68
       avlTree.put(1, "Str1");
69
70
       avlTree.print();
71
       System.out.println("=======");
       System.out.println("Inserting 5:");
72
73
       avlTree.put(5, "Str5");
74
       avlTree.print();
75
       System.out.println("==========");
       System.out.println("Inserting 3:");
76
       avlTree.put(3, "Str3");
       avlTree.print();
78
       System.out.println("=======");
       System.out.println("Inserting 6:");
80
       avlTree.put(6, "Str6");
81
       avlTree.print();
82
83
       System.out.println("===========");
       System.out.println("Inserting 4:1:");
84
       avlTree.put(4, "Str4:1");
85
       avlTree.print();
       System.out.println("=======");
87
       System.out.println("Inserting 4:2:");
88
       avlTree.put(4, "Str4:2");
89
       avlTree.print();
       System.out.println("=======");
91
92
       System.out.println("Getting 3 : " + avlTree.get(3));
       System.out.println("Getting 4 : " + avlTree.get(4));
93
       System.out.println("Getting 7 : " + avlTree.get(7));
95
       if (avlTree.getImpl() instanceof AVLTreeImplGVS)
96
         ((AVLTreeImplGVS<Integer, String>)avlTree.getImpl()).gvsTree.disconnect();
97
98
99
100
101
102
103
```

```
AVLTree.java
1.10.2018 20:09:33
                                                                Page 3/3
105 /* Session-Log:
106
107 Inserting 2:
  2 - Str2 : h=0 ROOT
108
110 Inserting 1:
111 1 - Strl : h=0 / parent(key)=2
112 2 - Str2 : h=1 ROOT
114 Inserting 5:
115 1 - Str1 : h=0 / parent(key)=2
116
   2 - Str2 : h=1 ROOT
117 5 - Str5 : h=0 \ parent(key)=2
119 Inserting 3:
  1 - Str1 : h=0 / parent(key)=2
121 2 - Str2 : h=2 ROOT
122 3 - Str3 : h=0 / parent(key)=5
123 5 - Str5 : h=1 \ parent(key)=2
125 Inserting 6:
126 	 1 - Str1 	 : h=0 / parent(key)=2
127 2 - Str2 : h=2 ROOT
128 3 - Str3 : h=0 / parent(key)=5
   5 - Str5 : h=1 \ parent(key)=2
129
130 6 - Str6 : h=0 \ parent(key)=5
132 Inserting 4:1:
  1 - Str1 : h=0 / parent(key)=2
133
134 2 - Str2 : h=3 ROOT
  3 - Str3 : h=1 / parent(key)=5
136 4 - Str4:1 : h=0 \ parent(key)=3
  5 - Str5 : h=2 \ parent(key)=2
138 6 - Str6 : h=0 \ parent(key)=5
139
140 Inserting 4:2:
   1 - Str1 : h=0 / parent(key)=2
141
142 2 - Str2 : h=3 ROOT
143 3 - Str3 : h=1 / parent(key)=5
  4 - Str4:2 : h=0 \ parent(key)=3
   5 - Str5 : h=2 \ parent(key)=2
145
146 6 - Str6 : h=0 \ parent(key)=5
148 Getting 3 :Str3
149 Getting 4 :Str4:2
150 Getting 7 :null
152 */
```

```
AVLTreelmpl.java
1.10.2018 20:09:33
                                                                                  Page 1/3
    * HSR - Uebungen 'Algorithmen & Datenstrukturen 2'
    * Version: Mon Oct 1 20:09:33 CEST 2018
3
   package uebung03.as.aufgabe03;
   import java.util.Collection;
   import java.util.LinkedList;
   import java.util.List;
   import uebung02.ml.aufgabe01.BinarySearchTree;
   class AVLTreeImpl<K extends Comparable<? super K>, V> extends
       BinarySearchTree<K, V> {
16
17
      * After the BST-operation 'insert()':
18
      * actionNode shall point to the parent of the new inserted node.
20
     protected AVLNode actionNode;
21
22
23
     protected class AVLNode extends BinarySearchTree<K, V>.Node {
24
25
26
       private int height;
       private Node parent;
27
28
       AVLNode(Entry<K, V> entry) {
29
30
         super(entry);
31
32
       protected AVLNode setParent(AVLNode parent) {
33
34
         AVLNode old = avlNode(this.parent);
         this.parent = parent;
35
         return old;
37
38
       protected AVLNode getParent() {
39
40
         return avlNode(parent);
41
42
       protected int setHeight(int height) {
43
44
         int old = this.height;
45
         this.height = height;
46
         return old;
47
48
       protected int getHeight() {
50
         return height;
51
52
53
       public AVLNode getLeftChild()
54
55
         return avlNode(super.getLeftChild());
56
57
58
       @Override
       public AVLNode getRightChild()
59
60
         return avlNode(super.getRightChild());
```

