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.

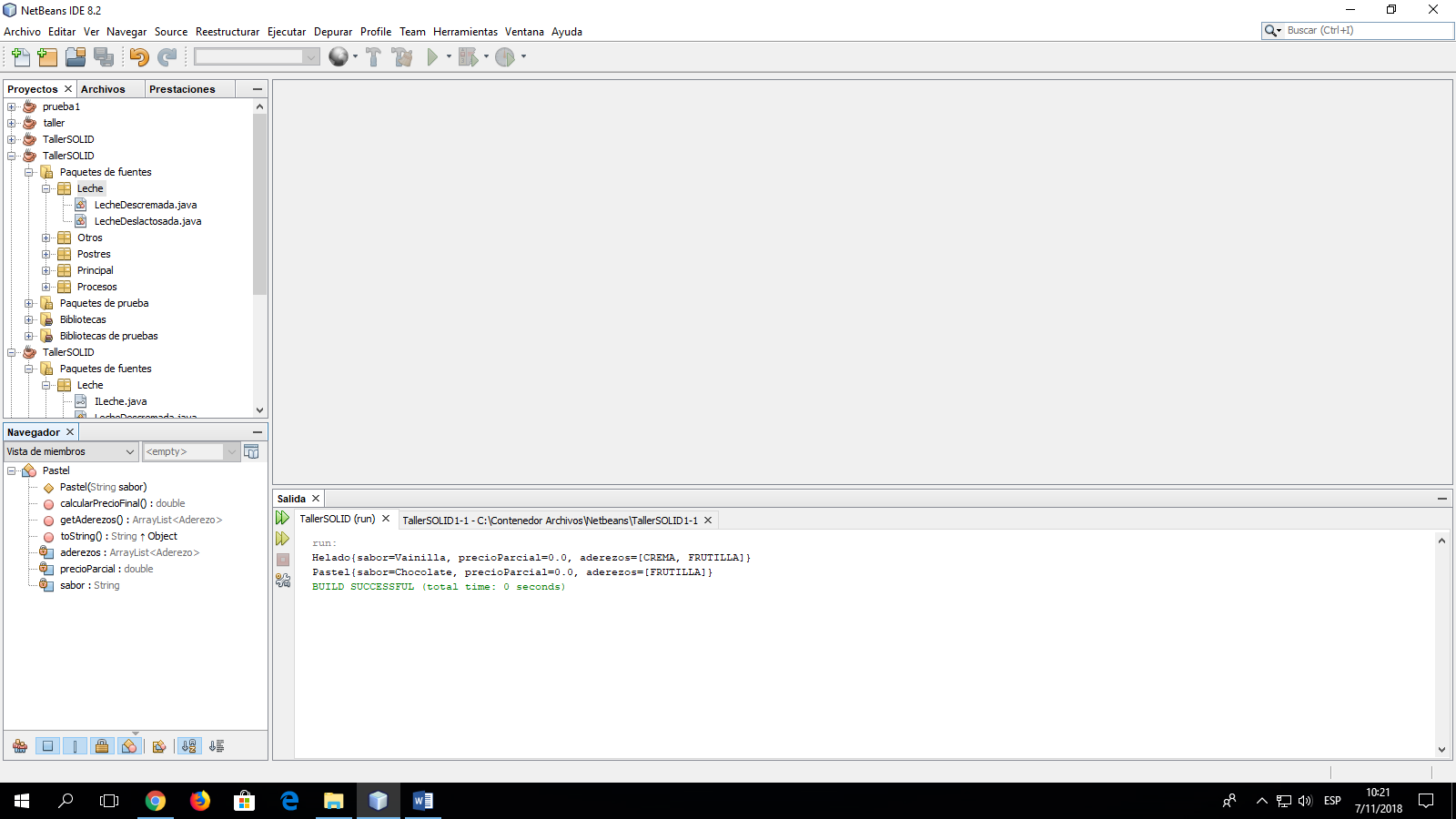
3. Por cada principio violado, muestre el código que lo incumple y luego muestre y explique como solucionó el problema con su código.

