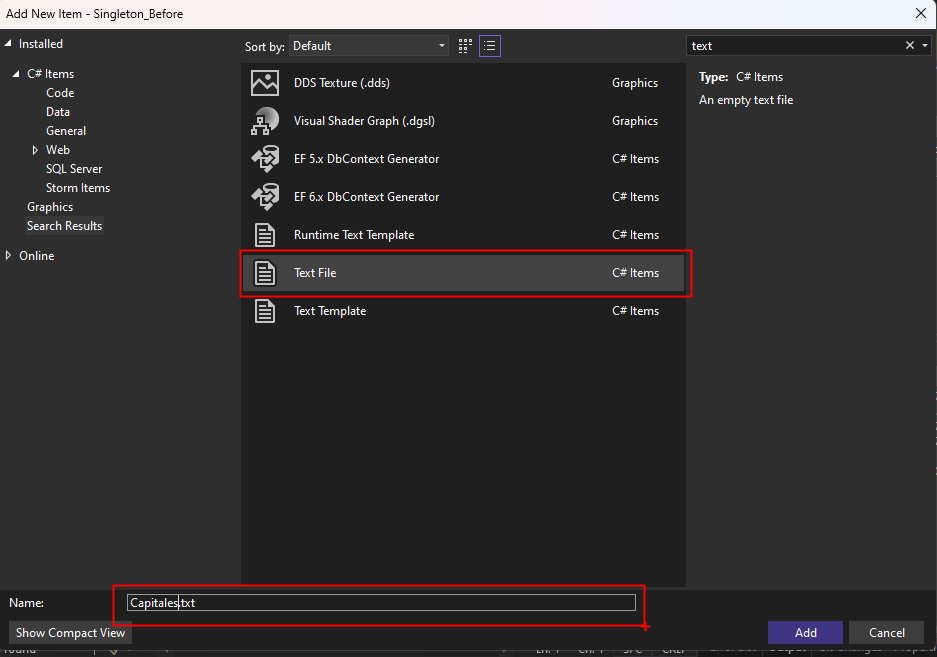
**SINGLETON**

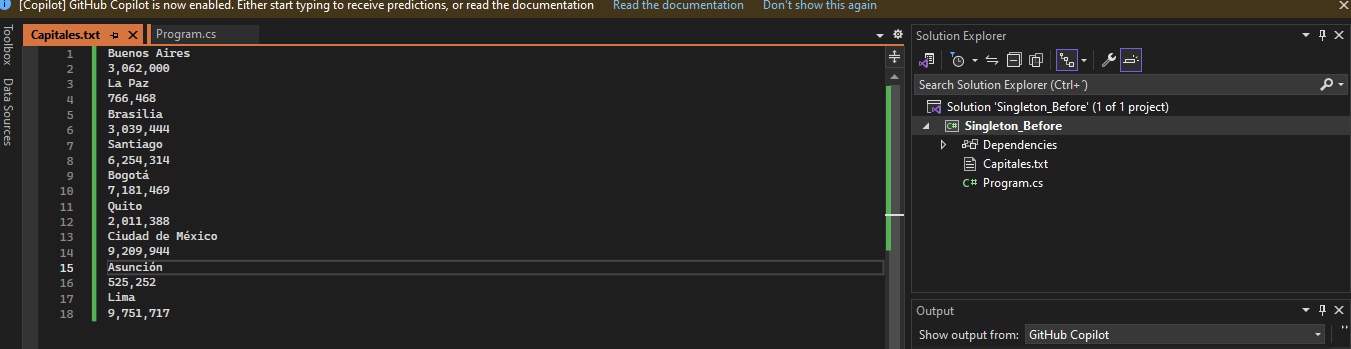
Su función principal es crear una única instancia de un objeto y esta poder compartirla en toda la aplicación

**EJERCICIO1:**

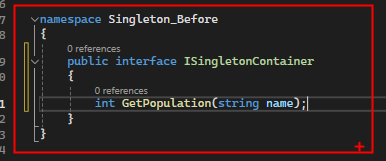
El ejercicio que vamos a tocar es la implementación de un programa el cual pueda leer desde un archivo las ciudades y su número de habitantes.