```
AVLTreelmpl.java
1.10.2018 20:09:33
                                                                                    Page 2/3
62
        @Override
        public String toString()
64
65
          String result = String.format("%2d - %-6s : h=%d",
                                  getEntry().getKey(), getEntry().getValue(), height);
66
          if (parent == null) {
            result += " ROOT";
68
69
           else
70
            boolean left = (parent.getLeftChild() == this) ? true : false;
            result += (left ? " / ": " \\ ") + "parent(key)="
71
                + parent.getEntry().getKey();
72
73
74
          return result;
75
      } // End of class AVLNode
77
78
79
     protected AVLNode getRoot() {
       return avlNode(root);
81
82
83
     public V put(K key, V value) {
        // TODO Implement here...
86
       return null;
87
88
     public V get(K key)
89
        // TODO Implement here...
90
91
        return null;
92
     @Override
95
     protected Node insert(Node node, Entry<K, V> entry) {
       // TODO Implement here...
96
        return null;
98
99
100
      * The height of the tree.
101
102
      * @return The actual height. -1 for an empty tree.
103
104
      @Override
105
106
     public int getHeight()
107
       return height(avlNode(root));
108
109
110
      * Returns the height of this node.
111
112
      * @param node
113
114
      * @return The height or -1 if null.
115
116
     protected int height(AVLNode node) {
       return (node != null) ? node.getHeight() : -1;
117
118
119
120
      * Assures the heights of the tree from 'node' up to the root.
121
122
123
      * @param node
124
                  The node from where to start.
125
     protected void assureHeights(AVLNode node)
126
        // TODO Implement here...
128
```

```
AVLTreelmpl.java
1.10.2018 20:09:33
                                                                                        Page 3/3
129
130
131
       * Assures the correct height for node.
132
133
         @param node
                   The node to assure its height.
135
136
     protected void setHeight(AVLNode node) {
137
        // TODO Implement here...
138
139
140
       * Factory-Method. Creates a new node.
141
142
       * @param entry
143
                  The entry to be inserted in the new node.
144
145
       * @return The new created node.
146
      protected Node newNode(Entry<K, V> entry) {
148
        // TODO Implement here...
149
        return null;
150
151
152
153
       * Generates an inorder-node-list.
154
155
156
         @param nodeList
                   The node-list to fill in inorder.
157
158
         @param node
                   The node to start from.
159
160
     protected void inorder(Collection<AVLNode> nodeList, AVLNode node)
161
162
        if (node == null)
          return:
163
        inorder(nodeList, node.getLeftChild());
164
        nodeList.add(node);
165
        inorder(nodeList, node.getRightChild());
166
167
168
      // Type-Casting: Node -> AVLNode (Cast-Encapsulation)
169
      @SuppressWarnings("unchecked")
170
     protected AVLNode avlNode(Node node) {
171
        return (AVLNode) node;
172
173
174
175
     public void print() {
        List<AVLNode> nodeList = new LinkedList<>();
176
        inorder(nodeList, avlNode(root));
177
        for (AVLNode node: nodeList) {
   System.out.println(node + " ");
178
179
180
182
183
184
```

```
AVLTreeImplGVS.java
1.10.2018 20:09:33
                                                                                  Page 1/2
    * HSR - Uebungen 'Algorithmen & Datenstrukturen 2'
    * Version: Mon Oct 1 20:09:33 CEST 2018
3
   package uebung03.as.aufgabe03;
   import qvs.tree.GVSBinaryTreeNode;
   import gvs.tree.GVSTreeWithRoot;
   import gvs.typ.node.GVSNodeTyp;
12
   class AVLTreeImplGVS<K extends Comparable<? super K>, V> extends
13
       AVLTreeImpl<K, V> {
     protected GVSTreeWithRoot gvsTree;
17
     private final int DELAY = 200;
18
     protected class AVLNodeGVS extends AVLTreeImpl<K, V>.AVLNode implements GVSBinaryTre
19
20
        protected AVLNodeGVS(Entry<K, V> entry) {
21
22
         super(entry);
23
24
        public GVSBinaryTreeNode getGVSLeftChild()
25
         return (GVSBinaryTreeNode) getLeftChild();
26
27
28
29
        public GVSBinaryTreeNode getGVSRightChild()
         return (GVSBinaryTreeNode) getRightChild();
30
32
33
        public String getNodeLabel() {
         Entry<K, V> e = getEntry();
34
         return e.getKey() + " "+ e.getValue();
          //return e.getKey().toString();
36
37
       public GVSNodeTyp getNodeTyp() -
         return null;
41
42
     } // class BinaryTreeTestGVS.NodeGVS
43
45
     AVLTreeImplGVS()
       this("AVLTreeGVS");
47
50
     AVLTreeImplGVS(String title) {
       gvsTree = new GVSTreeWithRoot(title);
51
52
53
55
     protected Node newNode(Entry<K, V> entry) {
       return new AVLNodeGVS(entry);
56
58
59
     @Override
     public V put(K key, V value) {
60
       V result = super.put(key, value);
       gvsTree.setRoot((GVSBinaryTreeNode) root);
62
63
       gvsTree.display();
       try {Thread.sleep(DELAY);} catch (InterruptedException e) {}
        return result;
66
```

