**PRINCIPIOS SOLID**

**Principio de Responsabilidad Única (SRP)**

**Ejericio1.**

**BEFORE**

Vamos a crear una aplicación de citas médicas donde necesitamos crear una clase **Appointment service**, y crear un método dentro de dicha clase, donde le pasemos por argumentos un nombre, un email, y una fecha, a su vez tenemos que hacer las validaciones correspondientes de cada parámetro, luego si todo es válido entonces se procederá a crear la cita médica.

Veremos que todo lo crearemos dentro de una sola clase -> lo cual no es correcto ya que viola el primer principio.

**AFTER**

1. Crear un objeto Paciente
2. Crear un objeto cita que tenga como parámetro el paciente y la fecha de la cita.
3. Creamos una clase **ValidationResult** para realizar las validaciones.
4. Crear una clase para las validaciones. -> **AppointmentServiceValidator**
5. Crear el método Create en la clase **AppointmentService**
6. Instanciamos nuestra clase Appointment

Ahora si, en conclusión, vemos como se ha refactorizado todo nuestro código con dicho principio.

**Ejercicio2.**

**BEFORE**

BEFORE

Crearemos un reporte de los cursos impartidos, creamos una clase **CourseReportEntry** donde vamos a almacenar todos los datos relacionados con reporte( **CourseReportEntry** -> Name: Nombre del curso. Students: Número de estudiantes inscritos. Rating: Valoración del curso.) luego vamos a crear una clase donde vamos a guardar o persistir los datos de ese reporte en la clase CourseReport y vamos a guardar dicho reporte en nuestro ambiente local.

**Clase CourseReport:**

* AddEntry(CourseReportEntry entry): Agrega un curso al reporte.
* RemoveEntryAt(int index): Elimina un curso del reporte basado en su índice.
* ToString(): Genera una representación en texto del reporte de cursos.
* SaveToFile(string directoryPath, string fileName, CourseReport report): Guarda el reporte en un archivo (responsabilidad de persistencia).

Actualmente, la clase **CourseReport** tiene dos responsabilidades:

* Manejar la lógica de los reportes de cursos.
* Guardar el reporte en un archivo (persistencia de datos).

**AFTER**

Analizamos todo aquello que no sea de responsabilidad única en nuestra clase **CourseReport** ya que esta solo se encargara de la persistencia de datos, por ello creamos otra clase **FileSaver** que será solo para guardar nuestro archivo

**Examen\_Ejercicio1**

**Before**

Vamos a crear una aplicación de reservas de hotel donde necesitamos crear una clase **BookingService**, y crear un método **Create** dentro de dicha clase, donde le pasemos por argumentos un **nombre**(string), un **email**(string), y una fecha de **check-in(**dateTime) y **check-out**(dateTime). Además, debemos hacer las validaciones correspondientes de cada parámetro **(nombre no vacío, email válido, fechas válidas).** Luego, si todo es válido, se procederá a crear la reserva del hotel.

Este ejercicio será desarrollado inicialmente dentro de una sola clase -> lo cual no es correcto ya que viola el primer principio de responsabilidad única (SRP). Posteriormente, se debe **refactorizar** el código para cumplir con el principio de Responsabilidad única(SRP).

**AFTER**

Implementar una clase **Usuario**, una clase **Booking**(definir las propiedades de la reserva), una clase **ValidationResult**, y una clase **BookingServiceValidator** y la clase que realiza la reserva **BookingService**