**Taller 2 sobre SRP**

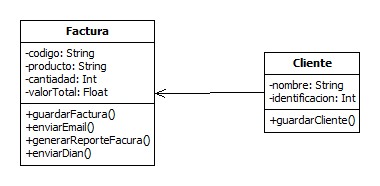
*Por: Santiago Duque Robledo*

Tenemos una clase llamada Factura y Cliente que representa un sistema de facturación. Esta clase tiene los siguientes métodos:

* guardarFactura(): Guarda información de la factura en la base de datos.
* enviarEmailFactura(): Envía un email a un cliente sobre la factura generada.
* guardarCliente(): Guarda la información del cliente en la base de datos.
* generarReporteFacturas(): Genera un reporte de las facturas realizadas en el mes actual.
* enviarFacturaDian(): Envía un email de la factura a la DIAN.

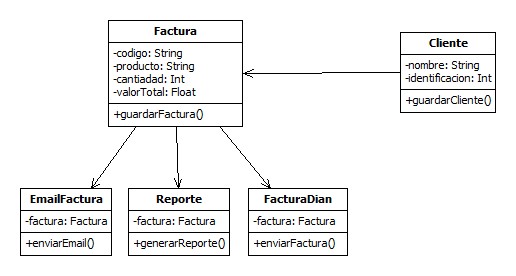
Tarea:

* Identifica las violaciones del SRP en la clase Factura: ¿Por qué esta clase tiene más de una responsabilidad?
* Refactoriza el código: Divide la clase Factura y Usuario en múltiples clases para cumplir con el SRP. Propón nombres adecuados para las nuevas clases y explica la razón de la separación.



*Solución:*

* La clase Factura tiene más de una responsabilidad porque combina múltiples tareas que pertenecen a diferentes dominios funcionales, como el envío de correos electrónicos, la generación de reportes, y el envío de información a la DIAN. Cada una de estas tareas podría cambiar por razones diferentes, lo que viola el Principio de Responsabilidad Única, haciendo que la clase sea difícil de mantener, probar y extender.



* La separación de la clase Factura en varias clases más pequeñas obedece a la necesidad de cumplir con SRP, que establece que una clase debe tener solo una responsabilidad bien definida. La clase original violaba este principio al abarcar múltiples responsabilidades. Al dividirla en clases específicas como Factura, EmailFactura, Reporte y FacturaDian, cada una se enfoca en una única tarea, lo que mejora la mantenibilidad, facilita las pruebas y permite futuras modificaciones sin afectar otras partes del sistema.