INSTITUTO KHIPU

Semestre III

Sesión 04

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

HERRAMIENTAS DE PROGRAMACION – C#

Tema

TIPOS DE METODOS: VARIABLES GLOBALES Y MÉTODO QUE RETORNA DATOS TIPO STRING

TIPOS DE METODOS

1. VARIABLES GLOBALES

En el mundo de la programación, las variables juegan un papel fundamental para el almacenamiento y manejo de datos. En C#, el concepto de **variables globales** puede ser interpretado de varias maneras, aunque técnicamente, el lenguaje no soporta directamente las variables globales como otros lenguajes como C o C++. Sin embargo, es posible lograr un comportamiento similar utilizando estrategias adecuadas. Este artículo aprenderas qué son las variables globales en C#, cómo se pueden implementar y las mejores prácticas para su uso.

¿Qué es una variable global?

Una **variable global** es aquella que puede ser accedida desde cualquier parte de un programa, independientemente de su ubicación. Son variables que tienen un alcance global dentro de una aplicación, lo que significa que están disponibles en todos los métodos y clases. Aunque esto puede parecer conveniente, un uso inadecuado de las variables globales puede llevar a problemas de mantenimiento, legibilidad y errores difíciles de depurar.

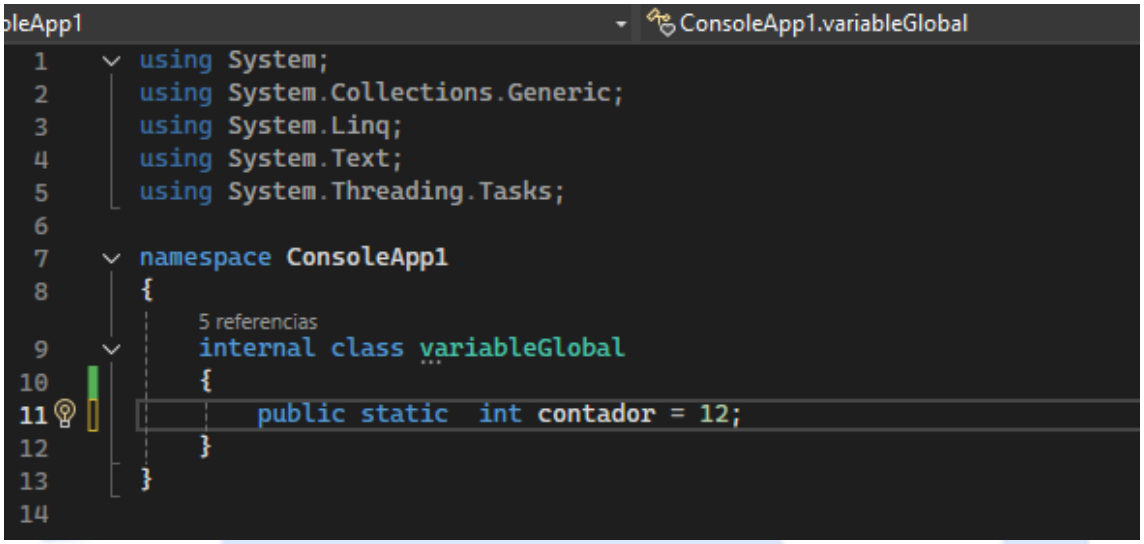
En C#, no existen variables globales propiamente dichas, pero podemos emular este comportamiento utilizando variables estáticas dentro de clases.

Implementación de variables "globales" en C#

La forma más común de crear una variable que se comporte como global en C# es declararla como static dentro de una clase. Las variables estáticas pertenecen a la

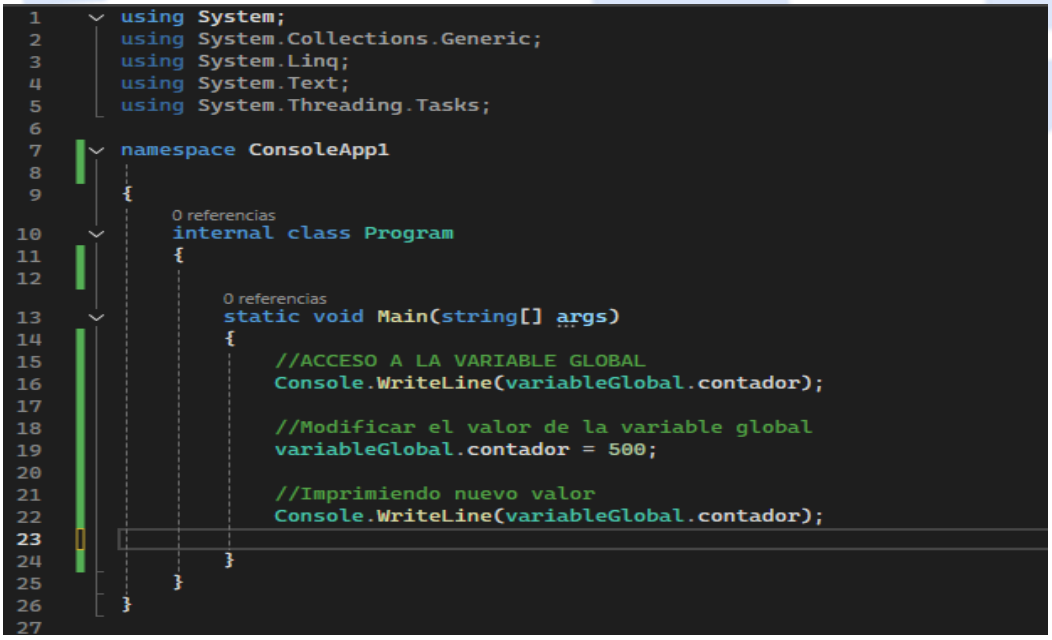
clase en lugar de a una instancia, lo que significa que pueden ser accedidas desde cualquier parte de la aplicación sin necesidad de crear una instancia de la clase.