Lo primero que haremos será crear un archivo de texto **Capitales.txt:**

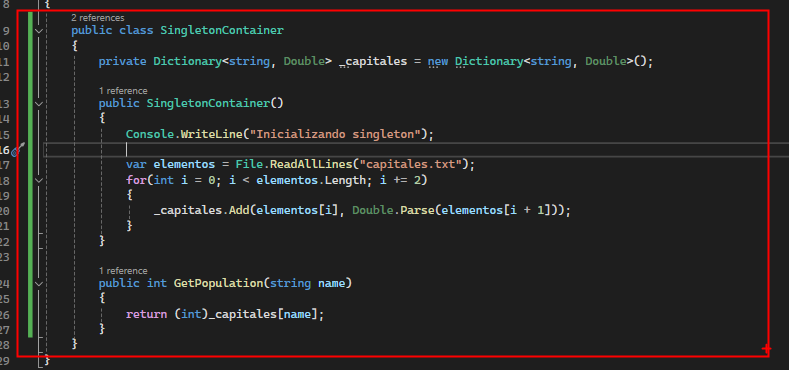
****



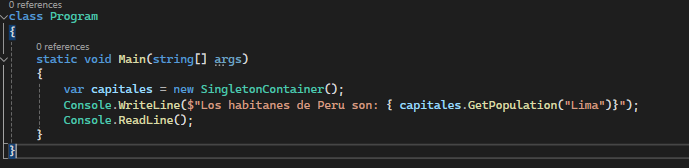
Seguidamente lo que haremos es crear una interfaz que definirá un método que me permitirá calcular el número de la población de una respetiva ciudad: **ISingletonContainer**

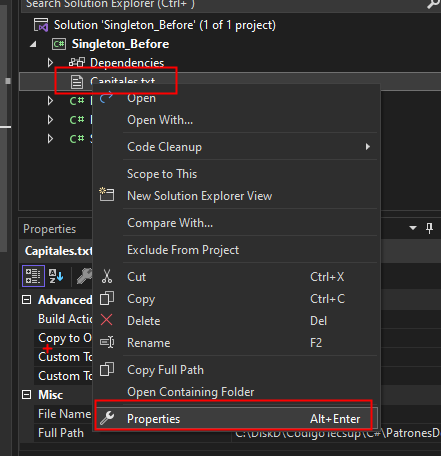
****

Y luego realizamos la implementación de dicha interfaz:

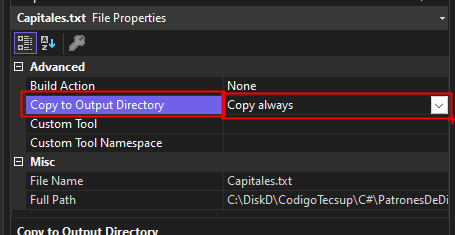


Ejecutamos el proyecto y nos aparecerá el siguiente error de que no se podrá encontrar el archivo, entonces vamos a las propiedades del archivo de texto:





Y seleccionamos:



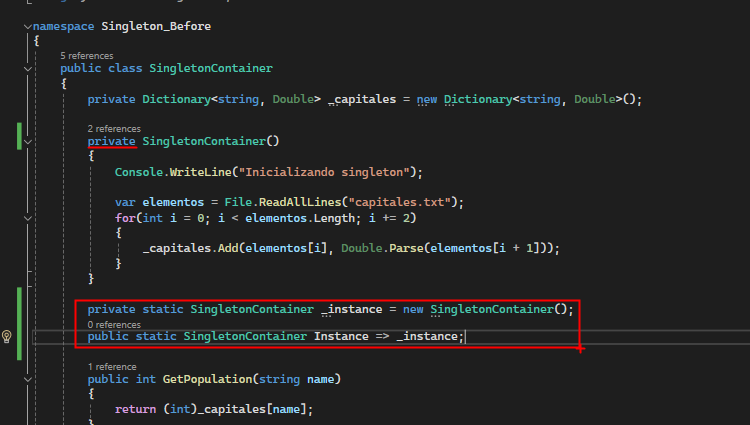
**Ahora aplicaremos el patrón** **Singleton**

**Abrimos el archivo Singleton\_After**

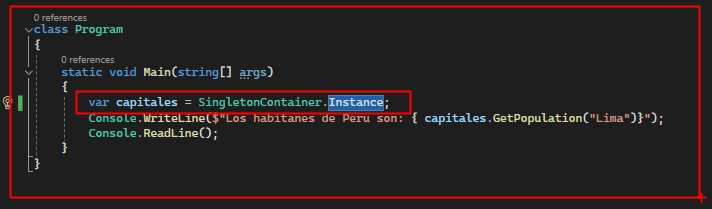
El cual nos indica que solo podemos crear una instancia de una clase:

