```python

# HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
# Path   : uebung05/ml/aufgabe01
# Version: Mon Oct 16 18:32:46 CEST 2023

from uebung05.ml.aufgabe01.map_impl import MapImpl


if __name__ == '__main__':

  the_map = MapImpl()

  print("map.size()     : " + str(the_map.size()))
  print("map.is_empty() : " + str(the_map.is_empty()))

  print("\nmap.put(1, \"one\") : " + str(the_map.put(1, "one")))
  the_map.printMap()
  print("map.size()     : " + str(the_map.size()))
  print("map.is_empty() : " + str(the_map.is_empty()))

  the_map.put(2, "two")
  the_map.put(3, "three 1")
  the_map.printMap("\n")

  print("\nmap.put(3, \"three 2\") : " + str(the_map.put(3, "three 2")))
  the_map.printMap()

  print()
  print("map.get(2) : " + str(the_map.get(2)))
  print("map.get(4) : " + str(the_map.get(4)))

  print("\nmap.remove(2) : " + str(the_map.remove(2)))
  the_map.printMap()

  print("\nmap.key_set()  : " + str(the_map.key_set()))
  print("map.values()   : " + str(the_map.values()))
  print("map.entrySet() : " + ",".join(map(str, the_map.entry_set())))


```



```

""" Session-Log (Note: The order of the entries is irrelevant):

map.size()     : 0
map.is_empty() : True

map.put(1, "one") : None
Printing map (1 Entries):
    1: one
map.size()     : 1
map.is_empty() : False

Printing map (3 Entries):
    1: one
    2: two
    3: three 1

map.put(3, "three 2") : three 1
Printing map (3 Entries):
    1: one
    2: two
    3: three 2

map.get(2) : two
map.get(4) : None

map.remove(2) : two
Printing map (2 Entries):
    1: one
    3: three 2

map.key_set()  : {1, 3}
map.values()   : ['one', 'three 2']
map.entrySet() : (1,one),(3,three 2)

"""
```

```
1
2   # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3   # Path   : uebung05/ml/aufgabe01
4   # Version: Mon Oct 16 18:32:46 CEST 2023
5
6   import enum
7   from uebung05.ml.aufgabe01.entry import Entry
8
9
10  class MapImpl:
11
12    def __init__(self):
13      self._list = list()
14
15    def size(self):
16      return len(self._list)
17
18    def is_empty(self):
19      if self._list:
20        return False
21      else:
22        return True
23
24    def put(self, key, value):
25      entry = self._find(key)
26      if (entry != None):
27        result = entry.set_value(value)
28      else:
29        self._list.append(Entry(key, value))
30        result = None
31      return result
32
33    def _find(self, key):
34      it = iter(self._list)
35      e = next(it, None)
36      while e != None:
37        this_key = e.get_key()
38        if this_key == key:
39          return e
40        e = next(it, None)
41      return None
42
43    def get(self, key):
44      return self._get_remove(key, remove = False)
45
46    def remove(self, key):
47      return self._get_remove(key, remove = True)
48
49    def _get_remove(self, key, remove):
50      entry = self._find(key)
51      if (entry != None):
52        result = entry.get_value()
53        if remove:
54          self._list.remove(entry)
55      else:
56        result = None
57      return result
58
59    def values(self):
60      values = list()
61      for entry in self._list:
62        values.append(entry.get_value())
63      return values
64
65    class _KeyOrEntry(enum.Enum):
66      KEY = enum.auto()
67      ENTRY = enum.auto()
68
69    def key_set(self):
70      return self._set_key_entry(MapImpl._KeyOrEntry.KEY)
71
```



```
71
72    def entry_set(self):
73      return self._set_key_entry(MapImpl._KeyOrEntry.ENTRY)
74
75    def _set_key_entry(self, key_or_entry):
76      the_set = set()
77      for e in self._list:
78        if key_or_entry == MapImpl._KeyOrEntry.KEY:
79          the_set.add(e.get_key())
80        else:
81          the_set.add(e)
82      return the_set
83
84    def printMap(self, prefix = ""):
85      print(prefix + "Printing map (" + str(self.size()) + " Entries): ")
86      for e in self._list:
87        print(f"  {e.get_key():3d}: {e.get_value()}")
88
```

```
1
2  # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3  # Path   : uebung05/ml/aufgabe01
4  # Version: Mon Oct 16 18:32:46 CEST 2023
5
6
7  class Entry:
8
9    def __init__(self, key, value):
10     self._key = key
11     self._value = value
12
13   def get_key(self):
14     return self._key
15
16   def get_value(self):
17     return self._value
18
19   def set_value(self, value):
20     old_value = self._value
21     self._value = value
22     return old_value
23
24   def __iter__(self):
25     yield self._key
26     yield self._value
27
28   def __eq__(self, other):
29     return (isinstance(other, type(self)) and tuple(self) == tuple(other))
30
31   def __hash__(self):
32     return hash(tuple(self))
33
34   def __str__(self):
35     return "(" + str(self._key) + "," + str(self._value) +")"
```