```
using System.Xml.Linq;

namespace DAMLib
{
    public class Set<T>
    {
        private T[] _set;
        private bool _testAtribute;     // Atributo utilizado en la
          funcion EqualsDeep()


        public bool IsEmpty => _set.Length == 0;
        public int Count
        {
            get
            {
                if (_set == null)
                    return 0;
                else
                    return _set.Length;
            }
        }

        public Set()
        {
            _set = new T[0];
        }

        // Funcion que añade un elemento SOLO en caso que NO exista
          dentro de la coleccion.
        public void Add(T element)
        {
            if (element == null)
                return;

            if (!Contains(element))
            {
                int count = _set.Length;
                T[] setResult = new T[count + 1];

                for (int i = 0; i < count; i++)
                {
                    setResult[i] = _set[i];
                }

                setResult[count] = element;
                _set = setResult;
            }
        }

        public bool Contains(T element)
        {
            if (element == null)
```

```
                return false;

            return IndexOf(element) >= 0;
        }

        public int IndexOf(T element)
        {
            if (element == null)
                return -1;

            for (int i = 0; i < _set.Length; i++)
            {
                if (_set[i].Equals(element))
                    return i;
            }

            return -1;
        }

        public void Remove(T element)
        {
            if (element == null)
                return;

            int index = IndexOf(element);

            if (index == -1)
                return;

            int count = _set.Length;
            T[] arrayResult = new T[count - 1];

            // Posibilidad 1. Con dos bucles 'for'.
            for (int i = 0; i < index; i++)
            {
                arrayResult[i] = _set[i];
            }

            for (int i = index; i < count - 2; i++)
            {
                arrayResult[i] = _set[i + 1];
            }

            // Posibilidad 2. Con instruccion 'continue'.
            /*
            for(int i = 0; i < count; i++)
            {
                if (i == index)
                    continue;
                arrayResult[i] = _set[i];
            }
            */

```

```
                _set = arrayResult;
            }

            public override bool Equals(object? obj)
            {
                return this == obj;
            }

            public override int GetHashCode()
            {
                return 133 * 533 * 224 * _testAtribute.GetHashCode();
            }

            public bool IsEqualsInDeep(object? obj)
            {
                if (this == obj)
                    return true;

                if (obj is not TestCar)
                    return false;

                TestCar car = (TestCar)obj;

                return (this._testAtribute == car.TestAtribute);
            }

            public void Clear()
            {
                _set = Array.Empty<T>();
            }

            public override string ToString()
            {
                string result = "";
                int count = 0;

                foreach (T element in _set)
                {
                    result += $"El elemento numero {count} de la coleccion 
                      es: {element}.\n";
                    count++;
                }

                return result;
            }
        }
    }
```