```
AVLTreelmplGVS.java
1.10.2018 20:09:33
                                                                                 Page 2/2
     @Override
     public Entry<K, V> insert(K key, V value) {
70
71
       Entry<K, V> newEntry = super.insert(key, value);
       gvsTree.setRoot((GVSBinaryTreeNode) root);
72
       qvsTree.display();
       try {Thread.sleep(DELAY);} catch (InterruptedException e) {}
74
       return newEntry;
75
76
77
78
     @Override
79
     public Entry<K, V> remove(Entry<K, V> entry)
       Entry<K, V> deletedEntry = super.remove(entry);
       gvsTree.display();
81
       try {Thread.sleep(DELAY);} catch (InterruptedException e) {}
83
       return deletedEntry;
84
85
87
88
89
```

```
AVLTreeJUnitTest.java
1.10.2018 20:09:33
                                                                                   Page 1/2
    * HSR - Uebungen 'Algorithmen & Datenstrukturen 2'
    * Version: Mon Oct 1 20:09:33 CEST 2018
3
   package uebung03.as.aufgabe03;
   import static org.junit.Assert.assertEquals;
8
   import static org.junit.Assert.assertNull;
   import java.util.Collection;
   import java.util.LinkedList;
12
   import org.junit.After;
   import org.junit.Before;
   import org.junit.FixMethodOrder;
   import org.junit.Test;
   import org.junit.runners.MethodSorters;
   @FixMethodOrder(MethodSorters.NAME ASCENDING)
21
   public class AVLTreeJUnitTest {
22
     AVLTreeImpl<Integer, String> avlTree;
24
25
26
     public void setUp() {
27
        //System.setProperty("NoGVS", "true");
28
        avlTree = new AVLTree<Integer, String>().getImpl();
29
30
31
     public void tearDown()
33
       if (avlTree instanceof AVLTreeImplGVS)
34
          ((AVLTreeImplGVS<Integer, String>)avlTree).gvsTree.disconnect();
35
37
38
39
     public void test01Put() {
        int[] keys = { 2, 1, 3 };
        String[] expected = {
42
            " 1 - Str1 : h=0 / parent(key)=2",
43
            " 2 - Str2 : h=1 ROOT",
44
            " 3 - Str3 : h=0 \\ parent(key)=2",
45
46
47
        runTest(keys, expected);
48
        assertEquals(1, avlTree.getHeight());
50
51
     public void test02Get() {
52
53
        int[] keys = { 2, 1, 4, 5, 3 };
        String[] expected = {
54
            " 1 - Str1 : h=0 / parent(key)=2",
55
            " 2 - Str2 : h=2 ROOT",
56
57
            " 3 - Str3 : h=0 / parent(key)=4",
            " 4 - Str4 : h=1 \\ parent(key)=2",
58
            " 5 - Str5 : h=0 \\ parent(key)=4",
59
60
        runTest(keys, expected);
61
62
        assertEquals(2, avlTree.getHeight());
       assertEquals("Str2", avlTree.get(2));
assertEquals("Str5", avlTree.get(5));
63
64
        assertNull(avlTree.get(0));
65
        assertNull(avlTree.get(6));
67
```

```
AVLTreeJUnitTest.java
1.10.2018 20:09:33
                                                                                    Page 2/2
69
     public void test03() {
70
71
        int[] keys = { 2, 3, 1 };
       String[] expected = {
72
            "1 - Str1 : h=0 / parent(key)=2",
            " 2 - Str2 : h=1 ROOT",
74
75
            " 3 - Str3 : h=0 \\ parent(key)=2",
76
77
       runTest(kevs, expected);
        assertEquals(1, avlTree.getHeight());
       avlTree.put(2, "Str2:2");
avlTree.put(2, "Str2:3");
79
80
       assertEquals(1, avlTree.getHeight());
81
82
        expected = new String[]
83
            " 1 - Str1 : h=0 / parent(key)=2",
84
            " 2 - Str2:3 : h=1 ROOT",
            " 3 - Str3 : h=0 \\ parent(key)=2",
85
        Collection<AVLTreeImpl<Integer, String>.AVLNode> nodes = new LinkedList<>();
87
        avlTree.inorder(nodes, avlTree.getRoot());
88
       verify(nodes, expected);
89
90
91
92
     private void runTest(int[] keys, String[] expected) {
93
        for (int key : keys) {
         avlTree.put(key, "Str" + key);
95
96
97
       Collection<AVLTreeImpl<Integer, String>.AVLNode> nodes = new LinkedList<>();
        avlTree.inorder(nodes, avlTree.getRoot());
98
        assertEquals(expected.length, nodes.size());
       verify(nodes, expected);
100
101
102
     private void verify(Collection<AVLTreeImpl<Integer, String>.AVLNode> nodes, String[]
    expected) {
104
        int i = 0;
        for (AVLTreeImpl<Integer, String>.AVLNode node: nodes) {
105
106
          String nodeStr = node.toString();
107
         String expectedStr = expected[i];
         assertEquals(expectedStr, nodeStr);
108
109
         i++;
110
111
112
113
```