**Los métodos en C#**

Un **método** es un bloque de código que contiene varias instrucciones que podemos ejecutar las veces que necesitemos. Son de gran ayuda cuando tienes que hacer lo mismo varias veces en un mismo programa para no repetir el mismo código una y otra vez.

En la clase anterior conocimos cómo invocar un método y utilizarlo cuantas veces necesitemos para realizar acciones específicas en nuestro programa. Ahora veamos algunos de estos métodos para el manejo de strings.

**Métodos de strings**

**using** System;

**public** **class** **Program**

{

 **public** **static** **void** **Main**()

 {

 **string** ClassTopic = "Métodos de strings";

 **string** School = "Platzi";

 Console.WriteLine("Estoy aprendiendo de "+ ClassTopic + " en " + School + ".");

 }

}

En la pieza de código anterior puedes ver un pequeño programa con dos variables tipo string y su impresión. Para este tipo de variables podemos utilizar una gran cantidad de métodos predefinidos dentro de la biblioteca estándar de C#.

Utilicemos estas dos variables strings para conocer algunos de ellos.

**Clone()**

 string **SchoolClone** = **School.Clone().ToString();**

 Console.WriteLine(**SchoolClone);**

El método Clone() crea un clon exacto de un string en una variable de tipo objeto. Por ello es necesario convertirlo a string utilizando ToString().

**CompareTo()**

**Console**.WriteLine(**ClassTopic**.CompareTo(**School**));

**Console**.WriteLine(**School**.CompareTo(**SchoolClone**));

Sirve para comparar el contenido de dos strings. Si son iguales devuelve 0 para true y si son diferentes retorna 1 para false. En este caso como School y SchoolClone son strings idénticos devolverá un 0. ¿Qué crees que se imprima en la primera línea?

**Contains()**

Console.WriteLine(School.Contains("Pla"));

Retorna True si el string contiene los caracteres que se pongan en los argumentos del método.

**EndsWith()**

Console.WriteLine(School.EndsWith("zi"));

Similar a Contains(), pero para comparar si los caracteres están al final del string.

**StartsWith()**

Console.WriteLine(School.StartsWith("Pl"));

Funciona de la misma manera que EndsWith(), pero en este caso comprueba si empieza con una serie de caracteres indicados.

**Equals()**

**Console**.WriteLine(**School**.Equals(**SchoolClone**));

Funciona de forma parecida a CompareTo(), pero para indicar si el contenido de ambos strings retorna True o False.

**IndexOf()**

Console.WriteLine(School.IndexOf("a"));

Regresa la posición dentro del string del carácter indicado en el argumento.

**ToLower() y ToUpper()**

**Console**.WriteLine(**ClassTopic**.ToLower());

**Console**.WriteLine(**ClassTopic**.ToUpper());

Convierten el string en todas letras minúsculas o mayúsculas.

**Insert()**

Console.WriteLine(**School.Insert(6,** " es educación online efectiva"));

Inserta un nuevo string dentro de otro en la posición indicada dentro de los argumentos. En este caso es en 6 que es la posición después de la última letra de “Platzi”.

**LastIndexOf()**

Console.WriteLine(ClassTopic.LastIndexOf("s"));

Regresa la posición de la última vez que aparece el carácter. En este caso la letra “s” aparece varias veces en el string y así podemos notar cómo devuelve la última posición en la que aparece.

**Remove()**

**Console**.WriteLine(**ClassTopic**.Remove(6));

Elimina los caracteres del string a partir de la posición que le indiquemos hasta el final.

**Replace()**

Console.WriteLine(ClassTopic.Replace("s", "z"));

Reemplaza todos los caracteres por otro indicado. En este caso todas las “s” serán cambiadas por “z”.

**Split()**

string[] **split** = ClassTopic.**Split**(new char[] { 's' });

 Console.WriteLine(**split**[0]);

 Console.WriteLine(**split**[1]);

 Console.WriteLine(**split**[2]);

Este es un caso especial donde creamos un nuevo string donde guardamos el string separado en trozos por la letra indicada en los argumentos del método. Para este caso se separa por la letra “s” en 3 trozos que son los que imprimimos.

**Substring()**

**Console**.WriteLine(**ClassTopic**.Substring(2,10));

Devuelve un substring o trozo de string de acuerdo a las posiciones indicadas en los argumentos.

**ToCharArray()**

**School**.ToCharArray();

Convierte el string en un arreglo de caracteres.

**Trim()**

**string** TextWithSpaces = " hola, había espacios al principio y al final ";

Console.WriteLine(TextWithSpaces.Trim());

Elimina espacios al principio y al final de un string.

