```
1
2   # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3   # Path   : uebung02/ml/aufgabe03
4   # Version: Mon Sep 25 19:12:19 CEST 2023
5
6   from uebung02.ml.aufgabe03.empty_stack_exception import EmptyStackException
7   import sys
8
9
10  class StackImplementation:
11    '''
12    Stack: a collection of objects that are inserted and removed according
13    to the last-in first-out principle.
14    '''
15    # --- nested _Node class: -----------------
16    class _Node:
17
18      def __init__(self, elem):
19        self._element = elem
20        self._next = None
21
22      def append_node(self, nextNode):
23        self._next = nextNode
24
25      def get_next(self):
26        return self._next
27
28      def get_element(self):
29        return self._element
30
31
32    # --- stack methods: -----------------
33
34    def __init__(self):
35      self._top = None
36      self._size = 0
37
38    def __len__(self):
39      return self._size
40
41    def size(self):
42      return self.__len__()
43
44    def is_empty(self):
45      return self._size == 0
46
47    def top(self):
48      if self._size == 0:
49        raise EmptyStackException("Could not get top of stack because stack is empty.")
50      return self._top.get_element()
51
52    def push(self, element):
53      new_node = self._Node(element)
54      new_node.append_node(self._top)
55      self._top = new_node
56      self._size += 1
57
58    def pop(self):
59      if self._size == 0:
60        raise EmptyStackException("Could not remove top of stack because stack is empty.
    ")
61      top_node = self._top
62      self._top = top_node.get_next()
63      self._size -= 1
64      return top_node.get_element()
```



```
65
66    def printout(self):
67      print("Stack: (", self._to_string(self._top, ""), ")")
68
69    def _to_string(self, node, content):
70      if node == None:
71        return content
72      if not content == "":
73        content += ", "
74      content += str(node.get_element())
75      return self._to_string(node.get_next(), content)
76
```

```
77
78  if __name__ == '__main__':
79    stack = StackImplementation()
80    stack.printout()
81    TEST_SIZE = 4
82    for i in range(TEST_SIZE):
83      stack.push(i)
84      stack.printout()
85      if stack.size() != i+1:
86        print("ERROR: Stack.size() != ", i+1)
87        sys.exit()
88      if stack.top() != i:
89        print("ERROR: Stack.top() != ", i)
90        sys.exit()
91    for i in range(TEST_SIZE-1, 0, -1):
92      if stack.pop() != i:
93        print("ERROR: Stack.pop() != ", i)
94        sys.exit()
95      stack.printout()
96      if stack.size() != i:
97        print("ERROR: Stack.size() != ", i)
98        sys.exit()
99      if stack.top() != i-1:
100       print("ERROR: Stack.top() != ", i-1)
101       sys.exit()
102   if stack.pop() != 0:
103     print("ERROR: Stack.pop() != ", 0)
104     sys.exit()
105   stack.printout()
106   if not stack.is_empty():
107     print("ERROR: Stack.empty() != true")
108     sys.exit()
109   if stack.size() != 0:
110     print("ERROR: Stack.size() != 0")
111     sys.exit()
112   try:
113     stack.top()
114     print("ERROR: no EmptyStackException for stack.top()!")
115     sys.exit()
116   except EmptyStackException:
117     pass
118   try:
119     stack.pop()
120     print("ERROR: no EmptyStackException for stack.pop()!")
121     sys.exit()
122   except EmptyStackException:
123     pass
124
125
126 """ Session-Log:
127
128 Stack: (  )
129 Stack: ( 0 )
130 Stack: ( 1, 0 )
131 Stack: ( 2, 1, 0 )
132 Stack: ( 3, 2, 1, 0 )
133 Stack: ( 2, 1, 0 )
134 Stack: ( 1, 0 )
135 Stack: ( 0 )
136 Stack: (  )
137
138 """
139
```



```
1
2   # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3   # Path   : uebung02/ml/aufgabe03
4   # Version: Mon Sep 25 19:12:19 CEST 2023
5
6   class EmptyStackException(Exception):
7
8       def __init__(self, err):
9         super().__init__(err)
10
11
```