Ejemplo básico:



En este ejemplo, contador es una variable estática que puede ser accesible desde cualquier parte de la aplicación sin necesidad de instanciar la clase variableGlobal.

Se puede acceder y modificar esta variable de la siguiente manera:



Beneficios y peligros del uso de variables globales

Beneficios:

- **Acceso centralizado a datos comunes:** Las variables estáticas o "globales" son útiles cuando se necesita compartir datos entre varias clases sin necesidad de pasar parámetros constantemente.
- **Simplicidad en programas pequeños:** En aplicaciones pequeñas o de un solo propósito, puede ser más simple implementar variables globales para datos que deben ser accedidos por múltiples partes del código.

Peligros:

- **Difícil depuración:** Al estar disponibles en todas partes, puede ser complicado rastrear qué partes del código modifican una variable global, lo que lleva a posibles errores de concurrencia o valores inesperados.
- **Mala práctica en diseño de software:** El abuso de variables globales va en contra de los principios de encapsulación y diseño orientado a objetos. El acceso sin restricciones a variables globales puede hacer que el código sea más difícil de mantener y escalar.
- **Problemas de concurrencia:** En aplicaciones multihilo, las variables estáticas pueden llevar a condiciones de carrera si varios hilos intentan leer y modificar la variable simultáneamente sin control de acceso adecuado.

Cuando usar variables globales

El uso de variables globales en C# debe hacerse con cuidado, ya que, si bien son útiles para compartir datos entre diferentes partes del código, también pueden causar problemas si se usan de manera inapropiada. Aquí hay algunos casos donde puede ser adecuado usarlas:

1. **Datos compartidos en toda la aplicación:** Si tienes datos que deben ser accesibles y modificables por varias clases o métodos en toda la aplicación (por ejemplo, un contador de usuarios conectados).
2. **Configuraciones globales:** Parámetros de configuración que rara vez cambian, como una cadena de conexión a una base de datos o ajustes de configuración que son constantes para la aplicación.

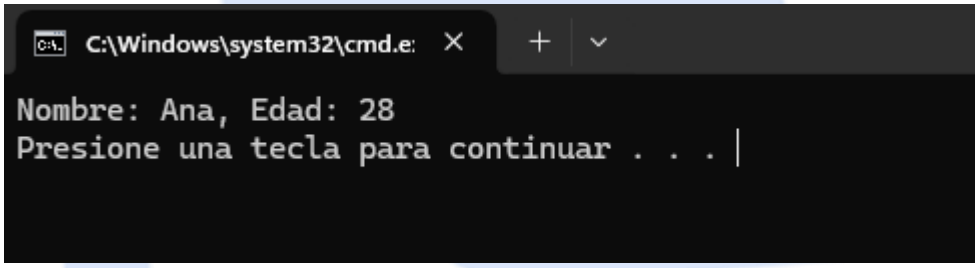
2. MÉTODO QUE RETORNA UNA CADENA DE TEXTO

Este método devuelve un valor de tipo string, que es una cadena de caracteres.

```
namespace ConsoleApp1
{
    3 referencias
    public class Persona
    {
        // Atributos privados
        private string nombre;
        private int edad;

        // Constructor para inicializar la persona
        1 referencia
        public Persona(string nombre, int edad)
        {
            this.nombre = nombre;
            this.edad = edad;
        }

        // Método público que retorna un string
        1 referencia
        public string ObtenerDetalles()
        {
            return "Nombre: " + nombre + ", Edad: " + edad;  // Retorna una cadena
        }

    }
}
```

```
using System;

namespace ConsoleApp1
{
    0 referencias
    internal class Program
    {

        0 referencias
        static void Main(string[] args)
        {
            Persona persona = new Persona("Ana", 28);
            string detalles = persona.ObtenerDetalles();
            Console.WriteLine(detalles);  // Imprime: "Nombre: Ana, Edad: 28"

        }
    }
}

```

Resultado esperado



INSTITUTO

KHIPU