El **Principio de Responsabilidad Única (SRP)** establece que una clase debe tener una única razón para cambiar, es decir, debe ser responsable de un solo propósito. En el ejemplo de la clase Factura, se observa que tiene múltiples responsabilidades: cálculo del total, aplicación del IVA, cálculo de deducciones, entre otras.

Vamos a refactorizar el código para separar esas responsabilidades y cumplir con el SRP.

**1. Refactorización para aislar las responsabilidades**

Vamos a crear tres clases para aislar las responsabilidades:

1. **Clase Factura**: Será responsable de almacenar los datos de la factura y delegar los cálculos.
2. **Clase IVA**: Se encargará de calcular el IVA.
3. **Clase Deduccion**: Se encargará de calcular la deducción.

**Conclusión**

Con esta refactorización:

* Separamos las responsabilidades en clases diferentes: Factura es responsable de manejar los datos de la factura, IVA calcula el IVA, y Deduccion calcula las deducciones.
* Si el **importe de deducción** cambia basándose en el importe de la factura, solo modificamos la lógica en la clase Deduccion.
* Si el **IVA** cambia, se ajusta la constante en la clase IVA.
* Si el **código de la factura** no debe aplicar IVA, ya está manejado en la clase IVA, sin necesidad de modificar Factura.

Esto asegura que los cambios futuros solo afecten a la clase que tiene la responsabilidad directa, cumpliendo el **Principio de Responsabilidad Única**.