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

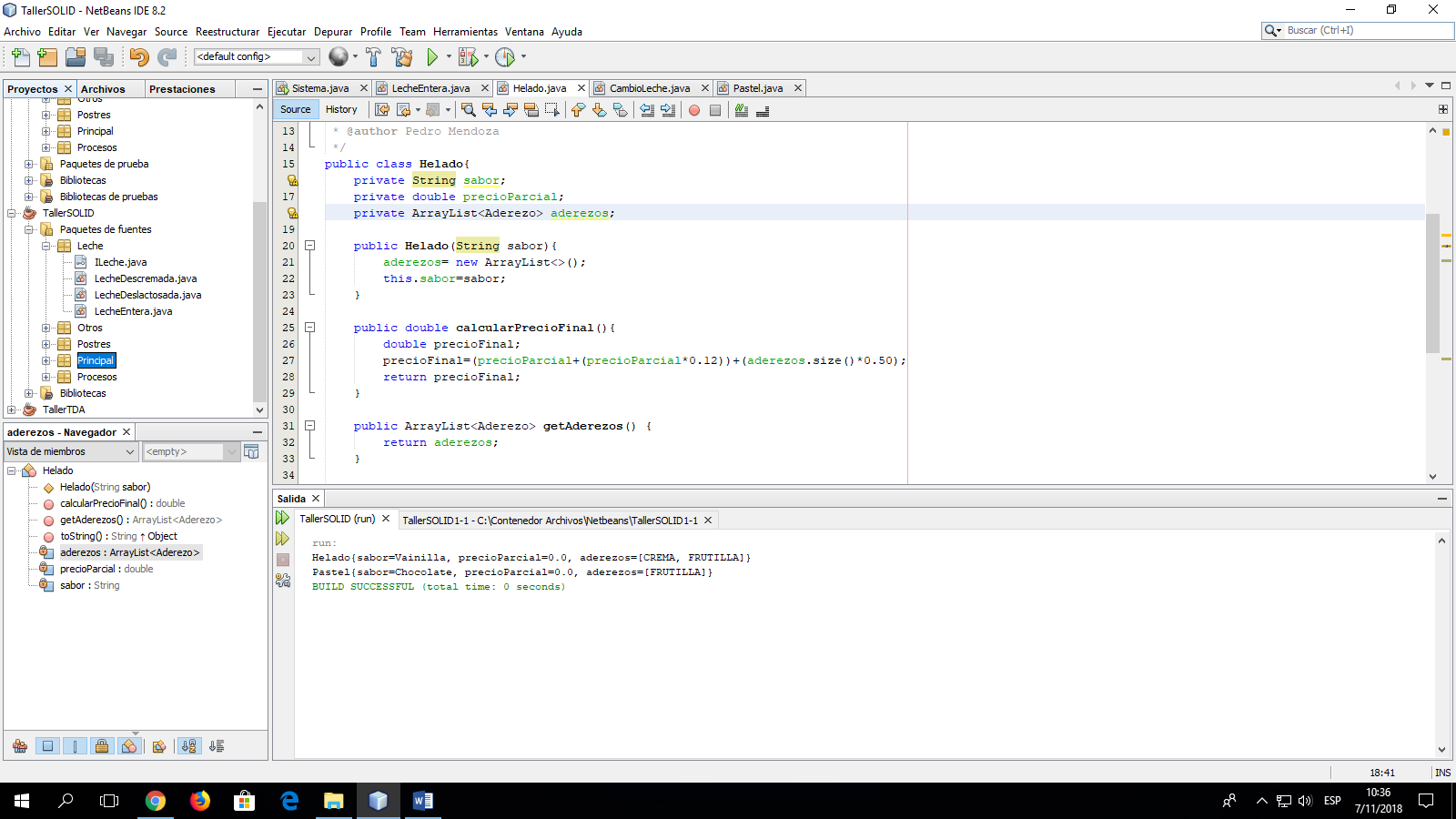
# **Desarrollo**

Aquí desarrolle lo anterior.

**Paquete de Leche**



Las clases LecheDescremada y LecheDeslactosada incumplen con el principio de LSP, pues ambas clases tienen métodos y atributos en común y podrían ser Derivaciones de una entidad base.

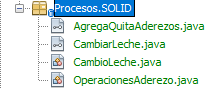


Al ya no existir una clase para Leche normal, se creó la clase leche entera que también es derivada de la interfaz ILeche.

**Paquete Procesos**

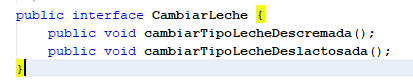
1 .- En este paquete existen 2 clases que no estan usando el principio de inversion de dependencia ya que puede darse el caso en que los metodos incluidos en estas clases cambien tanto de los parametros que reciben como su funcionalidad por lo que se recomendaria usar para ello interfaces.