**Tu reto**

Ya conoces varios de los métodos que puedes utilizar con strings. Estos son los más comunes, pero existen unos más.

Como reto de esta clase crea un programa donde utilices por lo menos 5 de estos métodos de strings o de algún otro que investigues. Pon a volar tu creatividad y comparte en los comentarios el programa que creaste.

# Cómo crear tus propios métodos

**Creando nuestro método**

Lo primero que debemos hacer es crear un nuevo proyecto con nuestra clase base tal cual lo hemos hecho hasta ahora. Nombra tu proyecto MyMethods y tendrás un código como este:

**using** System;

**public** **class** **MyMethods**

{

 **public** **static** **void** **Main**()

 {

 Console.WriteLine("Hello World");

 }

}

Arriba del método Main() crea tu primer método siguiendo una estructura similar de la siguiente manera:

 **public** **void** IntegerAddition(**int** a, **int** b)

 {

 **int** Addition = a + b;

 Console.WriteLine("El resultado es " + Addition);

 }

Este método tiene el objetivo de sumar dos números enteros. Para ello dentro de los paréntesis se incluyeron dos parámetros que son los dos números a sumar.

Dentro del cuerpo o bloque del método hay dos instrucciones. La primera int Addition = a + b; es donde se realiza la operación de suma, asignándose a una variable del mismo tipo, y en la siguiente imprimimos ese resultado.

**Accediendo al método**

Para utilizar nuestro método debemos llamarlo y lo haremos desde nuestro método Main().

 **public** **static** **void** **Main**()

 {

 // Aquí llamaremos al método

 }

Es necesario generar una referencia creando una instancia de nuestra clase donde se ejecuta el programa. Para ello escribe:

MyMethods MyProgram = **new** MyMethods();

Donde MyMethods es la clase del programa y MyProgram el nombre que se le da a la instancia que se crea y que se usará como referencia para ejecutar el método las veces que sea necesario como se ve a continuación:

**MyProgram**.IntegerAddition(3,6);

Esta línea probablemente te parece más parecida a lo que has visto con los métodos que hemos utilizado antes. Lo que hace es llamar a la instancia MyProgram y a su método IntegerAddition() que es el que creamos. Dentro de sus paréntesis se ponen los argumentos que necesitan los parámetros int a e int b que se le asignaron.

 **public** **static** **void** **Main**()

 {

 MyMethods MyProgram = **new** MyMethods();

 MyProgram.IntegerAddition(3,6);

 }

**Métodos que retornan un valor**

Algo que quizá no notaste es que el método anterior tenía una palabra clave “void”, lo que significa que no retorna algún valor. Es por ello que dentro del mismo método se imprimió el resultado.

Para generar métodos de los que podamos obtener valores conocerás la palabra clave return y al momento de crear un método se le pondrá el tipo de dato que retornará en lugar de void:

 **public** **int** **IntegerMultiplication**(**int** a, **int** b)

 {

 **int** Multiplication = a \* b;

 **return** Multiplication;

 }

Este es un método muy similar al anterior con la diferencia de que tiene un valor int a ser retornado.

Al final de todo el contenido del método se encuentra la instrucción return Multiplication; que indica, con la palabra clave return que regresará el valor que hay dentro de esa variable al punto donde se llame el método desde Main().

public **static** void **Main**()

 {

 int **result** = **MyProgram**.**IntegerMultiplication**(12,2);

 **Console**.**WriteLine**("El resultado de la multiplicación es " + **result**);

 }

En este caso, como el método regresa un valor, es necesario almacenarlo en una variable. Al imprimir ese resultado se verá cómo el método funciona correctamente ejecutando la multiplicación.

**using** **System**;

public class **MyMethods**

{

 public void **IntegerAddition**(int a, int b)

 {

 int **Addition** = a + b;

 **Console**.**WriteLine**("El resultado es " + **Addition**);

 }

 public int **IntegerMultiplication**(int a, int b)

 {

 int **Multiplication** = a \* b;

 **return** **Multiplication**;

 }

 public **static** void **Main**()

 {

 **MyMethods** **MyProgram** = new **MyMethods**();

 **MyProgram**.**IntegerAddition**(3,6);

 int **result** = **MyProgram**.**IntegerMultiplication**(12,2);

 **Console**.**WriteLine**("El resultado de la multiplicación es " + **result**);

 }

}

Ahora conoces cómo crear y utilizar tus propios métodos. Como reto de esta clase crea un método que ejecute la división de dos números. Como tip toma en cuenta que la mayoría de las divisiones no dan resultados enteros, por lo que necesitarás definir tu método y variables con otro tipo de dato.