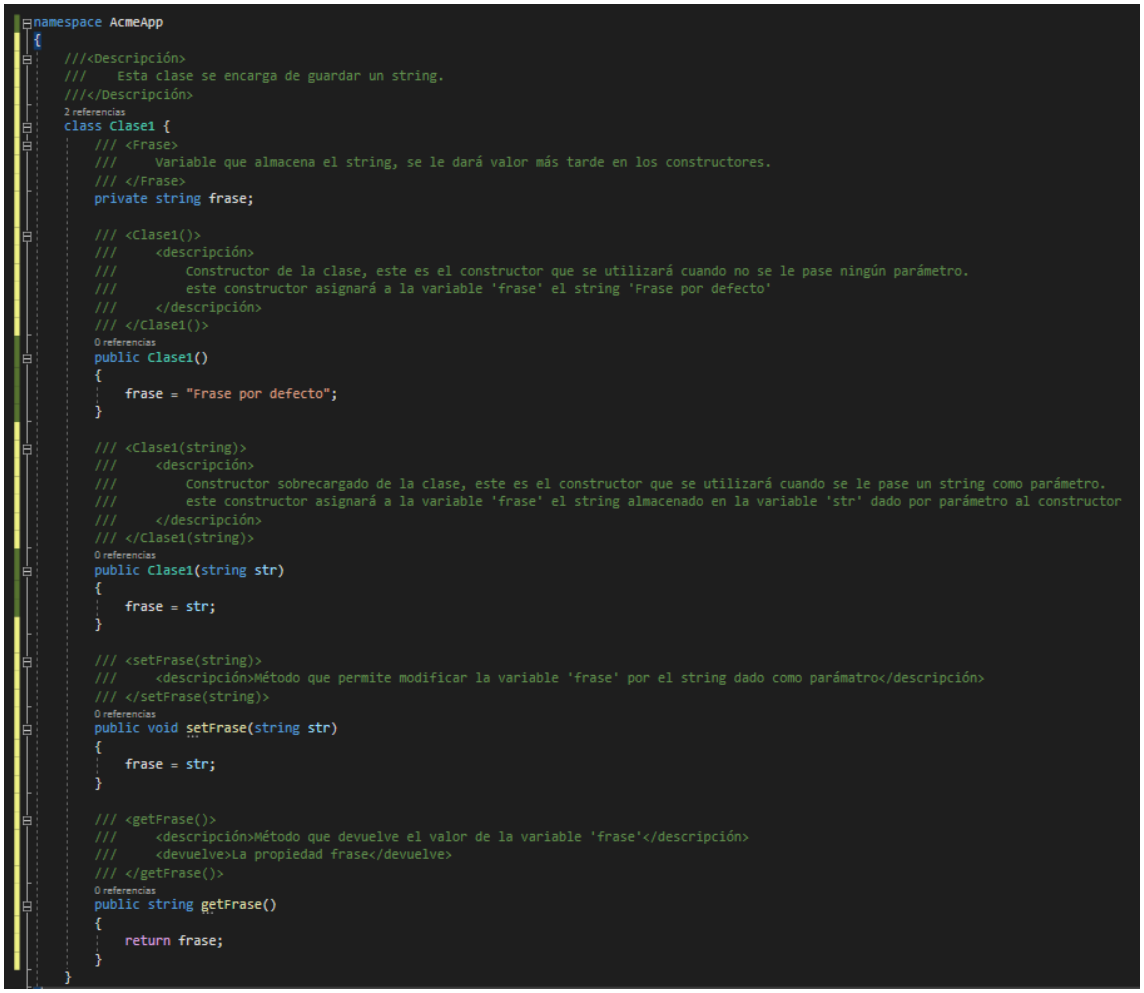
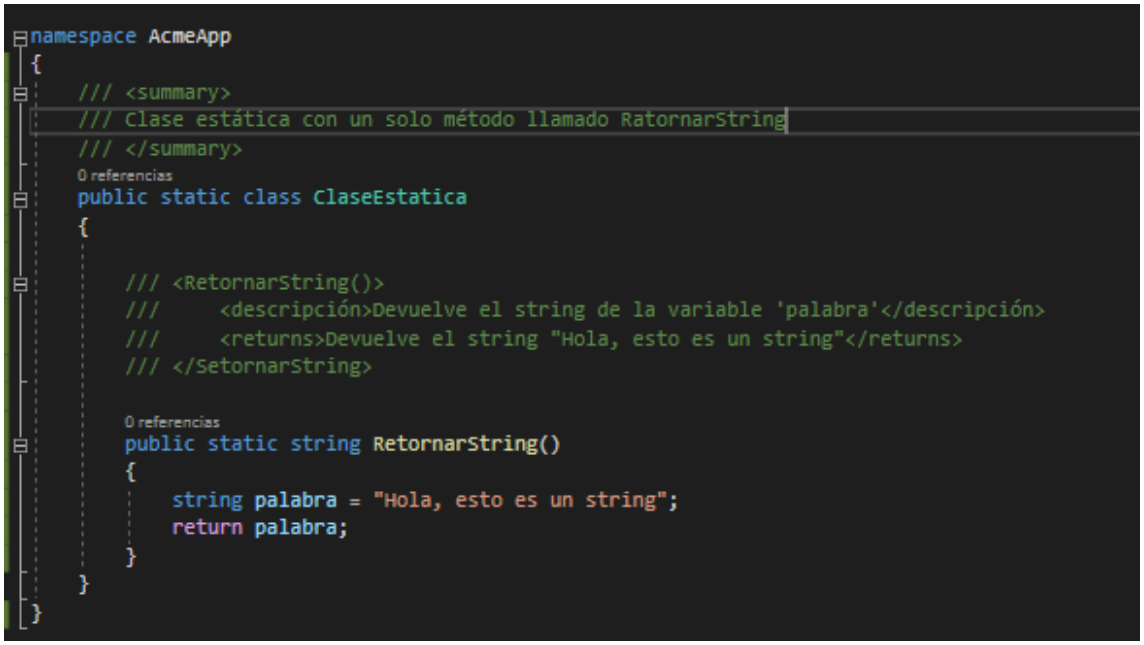
Autor: Mark Bort Tomás 1ºDAM

