Conocimientos Básicos

Que es el CLR?(Common Language Runtime)

El **CLR** es un compilador que se encarga de convertir el código que escribimos en los lenguajes de microsoft como C#, VBA, etc. En un código intermedio llamado **CIL**(Common Intermediate Language). Este ultimo se encarga de ir compilando este código intermedio en código maquina en tiempo real.

Las ventajas del CLR son:

- La integración entre lenguajes
- Control de excepciones entre lenguajes
- La seguridad mejorada
- La compatibilidad con la implementación y las versiones
- Proporciona un modelo simplificado de interacción y servicios de generación de perfiles y depuración

El código desarrollado con un compilador de lenguaje orientado al tiempo de ejecución(Como Java o .NET) se denomina **código administrado**.

Gestión de memoria del CLR

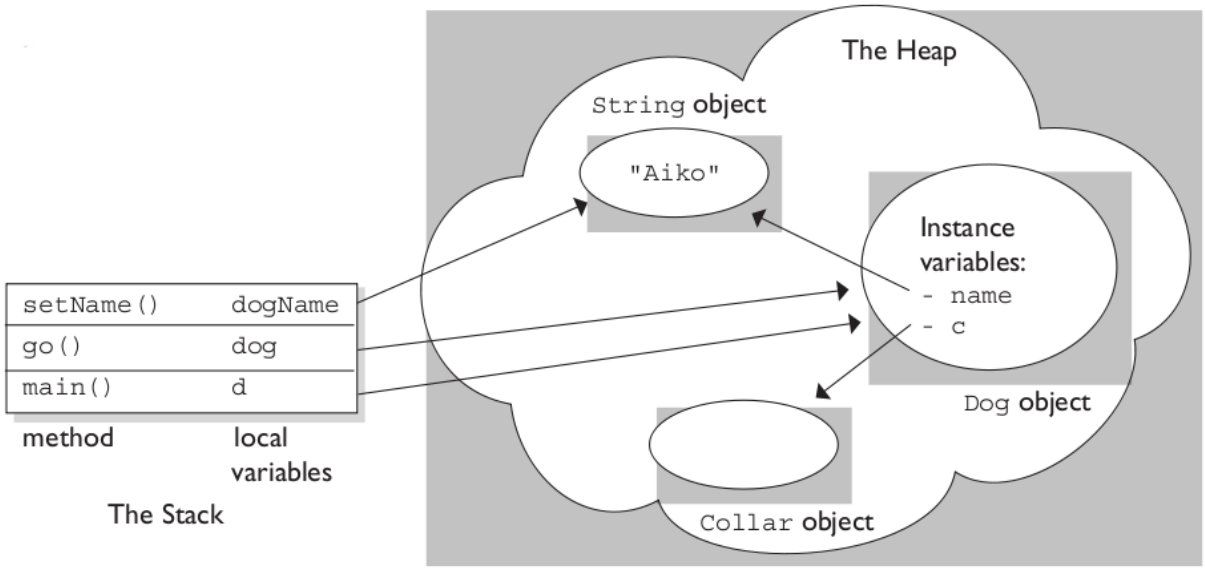
Stack(pila) & Heap(montículo)

En .NET las variables tienen dos tipos de almacenamiento: valores **tipo** y valores **referencia**.

Los valores **tipo** son las variables simples como `int`, `char`, `byte`, struct, Enum, etc. Estas se almacenan en memoria en el **STACK**

Los valores **referencia** generalmente son objetos de clase como `string`, también en arrays, etc. Que tienen cierta complejidad y por tanto se almacenen en el **HEAP**. Estos valores lo que hacen es una referencia en el heap la ubicación del objeto correspondiente en el stack.

Un ejemplo de los almacenamientos:



El GC actúa sobre el Heap y no sobre el stack porque los segmentos almacenados en la pila son pequeños y de un tamaño controlado, mientras que en el montículo se almacenan bloques de memoria de distinta índole y tamaño.

El recolector de basura (GC)

El GC es un algoritmo que esta programado para determinar cuando es necesaria liberar memoria y que es lo que hay que liberar del programa que hemos ejecutado.

Como sabe el GC que se puede liberar?

- **Una raíz es**: una variable global, un objeto estático, una variable local o un registro de CPU. Una raíz no se puede eliminar de memoria porque se entiende que se puede utilizar en cualquier momento.

El GC piensa que todo es basura y puede ser eliminado, para asegurarse recorre todo el Heap buscando las raíces.

Con las raíces localizadas obtiene un grafo con elementos que son accesibles desde el código y los que no. El GC tiene el concepto `alcanzable`, que son objetos accesibles desde una raíz. Estos tampoco pueden ser eliminados.

Existirán objetos que no están apuntados por nadie. Éstos objetos son candidatos a ser eliminados en la próxima pasada del GC. Esto ocurrirá cuando se detecte que la memoria se está acabando. Cuando esto suceda, los objetos que no se usan serán eliminados, los bloques de memoria utilizada serán compactados y se actualizará el **NextObjPtr** para apuntar a la nueva primera dirección de memoria libre para asignar.

Finalización de elementos

Existen formas dentro del entorno .NET para poder indicar al GC que queremos liberar de memoria ciertos objetos.

Para poder hacer uso de ello existen diferentes métodos e implementaciones para ello:

- `~NombreClase`: Instrucción para indicar al GC que contenido tiene que ejecutar cuando libere el objeto(cuando le toque liberarlo, no se le obliga a ello), para mas info: uso de ~
- `Dispose()`: Con este método se fuerza al recolector de basura a liberar el recurso que le hemos indicado, para mas info: uso de dispose
- `Close()`: Con close se indica que el objeto de podrá volver a usar pero por el momento sera liberado.