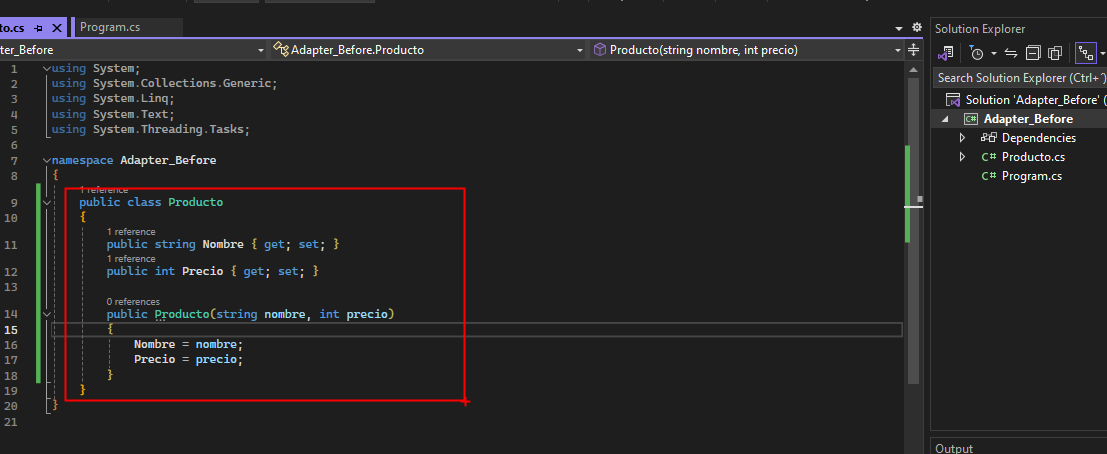
**ADAPTER**

**Ejercicio1:**

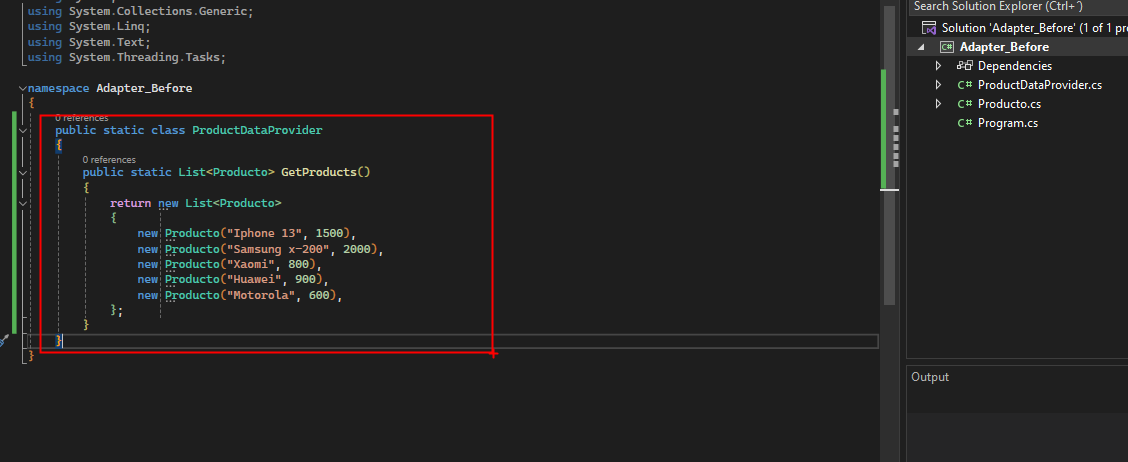
Lo que se encarga este patrón es comunicar interfaces que son distintas para que puedan trabajar entre sí.

Vamos a ver un ejemplo en donde estamos desplegando información atraves de un formato json de productos, vamos a crear una pequeña aplicación y veremos como podemos integrar este patrón de diseño **adapter**.

