```
1
2  # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3  # Path   : uebung03/al/aufgabe02
4  # Version: Sun Oct  1 20:01:13 CEST 2023
5
6  from uebung03.al.aufgabe02.vector_tree import VectorTree
7  from uebung03.al.aufgabe02.vector_tree import NoSuchNodeException
8
9  if __name__ == '__main__':
10
11   vt = VectorTree()
12
13   vt.print_vector("Empty tree:")
14
15   if vt.size() != 0:
16     raise Exception("Bad size: " + vt.size() + " != 0")
17
18   if vt.root() != None:
19       raise Exception("vt.root() != None")
20
21   a = 'A'
22   vt.set_root(a)
23   vt.print_vector("Setting root with 'A':")
24   if vt.size() != 1:
25     raise Exception("Bad size: " + vt.size() + " != 1")
26   if not vt.is_root(a):
27     raise Exception("not vt.root(a)")
28   if vt.root() != a:
29     raise Exception("vt.root() != a: " + vt.root())
30   if not vt.is_external(a):
31     raise Exception("not vt.is_external(a)")
32   if vt.parent(a) != None:
33     raise Exception("vt.parent(a) != None")
34
35
36   d = 'D'
37   vt.set_right_child(vt.root(), d)
38   vt.print_vector("Setting right child of 'A' with 'D':")
39   if  vt.size() != 2:
40     raise Exception("Bad size: " + vt.size() + " != 2")
41   if not vt.right_child(vt.root()) == d:
42     raise Exception("not vt.right_child(vt.root()) == d : " + vt.right_child(vt.root()
   ))
43   if not vt.is_external(d):
44     raise Exception("not vt.is_external(d)")
45   if not vt.is_internal(vt.root()):
46     raise Exception("!vt.is_internal(vt.root()")
47   if not vt.parent(d) == a:
48     raise Exception("not vt.parent(d) == a")
49
50
51   b = 'B'
52   vt.set_left_child(vt.root(), b)
53   vt.print_vector("Setting left child of 'A' with 'B':")
54   if vt.size() != 3:
55     raise Exception("Bad size: " + vt.size() + " != 3")
56
57   f = 'F'
58   vt.set_right_child(b, f)
59   vt.print_vector("Setting right child of 'B' with 'F':")
60
61   h = 'H'
62   vt.set_right_child(f, h)
63   vt.print_vector("Setting right child of 'F' with 'H':")
64
65
66
```



```
66
67   g = 'G'
68   vt.set_left_child(f, g)
69   vt.print_vector("Setting left child of 'F' with 'G':")
70   if vt.size() != 6:
71     raise Exception("Bad size: " + vt.size() + " != 6")
72   if not vt.is_internal(f):
73     raise Exception("not vt.is_internal(f)")
74   if not vt.is_external(h):
75     raise Exception("not vt.is_external(h)")
76   if not vt.right_child(vt.right_child(vt.left_child(vt.root()))) == h:
77     raise Exception("not vt.right_child(vt.right_child(vt.left_child(vt.root()))) == h
   ")
78
79   vt.remove_left_child(b)
80   if vt.size() != 6:
81     raise Exception("Bad size: " + vt.size() + " != 6")
82
83   vt.remove_right_child(b)
84   vt.print_vector("Removing right child of 'B':")
85   if vt.size() != 3:
86     raise Exception("Bad size: " + vt.size() + " != 3")
87   if not vt.is_external(b):
88     raise Exception("not vt.is_external(b)")
89
90   vt.set_right_child(d, 'J')
91   vt.print_vector("Setting right child of 'D' with 'J':")
92
93   vt.set_right_child(a, 'X')
94   vt.print_vector("Setting right child of root 'A' with 'X':")
95   if vt.size() != 3:
96     raise Exception("Bad size: " + vt.size() + " != 3")
97
98   vt.set_root('Y')
99   vt.print_vector("Setting root with 'Y':")
100  if vt.size() != 1:
101    raise Exception("Bad size: " + vt.size() + " != 1")
102
103  print("\nTesting if root is external: ", end = "")
104  if not vt.is_external(vt.root()):
105    raise Exception("not vt.is_external(vt.root())")
106  print("o.k.")