2.-

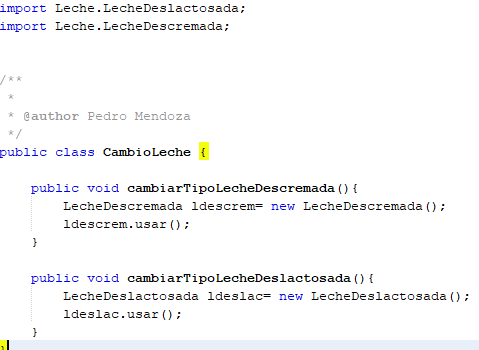


3.-

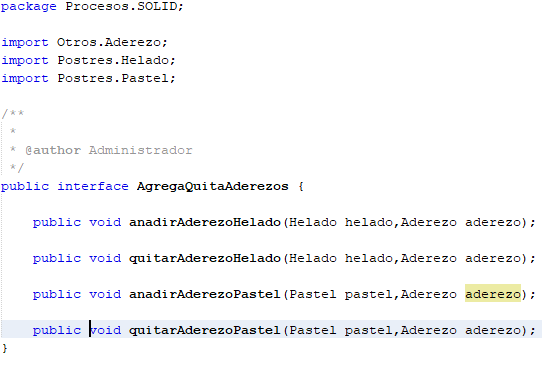
**Correcto**



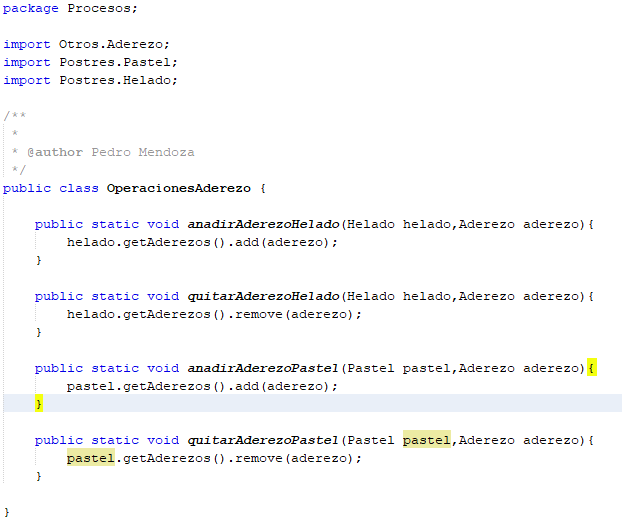
**Incorrecto**

****

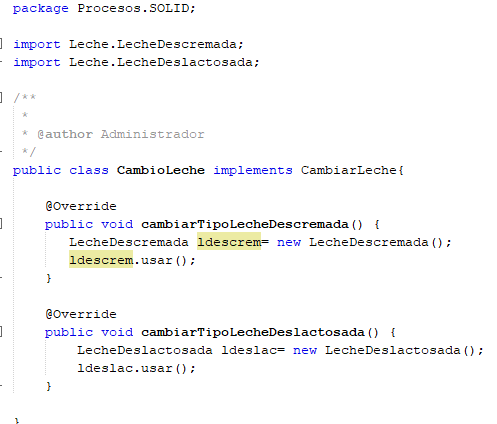
**Correcto**

****

**Incorrecto**

****

**Correcto**

****

****

**Paquete Postres**

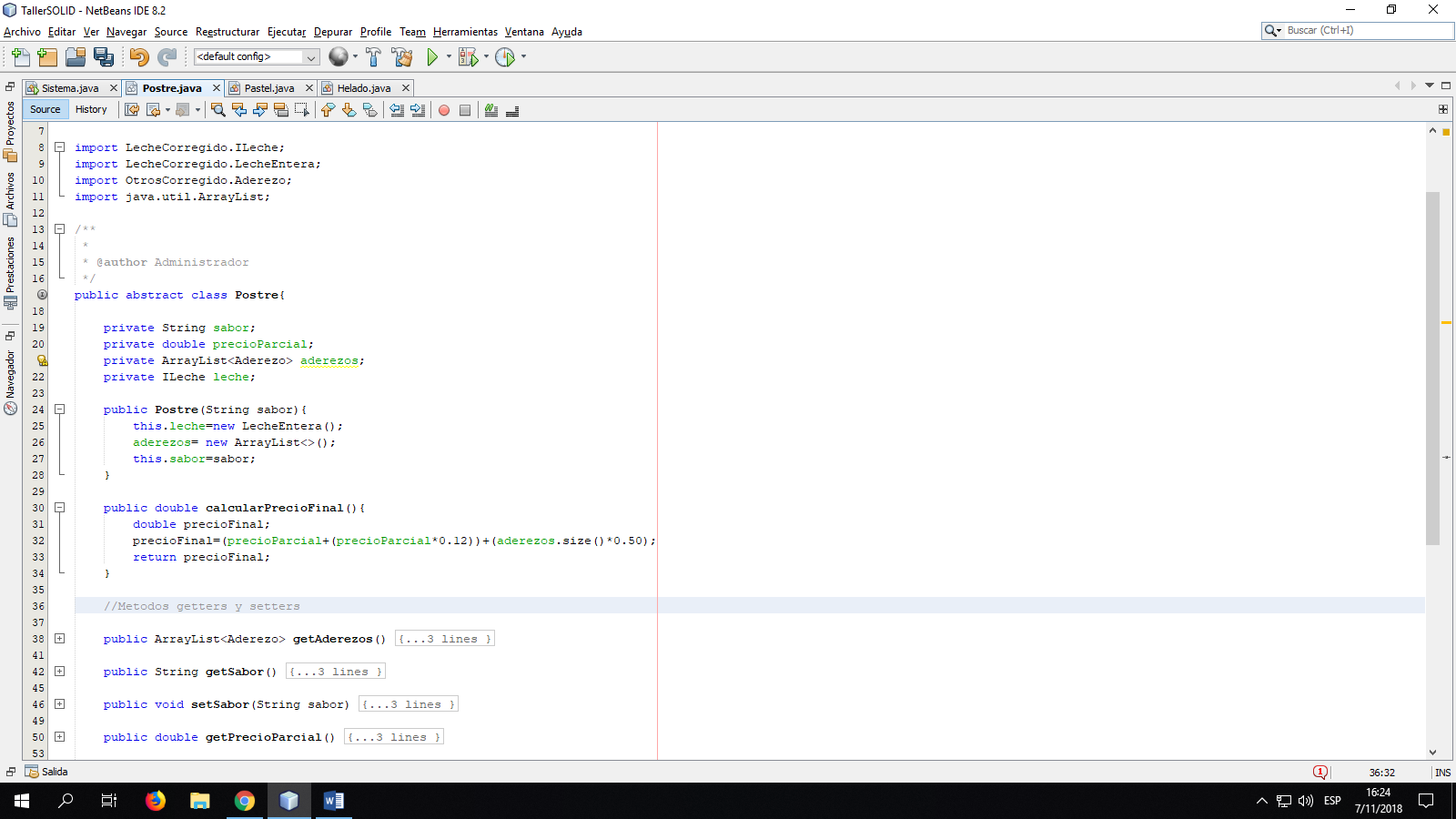
Las clases helado y pastel, incumplen el principio OCP (Open/Closed principle) ya que al tener los mismos atributos y poseen métodos en común, estas deben hacer uso de la herencia y de esa forma, expandir el comportamiento de estas sin modificar el código.

Así cuando implementemos un nuevo Postre solo hacemos que implementen este comportamiento y evitamos modificar el código existente.