1. Creamos nuestra clase Producto



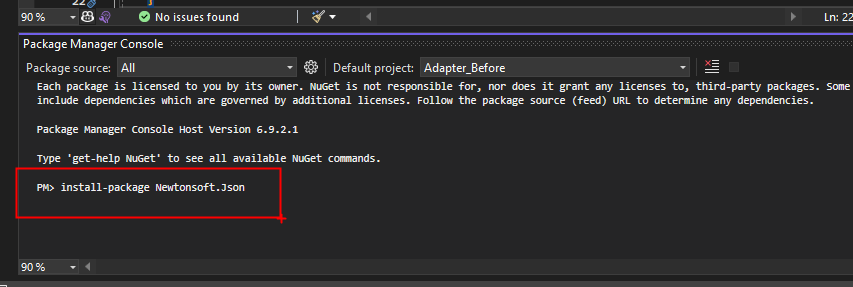
1. Creamos datos para los productos, una lista de celulares con sus respectivos precios

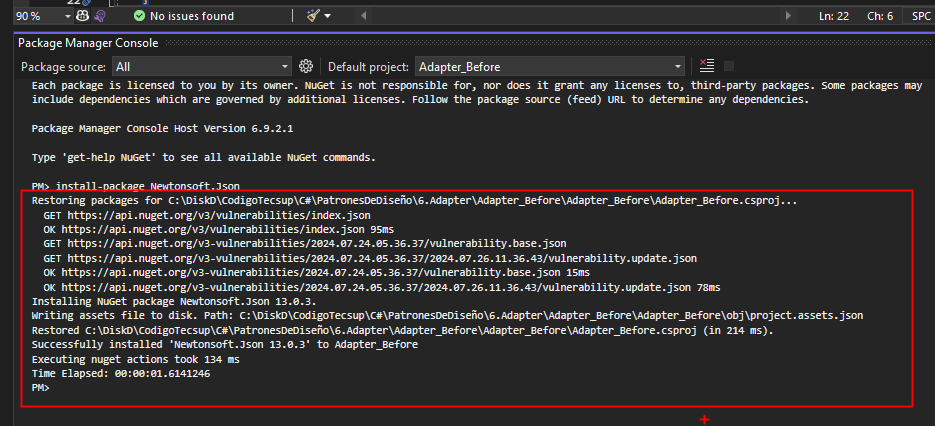
****

1. Lo que se hara en este ejercicio es imprimir los productos en un formato json

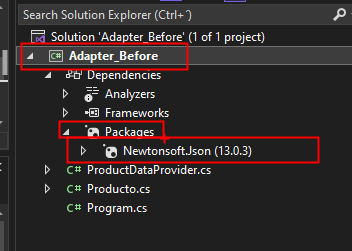
Para ello utilizaremos la librería de .net **newtonSoft.Json** para poder convertir dichos datos a un formato json

Veremos otra forma de instalar paquetes con el **package manager console:**

****

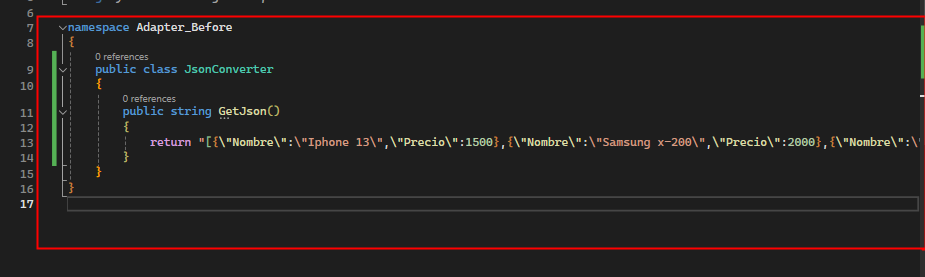


Y verificamos que este:

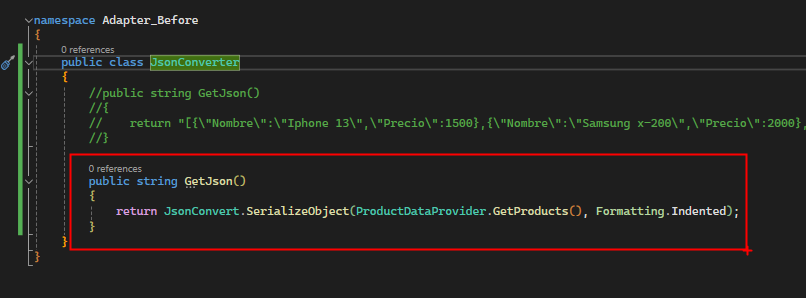


1. Lo siguiente que haremos es crear la clase de nombre **JsonConverter** que tendrá dicha responsabilidad de convertirlo en formato json

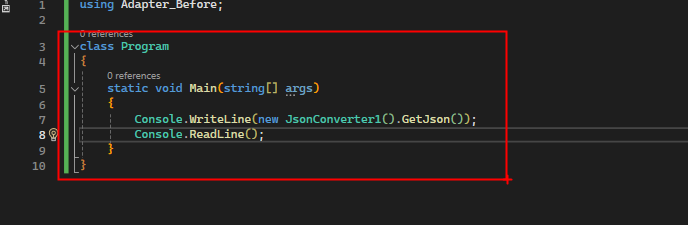
Si no tuviéramos esta librería Json tendríamos que convertirlo de forma manual asi:

