**CLASE 01**

1. ¿Qué es el CLR?

Cuando compilamos un programa desarrollado en plataforma .NET, se compila a un lenguaje denominado MSIL (Microsoft Intermediate Lenguage), el cual es traducido a lenguaje maquina mediante el CLR (Common lenguage Runtime).

El CLR es una máquina virtual que compila / traduce el código MSIL en tiempo de ejecución. “Su funcionamiento es muy parecido al hipervisor de una máquina virtual, lo cual permite ejecutar aplicaciones .NET multiplataforma.

A su vez, el CLR también se encarga de reservar espacio de memoria y liberar recursos del sistema una vez finalizada la ejecución de la aplicación gracias al garbage collector.

1. Describa el proceso de compilación de C#.

El código fuente, gracias al compilador de lenguaje, es traducido a un código intermedio (IL) ensamblado con metadatos, los cuales son tablas que contienen información / descripción completa del código.

Gracias a la metadata, el compilador JITTER (el cual se encuentra dentro del CLR), obtiene toda la información que este necesita para compilar el código a lenguaje máquina.

1. ¿Qué es el CTS?

El Common Type System (CTS) es el conjunto de reglas que se deben seguir en las definiciones de tipos de datos para que el CLR las acepte. En otras palabras, aunque cada lenguaje tiene su propia sintaxis para definir tipos de datos, en el MSIL resultante de la compilación de su código fuente se tienen que cumplir las reglas del CTS.

1. ¿Qué es un tipo por referencia (Reference Type)? ¿En qué se diferencia de un tipo por valor (Value Type)?

Una variable de un tipo de valor contiene un valor de un tipo determinado. (Entero, Flotante, etc). Estos tipos de variables se eliminan de la memoria stack cuando el método en donde se la está utilizando termina su ejecución.

Cuando se pasa una variable por valor, se pasa una copia de la misma y ésta también se almacena en la memoria stack.

Una variable por referencia en cambio contiene la dirección de memoria de donde se encuentran los datos de manera que se usa esa dirección para acceder de forma indirecta a los mismos. Se asigna en la memoria Heap y tiene la ventaja de poder ser accesible desde otras clases.

1. Explique las diferencias entre variables escalares y no escalares.

Un escalar es un determinado valor (que no varía todo a lo largo del programa informático), una variable, o un campo, que solamente puede tener un valor en un cierto momento; en comparación o contraposición, están los conceptos de array (vector), lista y objeto, que pueden tener almacenado en su estructura más de un valor.

1. ¿Cuál es el punto de entrada (entry point) para los programas en C#?

El entry point es un solo método público estático MAIN que recibe un array de tipo string como parámetro opcional.

1. ¿Cuál es la diferencia entre una conversión implícita y una explícita?

En una conversión implícita no interviene el programador y no hay perdida de datos, ya que no podes castear de algo grande a algo más chico y en la explícita sí puede haber perdida de datos, ya que en esta última puede haber perdida de datos (por ejemplo pasar de float a int y perder la parte decimal).

1. ¿Para qué sirven los modificadores out y ref? ¿En qué se diferencian?

Aunque ambas hacen que se pase una variable por referencia y no por valor, ref indica que el parámetro es de entrada/salida.

Out, es únicamente de salida y se puede utilizar en caso de necesitar varios valores de retorno (como en el método TryParse, donde obtenemos un valor booleano en el return tradicional y un valor parseado en el parámetro out).

Ésto nos lleva a una diferencia importante: los parámetros pasados como ref deben estar inicializados previamente (se va a modificar su valor), mientras que los parámetros pasados como out no es necesario que lo estén (se va a asignar un valor sin importarnos lo que tuviera previamente).

1. Explique cómo funciona el método TryParse. ¿Qué retorna? ¿Qué recibe? ¿Cuál es la diferencia con el método Parse?

Ambos métodos se usan para convertir un string a un tipo deseado por el programador.

**TipoDeDato.Parse():** Recibe como parámetro el string a convertir, y retorna una variable del tipo deseado. En el caso de que no haya podido hacerlo, puede devolver 3 excepciones según que parámetro se le haya pasado. (Parametro Nulo, Parametro con Error de formato y Parametro fuera de los limites del tipo deseado como retorno)

**TipoDeDato.TryParse():** Recibe como parámetro un string a convertir, una dirección de memoria del tipo deseado y retorna un valor booleano.

En el caso de que la conversión haya sido exitosa, retorna TRUE y guarda el valor convertido en la dirección de memoria mencionada, caso contrario, retorna FALSE.