```
 1  using System.Xml.Linq;
 2
 3  namespace DAMLib
 4  {
 5      public class ItemSet<T>
 6      {
 7          private Item[] _itemset;
 8
 9          private class Item
10          {
11              public T element;
12              public int hash;
13
14              public Item()
15              {
16
17              }
18              public Item(T element, int hash)
19              {
20                  this.element = element;
21                  this.hash = hash;
22              }
23          }
24
25          public int Count
26          {
27              get
28              {
29                  if (_itemset == null)
30                      return 0;
31                  return _itemset.Length;
32              }
33          }
34
35          public bool IsEmpty => _itemset.Length == 0;
36
37
38          public ItemSet()
39          {
40              _itemset = Array.Empty<Item>();
41          }
42
43          // Funcion publica que añade un elemento al Set.
44          public void Add(T element)
45          {
46              if (element == null)
47                  return;
48
49              if (Contains(element))
50                  return;
51              else
52                  AddElement(element);
53          }
```

```
54
55          private void AddElement(T element)
56          {
57              if (element == null)
58                  return;
59
60              int newLength = _itemset.Length + 1;
61              Item[] newItemArray = new Item[newLength];
62
63              int hash = element.GetHashCode();
64              Item newItem = new Item(element, hash);
65
66              for (int i = 0; i < newLength - 1; i++)
67              {
68                  newItemArray[i] = _itemset[i];
69              }
70              newItemArray[newLength - 1] = newItem;
71
72              _itemset = newItemArray;
73          }
74
75          // Funcion que elimina un elemento del Set.
76          public void RemoveAt(int index)
77          {
78              if (index < 0 || index >= _itemset.Length)
79                  return;
80
81              int newLength = _itemset.Length - 1;
82              Item[] newItemArray = new Item[newLength];
83
84
85              for (int i = 0; i < index; i++)
86              {
87                  newItemArray[i] = _itemset[i];
88              }
89
90              for (int i = index; i < newLength; i++)
91              {
92                  newItemArray[i] = _itemset[i + 1];
93              }
94
95              _itemset = newItemArray;
96          }
97
98          // Funcion que comprueba si contiene un elemento.
99          public bool Contains(T element)
100         {
101             return IndexOf(element) >= 0;
102         }
103
104         // Funcion que devuelve el indice de un elemento del Set.
105         public int IndexOf(T element)
106         {
```

```
107             if (element == null)
108                 return -1;
109 
110             int hash = element.GetHashCode();
111 
112             for (int i = 0; i < _itemset.Length; i++)
113             {
114                 Item item = _itemset[i];
115                 if (hash == item.hash && item.element.Equals(element))
116                 {
117                     return i;
118                 }
119             }
120             return -1;
121         }
122 
123         // Funcionn que elimina todo el contenido de la coleccion.
124         public void Clear()
125         {
126             _itemset = new Item[0];
127         }
128 
129         // Funcion que devuelve el codigo hash de un elemento.
130         public override int GetHashCode()
131         {
132             return 133 * 533 * 224 * _itemset.GetHashCode();
133         }
134 
135         public override string ToString()
136         {
137             string result = "";
138             int count = 0;
139 
140             foreach (Item item in _itemset)
141             {
142                 result = $"El elemento numero {count} del Set es
                      {item.element}";
143                 count++;
144             }
145             return result;
146         }
147     }
148 }
149 
```