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

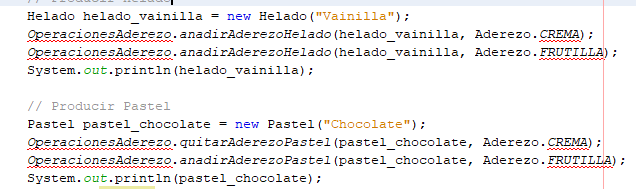
1. Identifique los principios SOLID que no están siendo respetados explicando el porqué.
2. Corrija el código para que solucionar los problemas previamente mencionados, creando un nuevo paquete donde si se respete los principios SOLID.
3. Por cada principio violado, muestre el código que lo incumple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

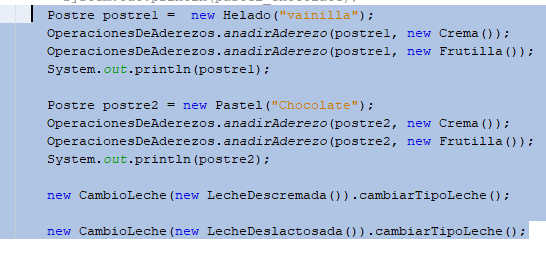
Puede crear clases o interfaces adicionales si considera necesario, pero no elimine el código para poder hacer la comparación final.

# Desarrollo

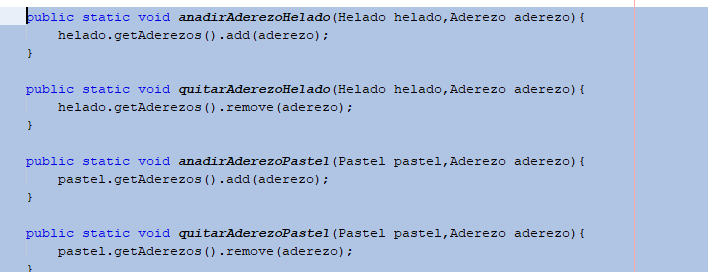
1. El primer principio que pude notar que estaba siendo violado fue el principio interface segregation, porque hay dos tipos de postres: helado y pastel, entonces hice una interface pastel con los métodos calcular precio final y obtener aderezos de manera que cuando se use en operaciones, solo se llameel método de la interface.

Antes

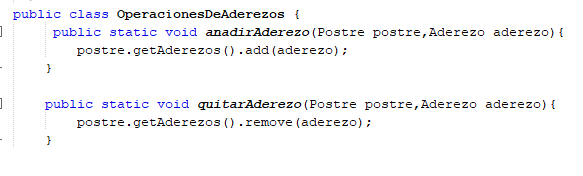


Después

1. Open-closed, esta violación la encontré en los enumerados. En lugar de tener enumerados hice una clase de cada tipo de aderezo, e hice una clase abstracta llamada aderezo, que me permite abstraerme de los diferentes tipos de aderezo, y cuando los pase por mi case operarAderezos, solo necesito enviarle una de las instancias de las clases hijas.

 Antes

Despues



1. Single responsability, esto lo encontré en la clase helado y pastel. Estas clases calculaban el valor del pastel en lugar de eso hice una nueva clase precio, que recibía el valor parcial del postre y calculaba su valor total.

Antes

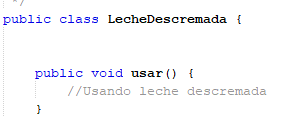
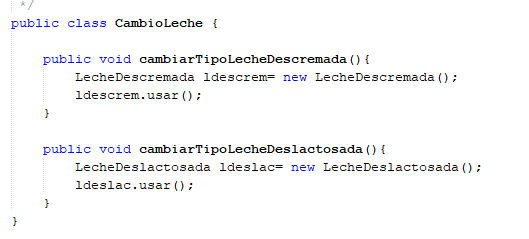


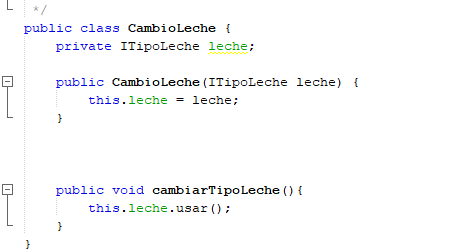
Despues



1. Liskovsubstitution, esta violación la encontré, con las clases de cambio de leche y para solucionarlo solo hice una interface llamada ITipoLeche que me permite, pasarle cualquier instancia de una clase que implemente esta interface a la clase cambiarLeche.

Antes



 Después

