ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION

**Taller de Principios SOLID – Diseño de Software**

# Objetivos:

* Identificar violaciones a los principios SOLID.
* Corregir código que viole principios SOLID.

Una pequeña empresa de postres ha comenzado a armar el sistema de todo su negocio. Los primeros productos que han sacado a la venta son pasteles y helados.

Se sabe también que ya tienen como calcular los precios de cada producto que vende. Los productos tienen un costo parcial fijo, al cual se les aumenta el valor del IVA. A cada postre se le puede añadir distintos aderezos que tienen un costo de 0.50 ctvs. cada uno. Sumando este valor adicional, al anterior se obtiene el valor final de cada producto que venden.

Todos los postres se realizan con leche entera, sin embargo, el negocio ofrece la facilidad de cambiar el tipo de leche a descremada o deslactosada, según el cliente lo prefiera sin ningún costo adicional.

Luego de unos días, una persona que conoce sobre diseño de software ha revisado el sistema, y ha identificado que existen ciertos principios SOLID están siendo violados por el sistema desarrollado.

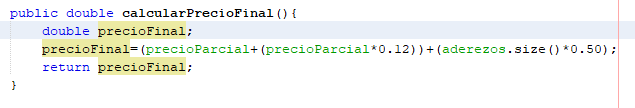
# Indicaciones

1. Identifique los principios SOLID que no están siendo respetados explicando el porqué.
2. Corrija el código para que solucionar los problemas previamente mencionados, creando un nuevo paquete donde si se respete los principios SOLID.
3. Por cada principio violado, muestre el código que lo incumple y luego muestre y explique como solucionó el problema con su código.

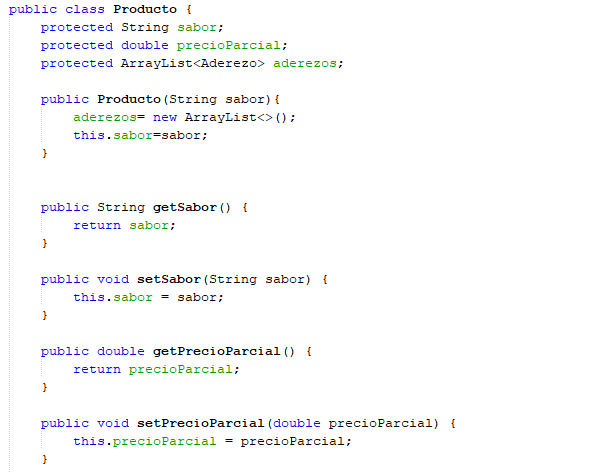
Puede crear clases o interfaces adicionales si considera necesario, pero no elimine el código para poder hacer la comparación final.

# Desarrollo

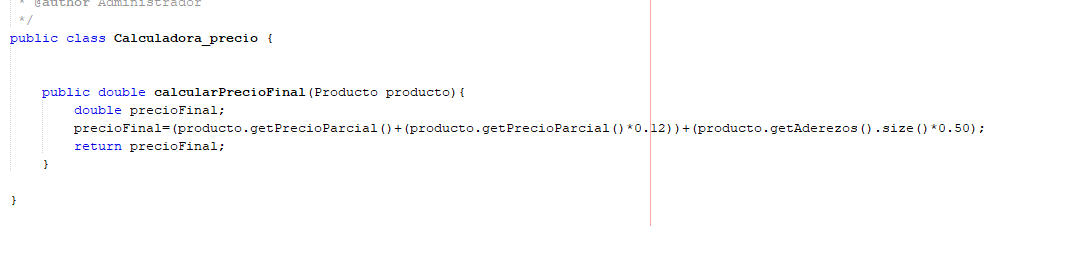
Se incumple el primer y el segundo principio en las clases helado y pastel. Va a existir conflicto cuando se necesite crear otro tipo de producto, además de que el método calcularPrecioFinal se repite en los postres



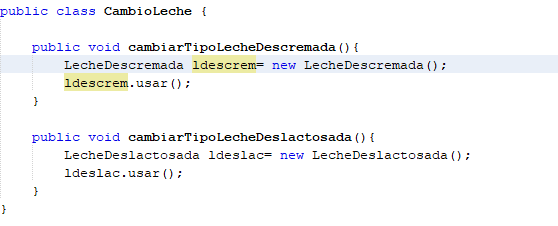
Las clases pastel y helado repiten el mismo método, por lo que se crea una clase padre que implemente aquel método evitando que se repita el código y facilitando la creación de un nuevo producto.



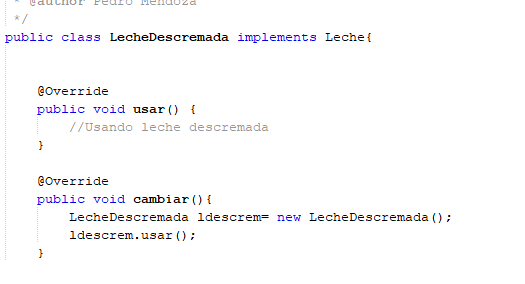
Y se implementa una clase para calcular los precios

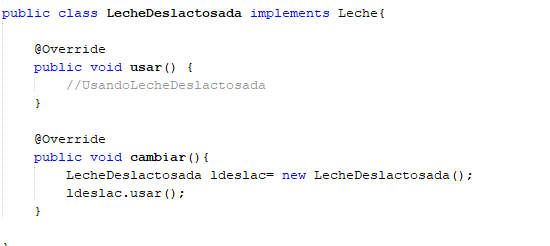


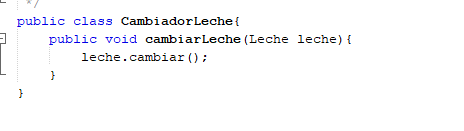
Se incumple el segundo principio en las clases de cambio de leche debido a que por cada tipo de leche se tiene que agregar otro método especifico al tipo de leche.



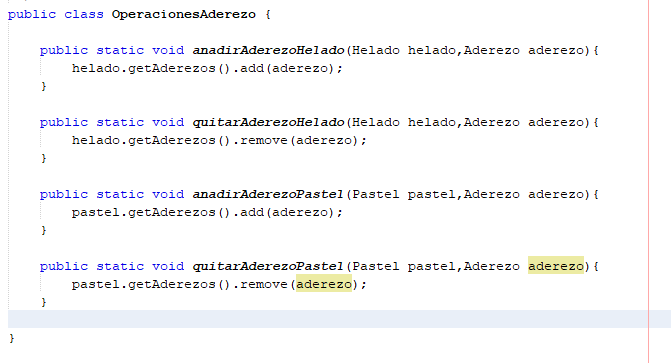
Para poder cumplir con este principio se va a crear una clase padre para leche y clases hijas para cada tipo de leche en donde se implemente el método usar. Adicionalmente la clase cambio leche va a implementar el método cambiarTipoLeche la cual recibirá como parámetro el tipo de leche.







Se incumple el segundo principio cuando se requiere añadir o remover un tipo de aderezo, debido a que se repite el proceso para cada uno de los tipos de postres.



Para poder cumplir con este principio se añadió una interfaz para obligar a cada uno de los postres a implementar su propio añadir y remover aderezo

