Grupo 3
UT5 - TA3
ADA I

Ejercicio 0:

Aplica SRP, porque se puede separar la construcción compleja de código de la lógica del producto.

No se viola ningún principio.

Ejercicio 1:

Utilizariamos un builder porque creemos que la construccion de un sandwich se puede hacer de mejor forma con este patron y dividir los pasos en tareas menos complejas.

Se podria una interfaz llamada ISandwich que seria

```
{
	SetBread(this)
	SetCheese(this)
	SetMeat(this)
	SetVegetables(this)
	SetCondiments(this)
}
```

```
@Inject ISandwich _sandwichBuilder

Var sandwich = new
Sandwich()._sandwichBuilder.SetBreat(“Blanco”).SetCheese(“Muzza”).SetMeat(“Jamon”).SetVe
getables(“Lechuga”).SetCondiments(“Mayonesa”);
```

Ejercicio 2:

Utililizariamos prototype porque al hacer copias del anterior se podría facilitar mucho la tarea.
Viola SRP

```
IClone
{
	Clone(this GameUnit)
}
```

@inject IClone _clone;

Var archer = new Archer();

Var archer2 = archer._Clone.Clone();

Ejercicio 3:

Dijo Jose que lo salteemos. Pero utilizariamos strategy aunque no sea un patron de creacion.

Ejercicio 4:

El libro no puede tener responsabilidad de prestarse y saber de a quienes se presto.
Entonces crearíamos una interfaz utilizando el patron singleton para llevar una unica lista de prestamos por libro. Cuando se presta un libro se agrega a la lista de prestado, no sea crea una nueva instancia de libro.

```
IPrestamo
{
        Prestar(string title, string Name)

        dict Prestamos(string title, List<string> Names)
}
```

@inject IPrestamo _prestamo

Book book = new Book();

_prestamo.Prestar(“Harry Potter”, “Francisco”);

Ejercicio 5:

Es un builder porque usa muchos parametros y algunos son opcionales.

```
ITravelPlan
{
        SetFlight(this)
        SetHotel(this)
        SetCarRental(this)
        SetActivities(this)
        SetRestaurants(this)
        ……..
}
```

Pasar las propiedades a TravelPlan

@inject ITravelPlan _travelPlan

TravelPlan plan1 = new()._travelPlan.SetFlight().SetHotel().....SetX();
Y se pueden usar todos o alguno o ninguno.

Ejercicio 6:

En singleton se quiere acceder a valores desde otras partes de la aplicacion.