1. ¿Cumple el principio de Inversión de Dependencia?

La implementación actual no cumple con el Principio de Inversión de Dependencia (DIP). Este principio establece que:

* Los módulos de alto nivel no deben depender de módulos de bajo nivel; ambos deben depender de abstracciones.
* Las abstracciones no deben depender de detalles; los detalles deben depender de abstracciones.

En el caso de la clase Factura, esta depende directamente de clases concretas como Deduccion e Iva. Esto significa que si alguna de estas clases cambia, también se tendría que modificar la clase Factura, lo que viola el principio de inversión de dependencias.

**Problemas en el diseño actual:**

* **Dependencia directa en clases concretas**: La clase Factura crea instancias de las clases Deduccion e Iva dentro del método calcularTotal, lo que genera un acoplamiento fuerte entre Factura y estas clases.
* **Dificultad para extender y probar**: Reemplazar o modificar el comportamiento de las clases Deduccion o Iva no es sencillo sin realizar cambios en la clase Factura.

**Conclusión**

Después de una refactorización, la clase Factura sí cumpliría con el Principio de Inversión de Dependencia (DIP) al:

* Depender de interfaces (como Deduccion e Iva) en lugar de implementaciones concretas.
* Permitir que las implementaciones de Deduccion e Iva puedan cambiar sin afectar a la clase Factura.
* Hacer el diseño más extensible y fácil de probar, ya que las implementaciones concretas pueden ser reemplazadas por otras nuevas o por mocks para pruebas.