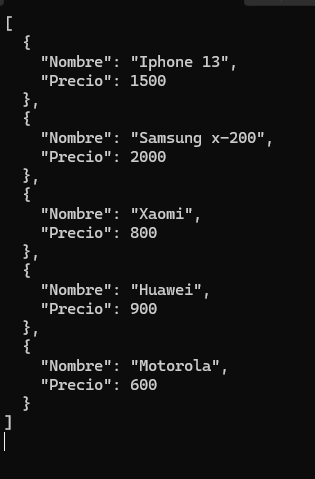


Pero como tenemos dicha librería **newtonSoft.Json** lo hacemos asi:



Y modificamos nuestro **Program.cs**

****



**Abrimos nuestro archivo Adaper\_After**

Inconveniente:

Ahora resulta que uno de los proveedores nos puede entregar la información pero en formato **XML.**

Pero nosotros en nuestro ejercicio no tenemos esa opción de convertir **XML** en formato **Json**

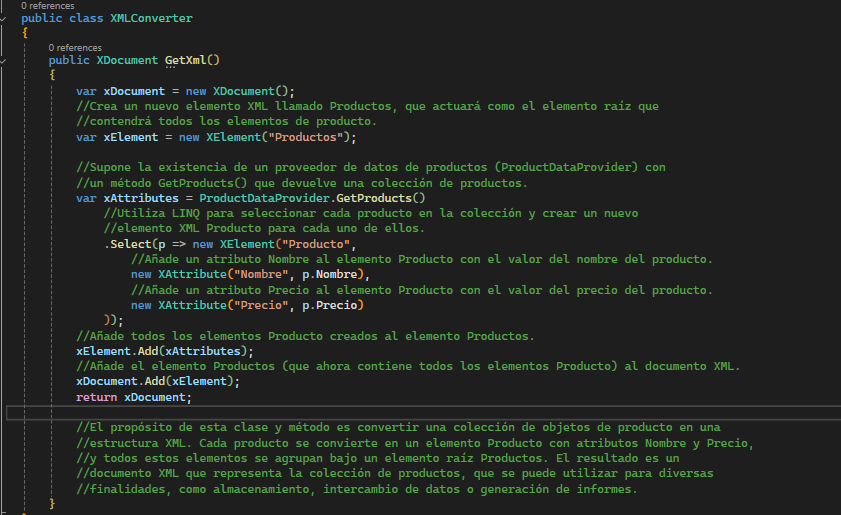
**Para eso utilizaríamos el patrón Adapter para poder convertir lo que nos entrega nuestro proveedor en formato XML y convertirlo asi en un formato Json que es con el que trabajamos**

**Antes de trabajar con un documento XML necesitamos saber que es:**

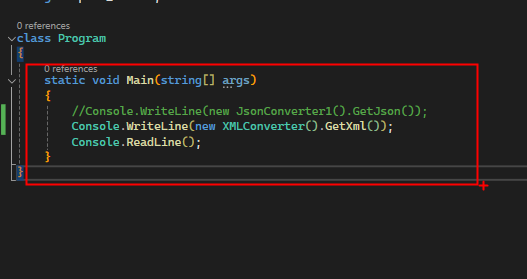
Un documento XML (Extensible Markup Language) es un formato de texto utilizado para almacenar y transportar datos. Es un estándar ampliamente adoptado que define una estructura jerárquica de elementos con etiquetas y atributos, permitiendo representar datos de manera legible tanto para humanos como para máquinas.

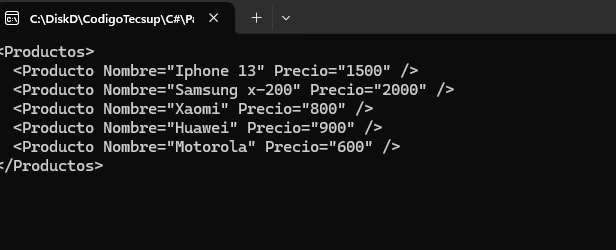
**Características de XML**

1. **Estructura Jerárquica**: XML organiza los datos en una estructura de árbol. Cada elemento puede tener subelementos, creando una jerarquía.
2. **Etiquetas Personalizadas**: A diferencia de HTML, XML no tiene un conjunto fijo de etiquetas. Las etiquetas se definen según las necesidades de los datos que se representan.
3. **Autodescriptivo**: XML es autodescriptivo porque las etiquetas proporcionan información sobre el tipo de datos que contienen.
4. **Portabilidad**: XML es un formato de texto, lo que lo hace fácilmente transportable y compatible con diferentes sistemas y plataformas.
5. **Extensible**: Puedes crear tus propias etiquetas y estructuras, lo que lo hace muy flexible para representar datos complejos.
6. Creamos nuestra clase **XMLConverter**