107
108  print("\nAsking for node which does not exist: ", end = "")
109  rightChild = vt.right_child('Y')
110  if rightChild != None:
111    raise Exception("rightChild != None")
112  print("o.k.")
113
114  print("\nUsing node which does not exist: ", end = "")
115  noSuchNodeException = None
116  try:
117    vt.set_right_child('A', 'B')
118  except (NoSuchNodeException) as e:
119    noSuchNodeException = e
120  if noSuchNodeException == None:
121    raise Exception("NoSuchNodeException missing!")
122  print("o.k.")
123
124
125
```

```
125
126 """ Session-Log::
127
128 Empty tree:
129 [None, None]
130
131 Setting root with 'A':
132 [None, 'A']
133
134 Setting right child of 'A' with 'D':
135 [None, 'A', None, 'D']
136
137 Setting left child of 'A' with 'B':
138 [None, 'A', 'B', 'D']
139
140 Setting right child of 'B' with 'F':
141 [None, 'A', 'B', 'D', None, 'F', None, None]
142
143 Setting right child of 'F' with 'H':
144 [None, 'A', 'B', 'D', None, 'F', None, None, None, None, None, 'H', None, None, None,
    None]
145
146 Setting left child of 'F' with 'G':
147 [None, 'A', 'B', 'D', None, 'F', None, None, None, None, 'G', 'H', None, None, None, N
    one]
148
149 Removing right child of 'B':
150 [None, 'A', 'B', 'D', None, None, None, None, None, None, None, None, None, None, None
    , None]
151
152 Setting right child of 'D' with 'J':
153 [None, 'A', 'B', 'D', None, None, None, 'J', None, None, None, None, None, None, None,
     None]
154
155 Setting right child of root 'A' with 'X':
156 [None, 'A', 'B', 'X', None, None, None, None, None, None, None, None, None, None, None
    , None]
157
158 Setting root with 'Y':
159 [None, 'Y', None, None, None, None, None, None, None, None, None, None, None, None, No
    ne, None]
160
161 Testing if root is external: o.k.
162
163 Asking for node which does not exist: o.k.
164
165 Using node which does not exist: o.k.
166
167 """
```



```
1
2  # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3  # Path   : uebung03/al/aufgabe02
4  # Version: Sun Oct  1 20:01:13 CEST 2023
5
6  import enum
7  from uebung03.al.aufgabe02.no_such_node_exception import NoSuchNodeException
8
9
10 ROOT_INDEX = 1
11
12 class VectorTree:
13
14   class _child_side(enum.Enum):
15     LEFT = enum.auto()
16     RIGHT = enum.auto()
17
18   def __init__(self):
19     self._binary_tree = []
20     self._binary_tree.append(None)
21     self._binary_tree.append(None)
22     self._size = 0
23
24   def root(self):
25     # TODO: Implement here...
26     pass
27
28   def set_root(self, root):
29     # TODO: Implement here...
30     pass
31
32   def parent(self, child):
33     # TODO: Implement here...
34     pass
35
36   def left_child(self, parent):
37     # TODO: Implement here...
38     pass
39
40   def right_child(self, parent):
41     # TODO: Implement here...
42     pass
43
44   def is_internal(self, node):
45     # TODO: Implement here...
46     pass
47
48   def is_external(self, node):
49     # TODO: Implement here...
50     pass
51
52   def is_root(self, node):
53     # TODO: Implement here...
54     pass
55
56   def set_right_child(self, parent, child):
57     # TODO: Implement here...
58     pass
59
60   def set_left_child(self, parent, child):
61     # TODO: Implement here...
62     pass
63
64   def remove_right_child(self, parent):
65     # TODO: Implement here...
66     pass
67
68   def remove_left_child(self, parent):
69     # TODO: Implement here...
70     pass
71
```

```
71
72    def size(self):
73      # TODO: Implement here...
74      pass
75
76    def print_vector(self, message):
77      print("\n" + message)
78      print(self._binary_tree)
```



```
1
2   # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3   # Path   : uebung03/al/aufgabe02
4   # Version: Sun Oct  1 20:01:13 CEST 2023
5
6   class NoSuchNodeException(Exception):
7
8       def __init__(self, err):
9         super().__init__(err)
10
11
```