Lo primero que haremos es poner nuestro constructor como privado, entonces de alguna forma tendríamos que exponer nuestra instancia para que todo siga funcionando normalmente

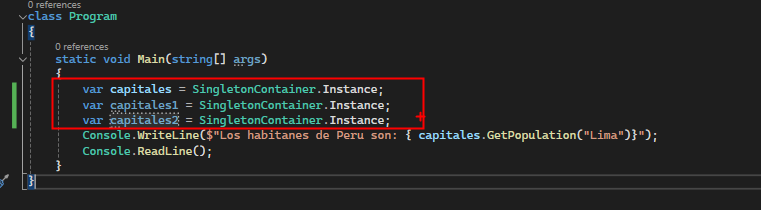
Entonces vamos a instanciar una clase estatica, y ahí la ponemos como privada e instancioamos nuestra nueva clase **(new SingletonContainer())**



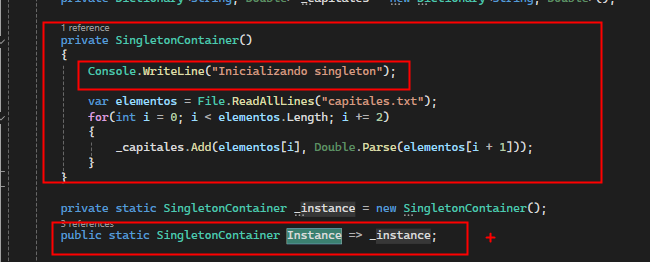
Y en el **Program** modificamos:

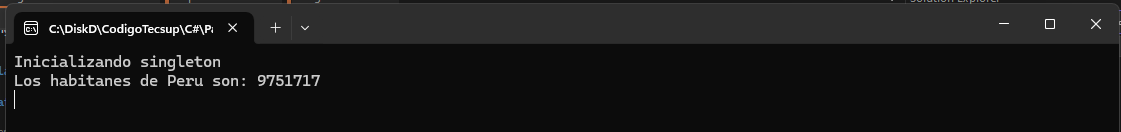


Ahora gregamos estas instancias haber si nuestro patrón de diseño **Singleton esta funcionando** y solo llama una vez a la instancia:



Vemos si entra una sola vez:



Y vemos que independientemente de que llame 3 veces a la instancia solo entra, se ejecuta una sola vez en sí: 

Otro cambio que podríamos realizar en nuestra implementación es realizar nuestras instancias **Lazy -> esto nos permite crear instancias solo cuando se necesite, es decir solo cuando llame a este método Instance** de esta forma le damos más seguridad a nuestra aplicación:

