1 Static

When we write:

```csharp
Console.WriteLine(Math.PI);
```

The `Math` actually refers to a class, and not to an object. How is that?

Actually, everything in the `MATH` class is static (public static class `Math`), and the PI constant is actually public! (public `const double PI`).

Class attribute: can be static or not, public or private, a constant or variable.

```csharp
public const double PI = 3.14159265358979;
```

We also have static methods:

```csharp
Math.Min(x,y);
Math.Max(x,y);
Math.Pow(x,y);
```

A static member (variable, method, etc) belongs to the type of an object rather than to an instance of that type.

1.1 Static Class Members

Class member = methods and fields (attributes)

Motivation: the methods we are using the most (`WriteLine`, `ConsoleRead`) are static, but all the methods we are writing are not (they are “non-static”, or “instance”).

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Non-static Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ClassName.MethodName(arguments)</code></td>
<td><code>ObjectName.MethodName(arguments)</code></td>
</tr>
<tr>
<td><code>Math.Pow(2, 5)</code> (2^5 = 32)</td>
<td><code>myRectangle.SetLength(5)</code></td>
</tr>
</tbody>
</table>

A static class member is associated with the class instead of with the object.

<table>
<thead>
<tr>
<th></th>
<th>Static Field</th>
<th>Non-static Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static method</td>
<td>OK</td>
<td>NO</td>
</tr>
<tr>
<td>Non-static method</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>
using System;

static class Lib
{
    public static int ValueIsIndex(int[] arrayP)
    {
        int res = 0;
        for (int i = 0; i < arrayP.Length; i++)
            if (arrayP[i] == i) res++;
        return res;
    }

    public static bool AtLeastOneValueIsIndex(int[] arrayP)
    {
        return (ValueIsIndex(arrayP) > 0);
    }

    public static int ValueMatch(int[] arrayP1, int[] arrayP2)
    {
        int res = 0;
        int smallestSize;
        if (arrayP1.Length < arrayP2.Length) smallestSize = arrayP1.Length;
        else smallestSize = arrayP2.Length;
        for (int i = 0; i < smallestSize; i++)
            if (arrayP1[i] == arrayP2[i]) res++;
        return res;
    }
}

using System;

class Program
{
    static void Main(string[] args)
    {
        int[] arrayA = {0, 3, 5, 12, 4, 5, 8};
        Console.WriteLine(Lib.ValueIsIndex(arrayA));
        Console.WriteLine(Lib.AtLeastOneValueIsIndex(arrayA));

        int[] arrayB = {3, 5, 4, 12, 5, 8};
        Console.WriteLine(Lib.ValueIsIndex(arrayB));
        Console.WriteLine(Lib.AtLeastOneValueIsIndex(arrayB));

        Console.WriteLine(Lib.ValueMatch(arrayA, arrayB));
    }
}