**Principio de Segregación de Interfaces (ISP) -> Interfaces pequeñas que cumplan una función especifica**

**Ejercicio1**

**BEFORE**

Vamos a realizar un proyecto en el que desarrollaremos los métodos necesarios para implementar la funcionalidad de una impresora, como **Fax, Print, Scan,** a su vez implementaremos la interfaz **IPrinterTask** que definirá todos los métodos que los diferentes tipos de impresora implementaran, los diferentes tipos de impresora que realicen dichas funcionalidades dependiendo de su gama, seran: **HPLaserJetPrinter (realiza las 3 funcionalidades)** y la impresora: **SimplePrinter (esta impresora solo imprimirá y scanee)**

Como vemos tenemos un problema al heredar la interfaz **IPrinterTask** ya que me trae todos los métodos de las impresoras, para lo que al implementar mi clase **SimplePrinter** no es correcto ya que solo necesitar de dos métodos no de los 3.

**AFTER**

Lo que realizaremos primeramente será crear una interfaz por cada método para que así se pueda heredar a la clase que lo necesite independientemente cada método.

**Ejercicio2**

**BEFORE**

Desarrollaremos un ejercicio que primeramente implemente una interfaz de vehiculo con el nombre **IVehiculo** que definirá los métodos: **Move(void), GetNumberOfWheel(int), startEngine(int), stopEngine(int),** luego vamos a tener dos clases hijas **bicicleta** y **automóvil**, como vemos bicicleta no puede tener todos los métodos de la interfaz **IVehiculo.**

**AFTER**

Analizamos el ejercicio y podemos diferenciar que ahí parte exlucivamente para vehiculo y otra que es para cualquier unidad de transporte, los métodos exclusivamente de vehiculo serán **startEngine, stopEngine** para lo cual implementaremos una interfaz propia llamada **IEngineVehiculo** y la otra interfaz será **Ivehiculo** para los métodos propios de cualquier unidad de transporte con los métodos: **Move, GetNumberOfWheel.**