Clase con dos constructores.



```
namespace AcmeApp
{
    ///<Descripción>
    ///    Esta clase se encarga de guardar un string.
    ///</Descripción>
    2 referencias
    class Clase1 {
        /// <Frase>
        ///     Variable que almacena el string, se le dará valor más tarde en los constructores.
        /// </Frase>
        private string frase;

        /// <Clase1()>
        ///     <descripción>
        ///         Constructor de la clase, este es el constructor que se utilizará cuando no se le pase ningún parámetro.
        ///         este constructor asignará a la variable 'frase' el string 'Frase por defecto'
        ///     </descripción>
        /// </Clase1()>
        0 referencias
        public Clase1()
        {
            frase = "Frase por defecto";
        }

        /// <Clase1(string)>
        ///     <descripción>
        ///         Constructor sobrecargado de la clase, este es el constructor que se utilizará cuando se le pase un string como parámetro.
        ///         este constructor asignará a la variable 'frase' el string almacenado en la variable 'str' dado por parámetro al constructor
        ///     </descripción>
        /// </Clase1(string)>
        0 referencias
        public Clase1(string str)
        {
            frase = str;
        }

        /// <setFrase(string)>
        ///     <descripción>Método que permite modificar la variable 'frase' por el string dado como parámatro</descripción>
        /// </setFrase(string)>
        0 referencias
        public void setFrase(string str)
        {
            frase = str;
        }

        /// <getFrase()>
        ///     <descripción>Método que devuelve el valor de la variable 'frase'</descripción>
        ///     <devuelve>La propiedad frase</devuelve>
        /// </getFrase()>
        0 referencias
        public string getFrase()
        {
            return frase;
        }
    }
```