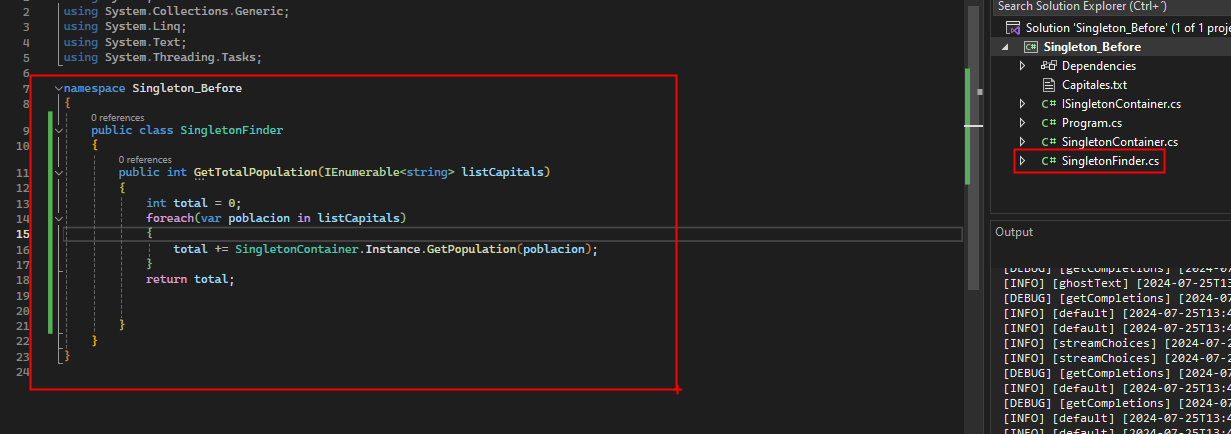


**INCONVENIENTES**

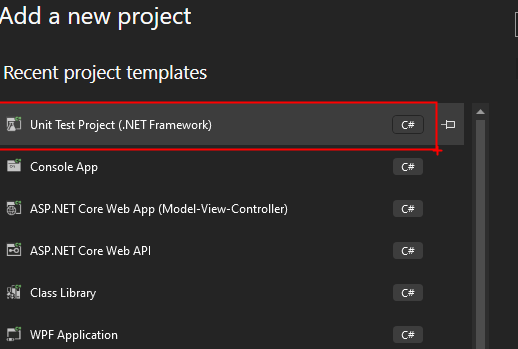
Por ejemplo, supongamos que necesitamos hacer pruebas dentro de nuestra aplicación y se nos pide que creemos una clase la cual implementa una función que nos retorne el total de población de la ciudad que proporcionemos

Abrimos el archivo **Singleton\_After\_B:**

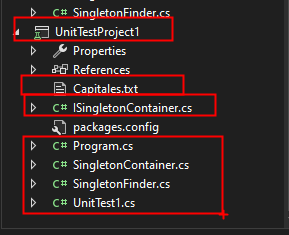
Y ahora creamos una clase que nos permita hacer la búsqueda y obtener las poblaciones de lo ingresado



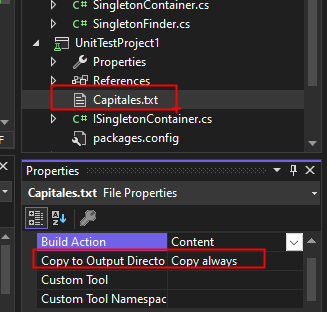
Ahora seguidamente vamos a crear un nuevo proyecto en nuestra solución de tipo prueba unitaria



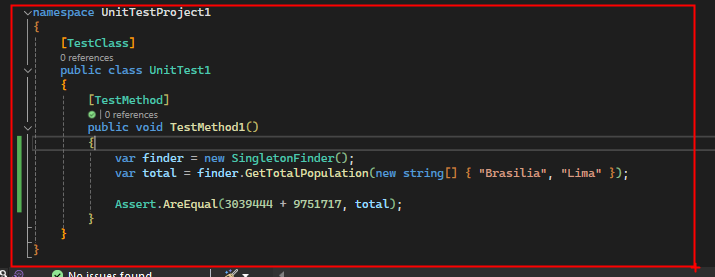
Como es de tipo .**Net** **framework** es probable que no reconozca las referencias que quisiéramos asociar por ello vamos a llevar todas las clases a este nuevo proyecto, para fines de este ejercicio:



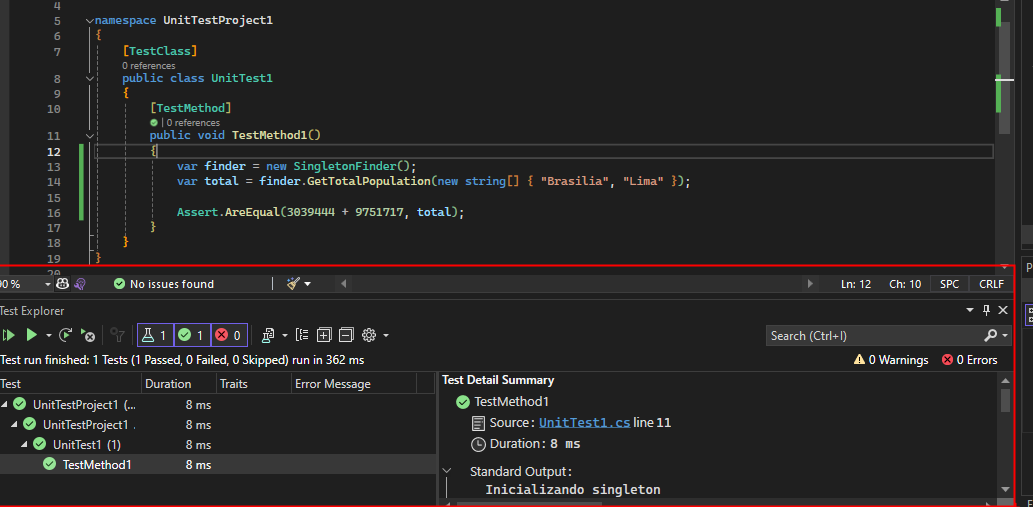
Y no olvidar modificar esto en **capitales.txt:**



