

```
1 using System.Xml.Linq;
2
3 namespace DAMLib
4 {
5     public class Set<T>
6     {
7         private T[] _set;
8         private bool _testAttribute;    // Atributo utilizado en la función EqualsDeep()
9
10
11         public bool IsEmpty => _set.Length == 0;
12         public int Count
13         {
14             get
15             {
16                 if (_set == null)
17                     return 0;
18                 else
19                     return _set.Length;
20             }
21         }
22
23         public Set()
24         {
25             _set = new T[0];
26         }
27
28         // Funcion que añade un elemento SOLO en caso que NO exista dentro de la coleccion.
29         public void Add(T element)
30         {
31             if (element == null)
32                 return;
33
34             if (!Contains(element))
35             {
36                 int count = _set.Length;
37                 T[] setResult = new T[count + 1];
38
39                 for (int i = 0; i < count; i++)
40                 {
41                     setResult[i] = _set[i];
42                 }
43
44                 setResult[count] = element;
45                 _set = setResult;
46             }
47         }
48
49         public bool Contains(T element)
50         {
51             if (element == null)
```

```
52         return false;
53
54         return IndexOf(element) >= 0;
55     }
56
57     public int IndexOf(T element)
58     {
59         if (element == null)
60             return -1;
61
62         for (int i = 0; i < _set.Length; i++)
63         {
64             if (_set[i].Equals(element))
65                 return i;
66         }
67
68         return -1;
69     }
70
71     public void Remove(T element)
72     {
73         if (element == null)
74             return;
75
76         int index = IndexOf(element);
77
78         if (index == -1)
79             return;
80
81         int count = _set.Length;
82         T[] arrayResult = new T[count - 1];
83
84         // Posibilidad 1. Con dos bucles 'for'.
85         for (int i = 0; i < index; i++)
86         {
87             arrayResult[i] = _set[i];
88         }
89
90         for (int i = index; i < count - 1; i++)
91         {
92             arrayResult[i] = _set[i + 1];
93         }
94
95         // Posibilidad 2. Con instruccion 'continue'.
96         /*
97         for(int i = 0; i < count; i++)
98         {
99             if (i == index)
100                 continue;
101             arrayResult[i] = _set[i];
102         }
103         */
104     }
```

```
105         _set = arrayResult;
106     }
107
108     public override bool Equals(object? obj)
109     {
110         return this == obj;
111     }
112
113     public override int GetHashCode()
114     {
115         return 133 * 533 * 224 * _testAttribute.GetHashCode();
116     }
117
118     public bool IsEqualsInDeep(object? obj)
119     {
120         if (this == obj)
121             return true;
122
123         if (obj is not TestCar)
124             return false;
125
126         TestCar car = (TestCar)obj;
127
128         return (this._testAttribute == car.TestAttribute);
129     }
130
131     public void Clear()
132     {
133         _set = Array.Empty<T>();
134     }
135
136     public override string ToString()
137     {
138         string result = "";
139         int count = 0;
140
141         foreach (T element in _set)
142         {
143             result += $"El elemento numero {count} de la coleccion ↗  

144                 es: {element}.\n";
145             count++;
146         }
147
148         return result;
149     }
150 }
```