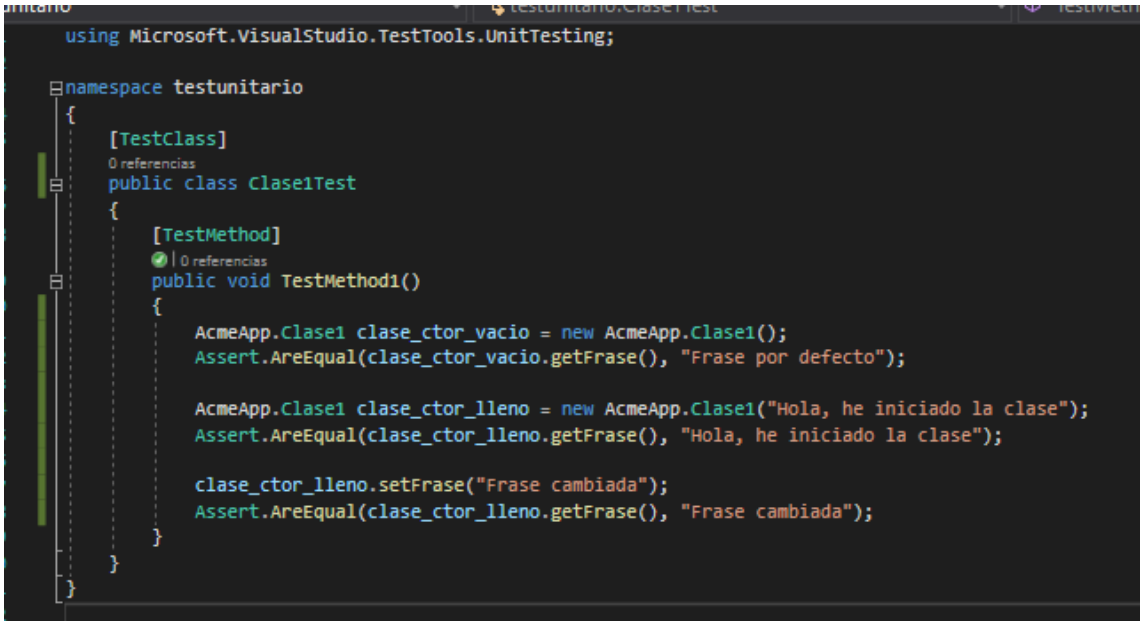
Clase estática.



```
namespace AcmeApp
{
    /// <summary>
    /// Clase estática con un solo método llamado RatornarString
    /// </summary>
    0 referencias
    public static class ClaseEstatica
    {

        /// <RetornarString()>
        ///     <descripción>Devuelve el string de la variable 'palabra'</descripción>
        ///     <returns>Devuelve el string "Hola, esto es un string"</returns>
        /// </SetornarString>

        0 referencias
        public static string RetornarString()
        {
            string palabra = "Hola, esto es un string";
            return palabra;
        }
    }
}
```