Por ultimo implementar nuestra clase de prueba unitaria:



Y ejecutarla:

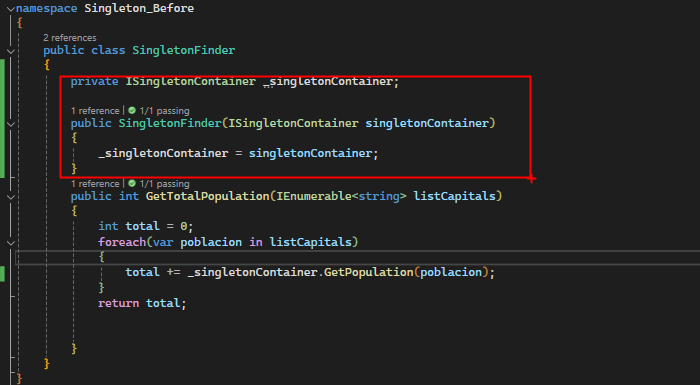


Ahora es necesario entender que utilizar una base de datos de prd en pruebas unitarias es una mala práctica.

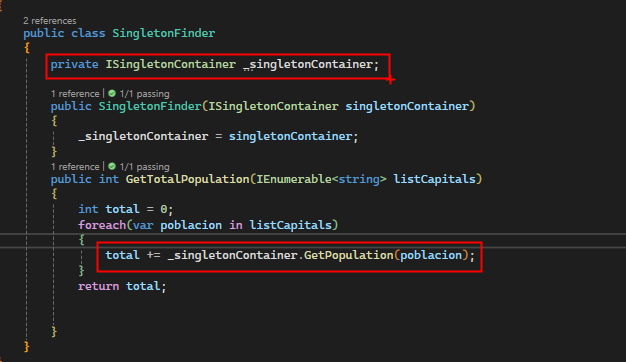
**Abrimos ahora nuestro archivo Singleton\_After\_C -> y utilizamos el archivo de prueba unitaria(ahí trabajremos)**

Ahora es necesario entender que utilizar una base de datos de prd en pruebas unitarias es una mala práctica.

Entonces para solucionar dicho problema vamos a utilizar inyección de dependencias, entonces en lugar de acceder directamente a la instancia de **SingletonContainer** la cual accede al listado de prd de **capitales.text** en lugar de eso vamos a utilizar nuestra interfaz **ISingletonContainer** y la vamos a inyectar en nuestro contructor de **SingletonFinder**

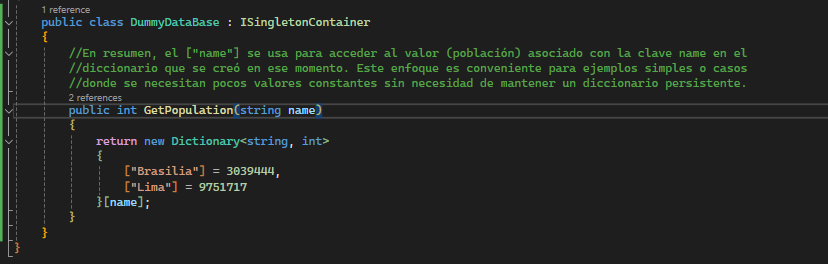


Y en vez de acceder directamente a la instancia de **SingletonContainer** accedemos a la inyección de dependencia:

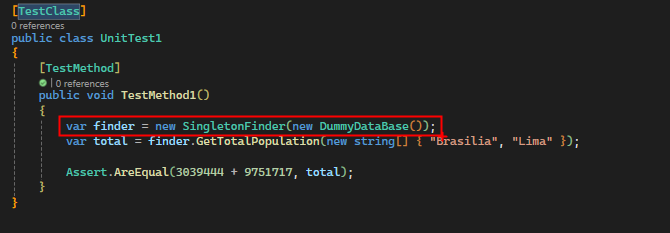


Ahora nos dirigimos a nuestra clase **UnitTest1** para realizar las modificaciones:

1. Primeramente, crearemos una clase de nombre **DummyDataBase** que implementara **ISingletonContainer** y en esta clase lo que haremos es simular un base de datos(datos falsos) con un diccionario:



Y ahora en nuestra clase **UnitTest1** tenemos que pasarle la inyección de la dependencia que vamos a usar en nuestro **SingletonFinder** de esta forma no estaríamos dependiendo directamente de una base de datos de prd:



Ejecutamos nuestra prueba unitaria y verificamos que funcione correctamente.