```
 1  using System.Collections.Generic;
 2  using System.Security.Cryptography.X509Certificates;
 3
 4  namespace DAMLib
 5  {
 6      public class DictionaryCollection<K, V>
 7      {
 8          private Item[] _item = new Item[0];
 9
10          public delegate bool DelegateFilterKeyValue(K key, V value);
11          public delegate bool DelegateFilterWithoutParameters();
12          public delegate bool DelegateFilterKey(K key);
13
14          private class Item
15          {
16              public K Key;
17              public V Value;
18
19              public Item(K key, V value)
20              {
21                  this.Key = key;
22                  this.Value = value;
23              }
24          }
25
26          public int Count => _item.Length;
27          public bool IsEmpty => _item.Length < 0;
28
29          public int GetIndexOf(V value)
30          {
31              if (value == null)
32                  return 0;
33
34              for (int i = 0; i < _item.Length; i++)
35              {
36                  if (_item[i].Value.Equals(value))
37                      return i;
38              }
39              return -1;
40          }
41
42          public V? GetElementAt(K key)
43          {
44              if (key == null)
45                  return default(V);
46
47              for (int i = 0; i < _item.Length; i++)
48              {
49                  if (_item[i].Key.Equals(key))
50                      return _item[i].Value;
51              }
52              return default(V);
53          }
```

```
54
55
56          public void Add(K key, V value)
57          {
58              if (ContainsKey(key))
59                  return;
60
61              int count = _item.Length;
62              Item[] setResult = new Item[count + 1];
63              Item element = new Item(default, default);  // TODO:
                  REVISAR
64              setResult[count] = element;
65
66              for (int i = 0; i < count; i++)
67              {
68                  setResult[i] = _item[i];
69              }
70
71              setResult[count].Key = key;
72              setResult[count].Value = value;
73
74              _item = setResult;
75          }
76
77          public void RemoveAt(int index)
78          {
79              if (index < 0 || index > _item.Length)
80                  return;
81
82              if (index == -1)
83                  return;
84
85              int count = _item.Length;
86              Item[] arrayResult = new Item[count - 1];
87
88              for (int i = 0; i < index; i++)
89              {
90                  arrayResult[i] = _item[i];
91              }
92
93              for (int i = index; i < count - 2; i++)
94              {
95                  arrayResult[i] = _item[i + 1];
96              }
97
98              _item = arrayResult;
99          }
100
101         private bool ContainsKey(K key)
102         {
103             if (key == null)
104                 return false;
105
```

```
106             for (int i = 0; i < _item.Length; i++)
107             {
108                 if (_item[i].Key.Equals(key))
109                     return true;
110             }
111             return false;
112         }
113 
114         public bool Contains(V value)
115         {
116             // return IndexOf >= 0;
117             if (value == null)
118                 return false;
119 
120             for (int i = 0; i < _item.Length; i++)
121             {
122                 if (_item[i].Value.Equals(value))
123                     return true;
124             }
125             return false;
126         }
127 
128         // Funcion que devuelve si dos DICCIONARIOS son iguales.
129         public override bool Equals(object? obj)
130         {
131             return (this == obj);
132         }
133 
134         public bool AreIdentical(object? obj)
135         {
136             // Comprueba que DOS DICCIONARIOS SON IGUALES
137             // NO COMPRUEBA SI DOS ITEMS SON IGUALES
138             if (obj == null)
139                 return false;
140             if (obj is not Item)
141                 return false;
142             Item identicalItem = (Item)obj;
143 
144             for(int i = 0; i < _item.Length;i++)
145             {
146                 if (_item[i].Key.Equals(identicalItem.Key) &&
147                     _item[i].Value.Equals(identicalItem.Value))
148                     return true;
149             }
150             return false;
151         }
152 
153         public override int GetHashCode()
154         {
155             return 133 * 533 * 224 * _item.GetHashCode();
156         }
157 
158         // Funcion delegada Filter que devuelve un diccionario.
```

```
159         public DictionaryCollection<K, V> Filter(DelegateFilterKeyValue
               del)
160         {
161             DictionaryCollection<K, V> dictionaryResult = new
                  DictionaryCollection<K, V>();
162
163             for (int i = 0; i < _item.Length; i++)
164             {
165                 bool InsertIntoCollection = del(_item[i].Key, _item
                      [i].Value);
166                 if (InsertIntoCollection)
167                 {
168                     dictionaryResult.Add(_item[i].Key, _item[i].Value);
169                 }
170             }
171
172             return dictionaryResult;
173         }
174
175         public DictionaryCollection<K, V> Filter(DelegateFilterKey del)
176         {
177             DictionaryCollection<K, V> dictionaryResult = new
                  DictionaryCollection<K, V>();
178
179             for (int i = 0; i < _item.Length; i++)
180             {
181                 bool InsertIntoCollection = del(_item[i].Key);
182                 if (InsertIntoCollection)
183                 {
184                     dictionaryResult.Add(_item[i].Key, _item[i].Value);
185                 }
186             }
187
188             return dictionaryResult;
189         }
190         public DictionaryCollection<K, V> Filter
              (DelegateFilterWithoutParameters del)
191         {
192             DictionaryCollection<K, V> dictionaryResult = new
                  DictionaryCollection<K, V>();
193
194             for (int i = 0; i < _item.Length; i++)
195             {
196                 dictionaryResult.Add(_item[i].Key, _item[i].Value);
197             }
198
199             return dictionaryResult;
200         }
201
202         public void Clear()
203         {
204             _item = new Item[0];
205         }
```

```
206
207         public override string ToString()
208         {
209             string result = "";
210             foreach (Item i in _item)
211             {
212                 result += $"La key {i.Key}, contiene el value {i.Value} 
                      \n";
213             }
214             return result;
215         }
216     }
217 }
```