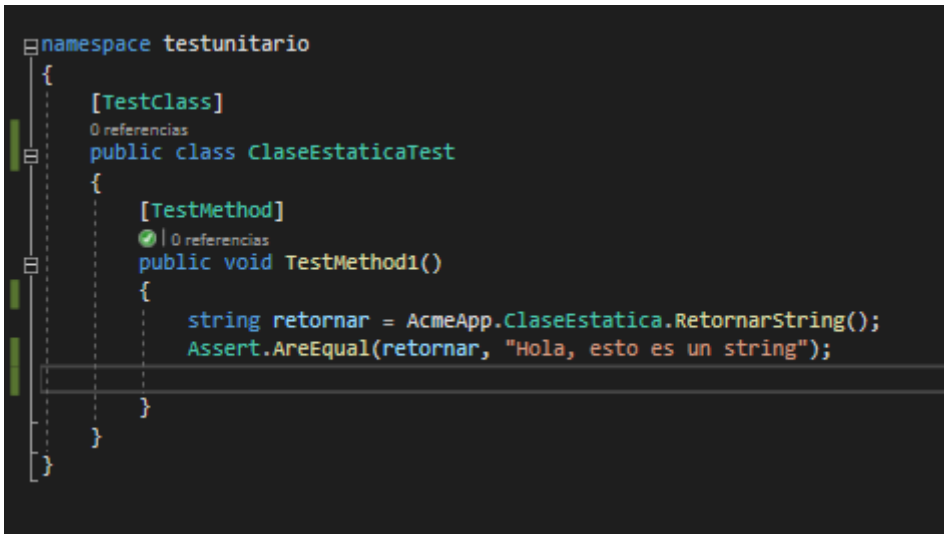
Autor: Mark Bort Tomás 1ºDAM

Prueba unitaria clase con dos constructores



```
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace testunitario
{
    [TestClass]
    0 referencias
    public class Clase1Test
    {
        [TestMethod]
        0 referencias
        public void TestMethod1()
        {
            AcmeApp.Clase1 clase_ctor_vacio = new AcmeApp.Clase1();
            Assert.AreEqual(clase_ctor_vacio.getFrase(), "Frase por defecto");

            AcmeApp.Clase1 clase_ctor_lleno = new AcmeApp.Clase1("Hola, he iniciado la clase");
            Assert.AreEqual(clase_ctor_lleno.getFrase(), "Hola, he iniciado la clase");

            clase_ctor_lleno.setFrase("Frase cambiada");
            Assert.AreEqual(clase_ctor_lleno.getFrase(), "Frase cambiada");
        }
    }
}
```

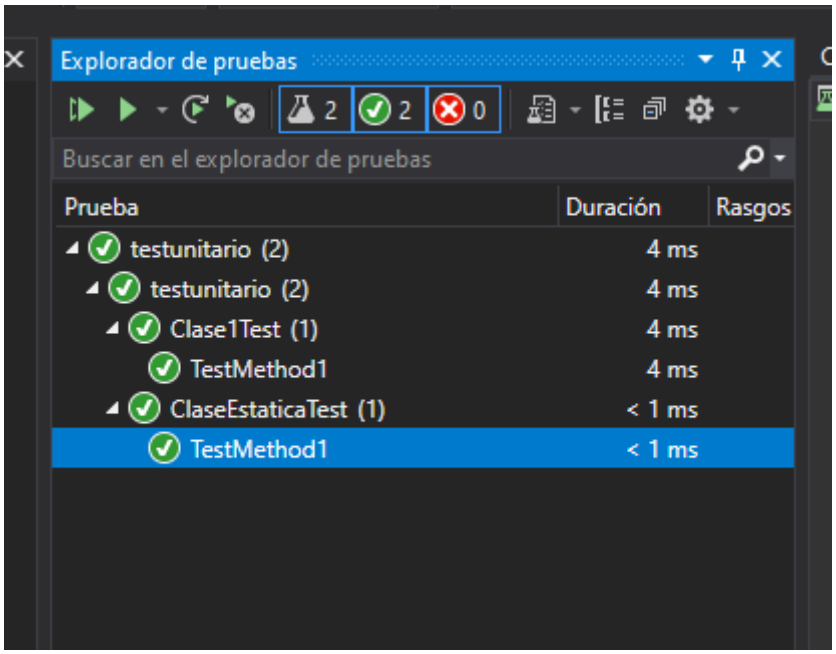
Prueba unitaria clase estática.



```
namespace testunitario
{
    [TestClass]
    0 referencias
    public class ClaseEstaticaTest
    {
        [TestMethod]
        0 referencias
        public void TestMethod1()
        {
            string retornar = AcmeApp.ClaseEstatica.RetornarString();
            Assert.AreEqual(retornar, "Hola, esto es un string");

        }
    }
}
```

Autor: Mark Bort Tomás 1ºDAM

Test



Estructura final de la solución