**Incorrecto**

**Correcto**

Se crea la clase abstracta que defina el comportamiento general de un Postre, como sabor, aderezos y precio además del método calculaPrecioFinal.



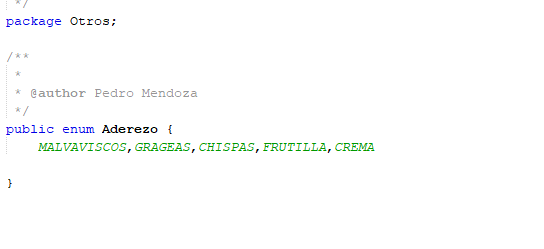
Las clases Pastel y Helado, implementan la clase Postre haciendo que hereden sus métodos y atributos generales, entonces se definen solo los métodos y comportamientos únicos de la clase, como el nombre.

Lo cual nos facilita mucho al momento de crear un nuevo postre.

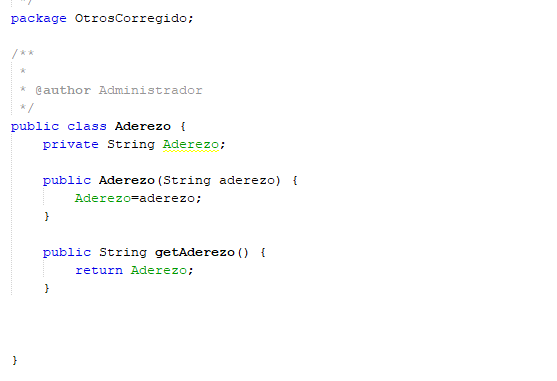
**Paquete Otros**

la clase Aderezo incumple con el principio de OCP, por lo que es mejor instanciar la clase y tener el tipo de aderezo como atributo.

**Incorrecto**

****

**Correcto**

****