**PRINCIPIOS SOLID**

**Principio Abierto/Cerrado (OCP)**

**Ejercicio1**

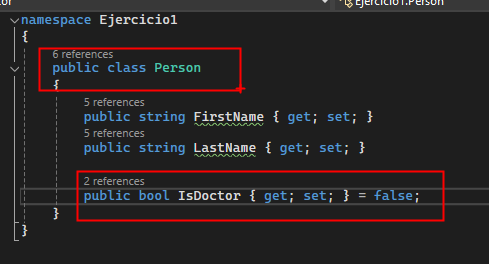
**BEFORE**

Vamos a tener una serie de personas las cuales van a ser postulantes a un trabajo en un hospital. Para ello primeramente definiremos una clase **Person** con los campos: **FirstName y LastName**. Luego crearemos nuestra clase **Staff** con los campos: **FirstName, LastName y Email** luego en una clase **AccountService,** en el método **Create** crearemos el usuario de dicha persona y su correo institucional para cada nuevo empleado.

**1.1 Primer requerimiento**

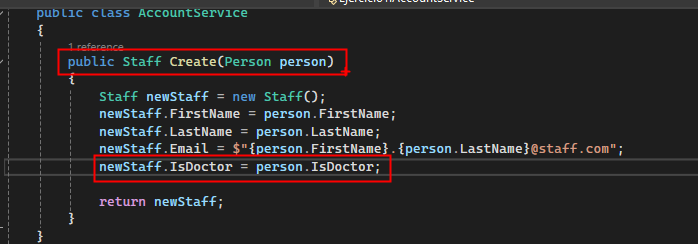
Ahora lo que necesitamos implementar es si el nuevo empleado es un doctor o no lo es:

Entonces vamos agregar una condición en nuestra clase **Person** si es doctor o no:

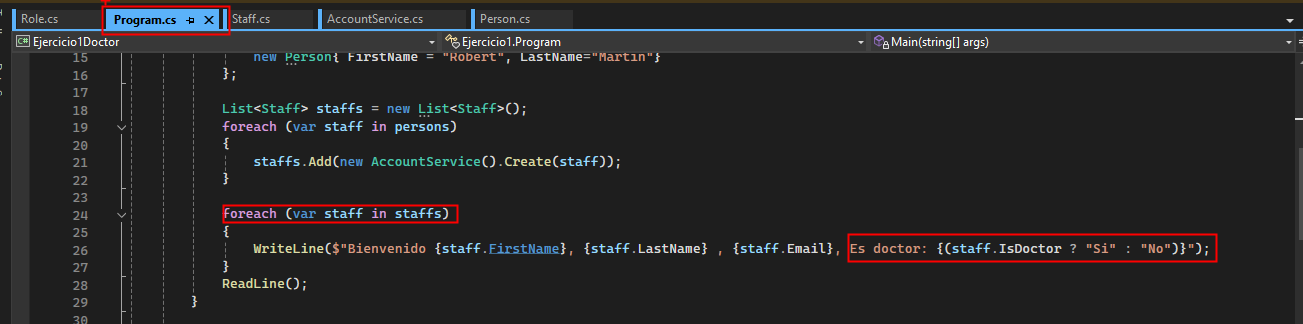


Lo mismo tendríamos que hacer en nuestra clase **Staff**

Y hacemos lo propio en nuestro método **Create**:

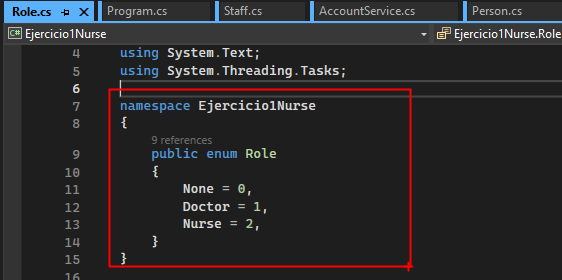


Y lo imprimimos en nuestro **program**:

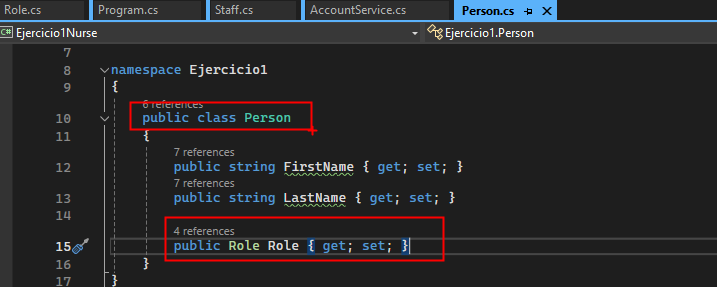


**1.2 Segundo Requerimiento: verificar si es una enfermera si es un doctor tendrá correo especial y si es enfermera también tendrá un correo especial**

