```java
/*
 * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
 * Version: Mon Sep 30 18:01:29 CEST 2024
 */

package uebung03.as.aufgabe02;


public class VectorTreeTest {

  public static void main(String[] args) throws NoSuchNodeException {

    VectorTree<Character> vt = new VectorTree<>();

    vt.printVector("Empty tree:");

    if (vt.size() != 0) {
      throw new Error("Bad size: " + vt.size() + " != 0");
    }
    if (vt.root() != null) {
      throw new Error("vt.root() != null");
    }

    Character a = 'A';
    vt.setRoot(a);
    vt.printVector("Setting root with 'A':");
    if (vt.size() != 1) {
      throw new Error("Bad size: " + vt.size() + " != 1");
    }
    if (!vt.isRoot(a)) {
      throw new Error("!vt.root(a)");
    }
    if (!vt.root().equals(a)) {
      throw new Error("!vt.root().equals(a) : " + vt.root());
    }
    if (!vt.isExternal(a)) {
      throw new Error("!vt.isExternal(a)");
    }
    if (vt.parent(a) != null) {
      throw new Error("vt.parent(a) != null");
    }

    Character d = 'D';
    vt.setRightChild(vt.root(), d);
    vt.printVector("Setting right child of 'A' with 'D':");
    if (vt.size() != 2) {
      throw new Error("Bad size: " + vt.size() + " != 2");
    }
    if (!vt.rightChild(vt.root()).equals(d)) {
      throw new Error("!vt.rightChild(vt.root()).equals(d) : "
          + vt.rightChild(vt.root()));
    }
    if (!vt.isExternal(d)) {
      throw new Error("!vt.isExternal(d)");
    }
    if (!vt.isInternal(vt.root())) {
      throw new Error("!vt.isInternal(vt.root()");
    }
    if (!vt.parent(d).equals(a)) {
      throw new Error("!vt.parent(d).equals(a)");
    }

    Character b = 'B';
    vt.setLeftChild(vt.root(), b);
    vt.printVector("Setting left child of 'A' with 'B':");
    if (vt.size() != 3) {
      throw new Error("Bad size: " + vt.size() + " != 3");
    }


```



```java

    Character f = 'F';
    vt.setRightChild(b, f);
    vt.printVector("Setting right child of 'B' with 'F':");

    Character h = 'H';
    vt.setRightChild(f, h);
    vt.printVector("Setting right child of 'F' with 'H':");

    Character g = 'G';
    vt.setLeftChild(f, g);
    vt.printVector("Setting left child of 'F' with 'G':");
    if (vt.size() != 6) {
      throw new Error("Bad size: " + vt.size() + " != 6");
    }
    if (!vt.isInternal(f)) {
      throw new Error("!vt.isInternal(f)");
    }
    if (!vt.isExternal(h)) {
      throw new Error("!vt.isExternal(h)");
    }
    if (!vt.rightChild(vt.rightChild(vt.leftChild(vt.root()))).equals(h)) {
      throw new Error(
          "!vt.rightChild(vt.rightChild(vt.leftChild(vt.root()))).equals(h)");
    }

    vt.removeLeftChild(b);
    if (vt.size() != 6) {
      throw new Error("Bad size: " + vt.size() + " != 6");
    }

    vt.removeRightChild(b);
    vt.printVector("Removing right child of 'B':");
    if (vt.size() != 3) {
      throw new Error("Bad size: " + vt.size() + " != 3");
    }
    if (!vt.isExternal(b)) {
      throw new Error("!vt.isExternal(b)");
    }

    vt.setRightChild(d, 'J');
    vt.printVector("Setting right child of 'D' with 'J':");

    vt.setRightChild(a, 'X');
    vt.printVector("Setting right child of root 'A' with 'X':");
    if (vt.size() != 3) {
      throw new Error("Bad size: " + vt.size() + " != 3");
    }

    vt.setRoot('Y');
    vt.printVector("Setting root with 'Y':");
    if (vt.size() != 1) {
      throw new Error("Bad size: " + vt.size() + " != 1");
    }

    System.out.print("\nTesting if root is external: ");
    if (!vt.isExternal(vt.root())) {
      throw new Error("!vt.isExternal(vt.root())");
    }
    System.out.println("o.k.");

    System.out.print("\nAsking for node which does not exist: ");
    Character rightChild = vt.rightChild('Y');
    if (rightChild != null) {
      throw new Error("rightChild != null");
    }
    System.out.println("o.k.");


```

```
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 */
```



```
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
```

```java
/*
 * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
 * Version: Mon Sep 30 18:01:29 CEST 2024
 */

package uebung03.as.aufgabe02;

import java.util.ArrayList;

public class VectorTree<T> implements TreeInterface<T> {

  private final static int ROOT_INDEX = 1;

  protected ArrayList<T> binaryTree;
  protected int size;

  public VectorTree() {
    binaryTree = new ArrayList<>();
    binaryTree.add(0, null);
    binaryTree.add(ROOT_INDEX, null);
  }

  @Override
  public T root() {
    // TODO: Implement here...
    return null;
  }

  @Override
  public void setRoot(T root) {
    // TODO: Implement here...
  }

  @Override
  public T parent(T child) throws NoSuchNodeException {
    // TODO: Implement here...
    return null;
  }

  @Override
  public T leftChild(T parent) throws NoSuchNodeException {
    // TODO: Implement here...
    return null;
  }

  @Override
  public T rightChild(T parent) throws NoSuchNodeException {
    // TODO: Implement here...
    return null;
  }

  @Override
  public boolean isInternal(T node) throws NoSuchNodeException {
    // TODO: Implement here...
    return false;
  }

  @Override
  public boolean isExternal(T node) throws NoSuchNodeException {
    // TODO: Implement here...
    return false;
  }

  @Override
  public boolean isRoot(T node) {
    // TODO: Implement here...
    return false;
  }

```



```java

  @Override
  public void setRightChild(T parent, T child) throws NoSuchNodeException {
    // TODO: Implement here...
  }

  @Override
  public void setLeftChild(T parent, T child) throws NoSuchNodeException {
    // TODO: Implement here...
  }

  @Override
  public void removeRightChild(T parent) throws NoSuchNodeException {
    // TODO: Implement here...
  }

  @Override
  public void removeLeftChild(T parent) throws NoSuchNodeException {
    // TODO: Implement here...
  }

  @Override
  public int size() {
    // TODO: Implement here...
    return -1;
  }

  public void printVector(String message) {
    System.out.println("\n" + message);
    System.out.println(binaryTree);
  }

}
```

```java
/*
 * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
 * Version: Mon Sep 30 18:01:29 CEST 2024
 */

package uebung03.as.aufgabe02;

/**
 * Thrown if a reference to a node (parent or child) is given which is not part
 * of the tree.
 */
public class NoSuchNodeException extends Exception {

  private static final long serialVersionUID = 1L;

}

```