

The background features several large, stylized geometric shapes in the corners. In the top-left and bottom-right, there are dark gray shapes with a white outline, resembling chevrons or stylized triangles. In the top-right and bottom-left, there are lighter gray shapes, also stylized. The central text is positioned between these decorative elements.

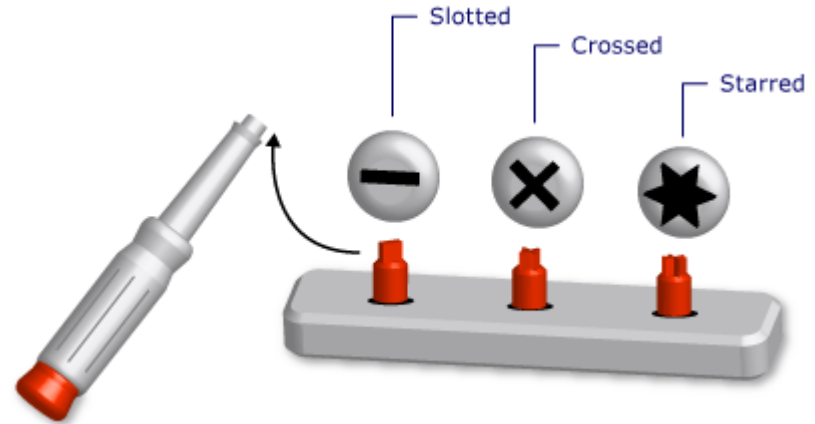
# Clase 12

## Generics

## ¿Qué son los GENERICS?

Elementos que se adaptan para realizar la misma funcionalidad para una variedad de tipos de datos.

Sin necesidad de definir una versión para cada tipo.



# Clases Genéricas



```
1 List<string> paises = new List<string>();
2
3 paises.Add("España");
4 paises.Add("Rusia");
5
6 paises.RemoveAt("España");
7
8 List<Jugador> jugadores = new List<Jugador>();
9
10 Jugador j1 = new Jugador(1,"A");
11 jugadores.Add(j1);
12 Jugador j2 = new Jugador(3,"B");
13 jugadores.Add(j2);
14
15 jugadores.Remove(j1);
16 jugadores.Remove(j2);
```



```
1 public class List<T>
2     : IList<T>, System.Collections.IList, IReadOnlyList<T>
3 {
4     private const int _defaultCapacity = 4;
5
6     private T[] _items;
7     [ContractPublicPropertyName("Count")]
8     private int _size;
9     private int _version;
10
11     static readonly T[] _emptyArray = new T[0];
12
13     public List()
14     {
15         _items = _emptyArray;
16     }
17
18     public void Add(T item)
19     {
20         if (_size == _items.Length)
21         {
22             EnsureCapacity(_size + 1);
23         }
24         _items[_size++] = item;
25         _version++;
26     }
27
28     //...
29 }
```

### SEGURIDAD DE TIPO

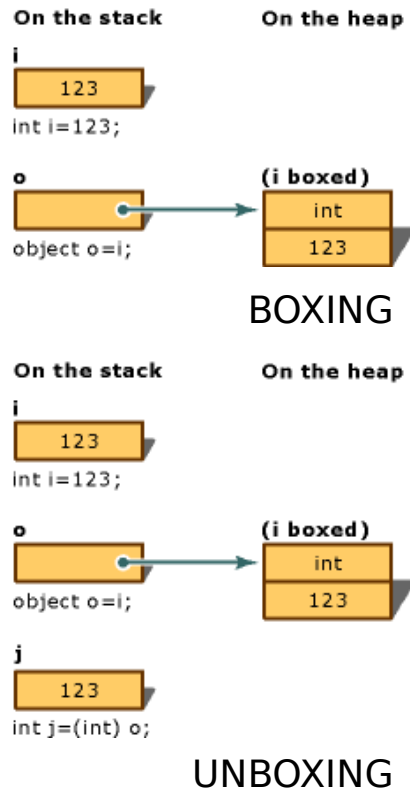
Previene posibles errores

### REUTILIZACION DE CODIGO

Misma tarea con tipos distintos

### Generics VS No-Generics

Eficiencia. Evita boxing y Unboxing



# Métodos Genéricos



```
1  Array arrayStr = Array.Empty<string>();  
2  
3  Array arrayJugadores = Array.Empty<Jugadores>();  
4
```

## RESTRICCIONES

<b>where T : struct</b>	El argumento de tipo debe ser un tipo de valor.
<b>where T : class</b>	El argumento de tipo debe ser un tipo de referencia.
<b>where T : unmanaged</b>	No puede ser tipo de referencia. Implica struct
<b>where T : new()</b>	El argumento de tipo debe tener un constructor público sin parámetros.
<b>where T : &lt;nombre de la clase base&gt;</b>	El argumento de tipo debe ser de la clase especificada o derivada
<b>where T : &lt;nombre de interfaz&gt;</b>	El argumento de tipo debe implementar la interfaz especificada
<b>where T : U</b>	El argumento de tipo <b>T</b> debe ser del tipo <b>U</b> o derivados