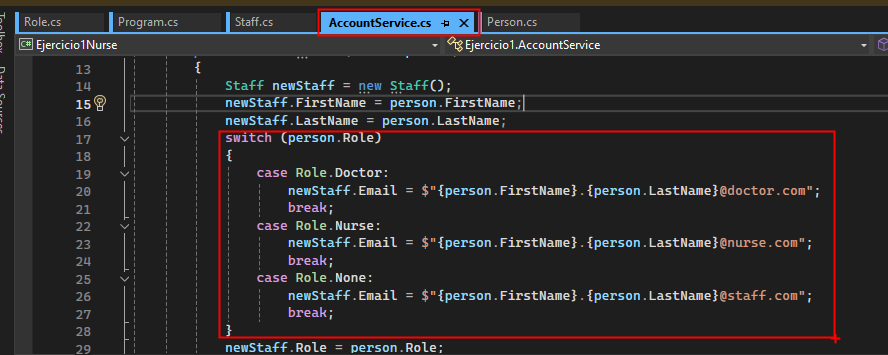
Entonces tenemos ahora dos requerimientos si es doctor o si es una enfermera, por ello crearíamos un rol,



y modificaríamos las clases de **PERSON**, **STAFF** y **ACCOUNTSERVICE** y posteriormente luego el **Program**

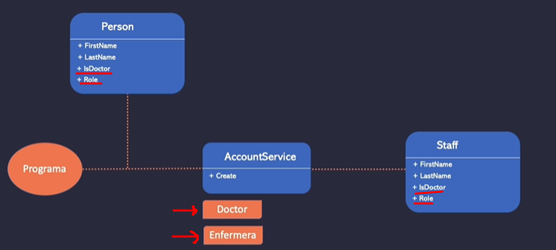


Por ultimo modificamos nuestra clase **AccountService**



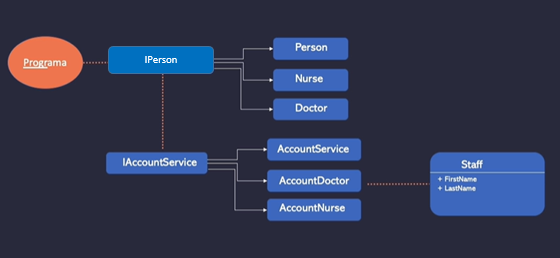
**1.3 Resumen de los dos requerimientos hasta ahora:**

**Si bien se cumplió con los dos primeros requerimientos se hizo varias modificaciones en las clases PERSON, STAFF y ACCOUNTSERVICE**



**AFTER**

Lo correcto sería hacer este tipo de implementación

****

En conclusión, nos permite realizar modificaciones en código que ya está funcionando, sino que simplemente podamos extender código y eso se hace atraves de interfaces por ello trabajar con interfaces nos permite tener flexibilidad.

**EJERCICIO 2**

**BEFORE**

Realizar el cálculo del área de diferentes objetos que se tengan, para este ejercicio tendremos una clase rectángulo y una clase circulo.

Como vemos se tiene que implementar diferentes **“if”** si queremos ir aumentando más formas geométricas, lo cual no es correcto ya que viola nuestro principio.

**AFTER**

Realizar el refactor utilizando dicho principio

**EJERCICIO 3**

**BEFORE**

Realizar la creación de un ejercicio que implemente una clasificación de animales por su **Class** y por su orden **Order,** para ello primeramente definir la clase **Animal** la cual tendrá como propiedades: un **nombre**, la clase **Class** y el **Order,** asi mismo será necesario crear los **enums** de **clases** de animales (mamíferos, peces, aves, reptiles) y **order** de animales (carnívoro, herbivoro y omnívoro), luego crear la clase **Animalfilter** que se encargará de hacer el filtrado de los animales ingresados por su **clase**. Crear una lista de animales en el **Program** y mostrar su nombre, su clase y su orden, y posteriormente filtrarlos por su clase

**AFTER**

Hacer el Refactor correspondiente aplicando dicho principio