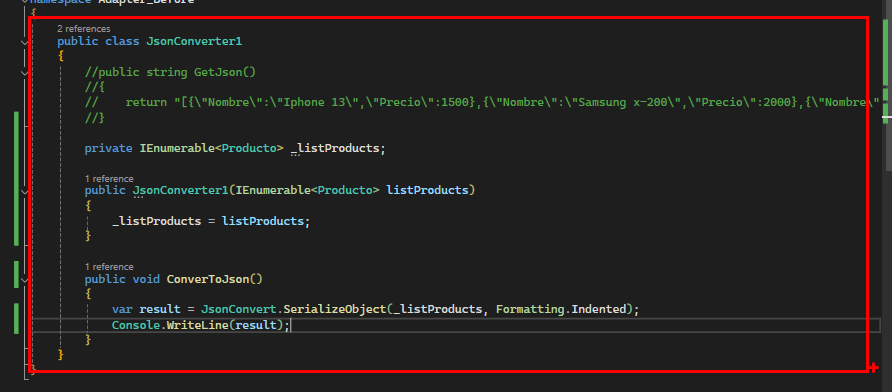


Probamos este archivo **XMLConvert**



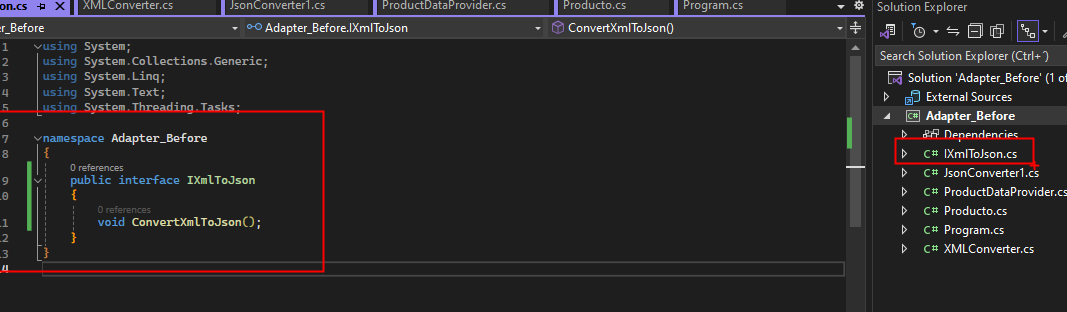


Ahora modificamos nuestra clase **JsonConverter1:**

****

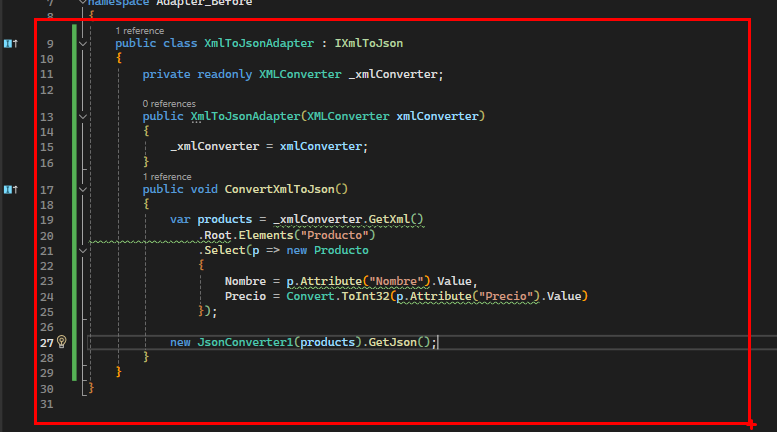
Lo siguiente que tenemos que hacer es que nuestro programa se siga interpretando como **Json**

Creamos primeramente la interfaz:



Ahora creamos dicha implementación

Creamos nuestra clase **XmlToJsonAdapter**:



Por ultimo modificamos nuestro **Program.cs**

