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

**Paquete Leche y Procesos**

En esos paquetes se está violando el OCP ya que la leche se implementa una función usar que se repite en dos tipos de leches diferentes y si se agrega un tercer tipo de leche es necesario crear una nueva clase con una nueva forma de implementar usar leche. Además en el paquete procesos, el proceso cambiar leche posee una función cambiarLeche\_(tipo) para cada tipo diferente leche. La solución consiste en implementar una clase abstracta leche con un métodos abstracto usar(), que permite implementarlo de diferente forma para cada tipo de leche y además en la clase cambioLeche implementar una única función cambiarLeche que utilice un objeto Leche, creado en la clase cambioLeche.

**Paquete postres**

Se está violando el SRP puesto que en la clase Helado y Pastel hay responsabilidades que no pertenecen a las mismas, en este caso la de calcular el precio final de ambos. Adicionalmente se viola el OCP, ya que el método no está directamente relacionado con el producto, el cálculo del precio es el mismo.

**Paquete Otros y Procesos**

Se viola el OCP porque la clase OperacionesAderezo tiene código para cada tipo de postre, y si se agrega un nuevo postre habrá que modificar la clase. La solución es hacer que los métodos agregarAderezo y quitarAderezo reciban un Postre, no un Helado o un Pastel. Además, esto permite usar el DIP, porque la clase OperacionesAderezo ya no depende de implementaciones de los postres, sino de la clase Postre.