ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION

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

**Integrantes:**

Danny De La A

Diego Muñoz

Eduardo Tigse

Henry Maticurena

Karen Bermudez

# 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

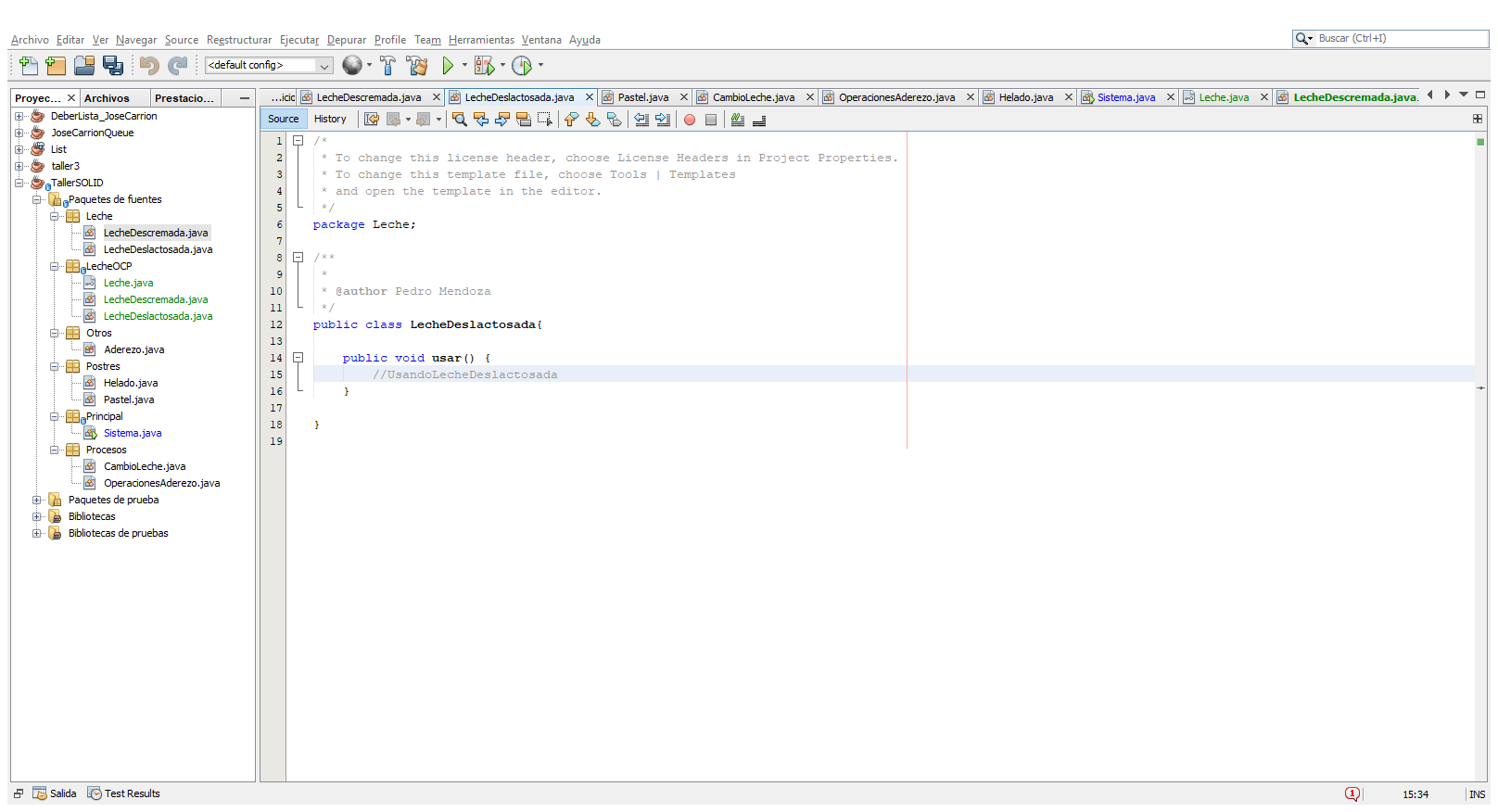
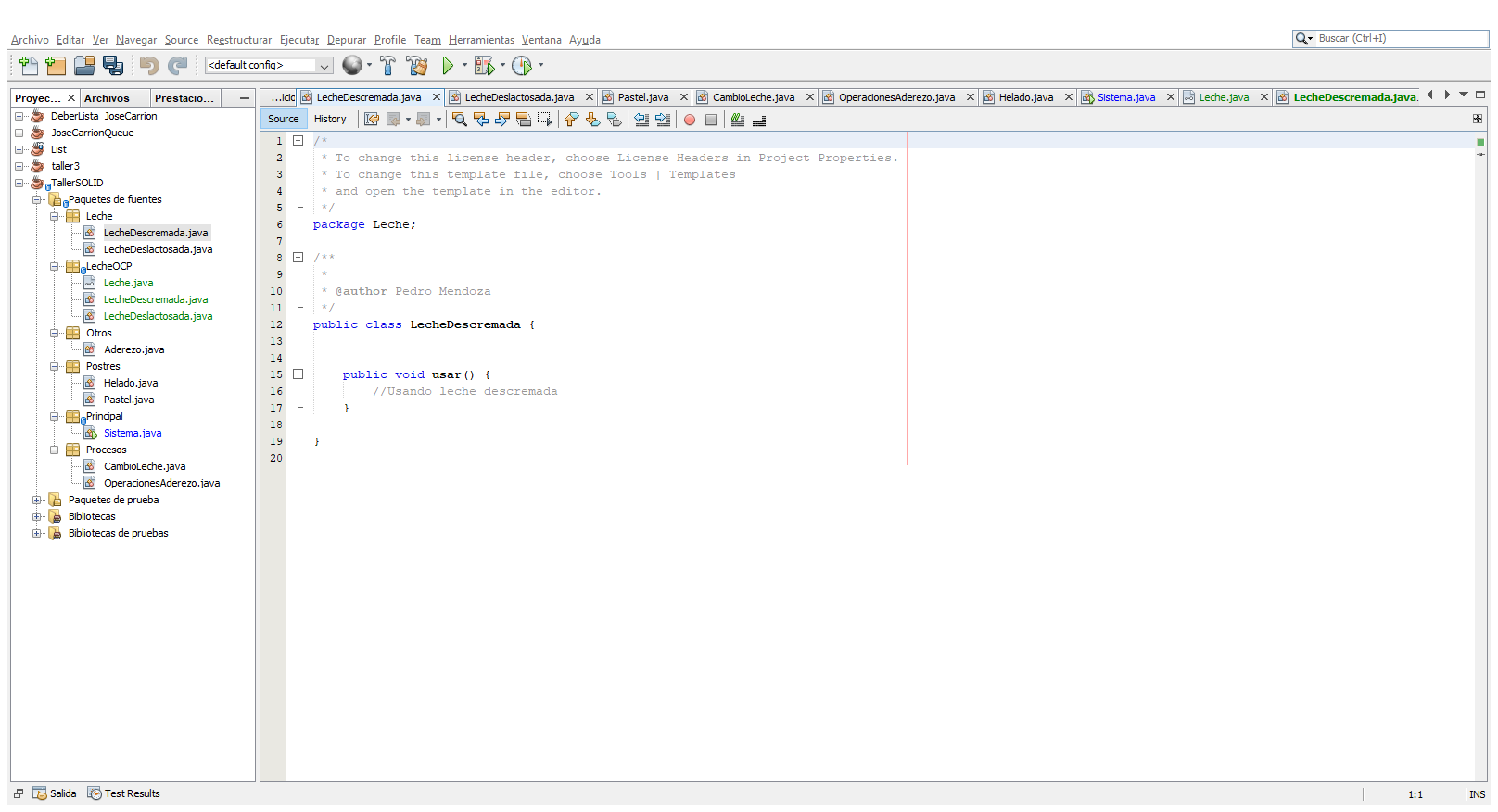
1. Identifique los principios SOLID que no están siendo respetados explicando el porqué.
2. Corrija el código para que solucionar los problemas previamente mencionados, creando un nuevo paquete donde si se respete los principios SOLID.
3. Por cada principio violado, muestre el código que lo incumple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

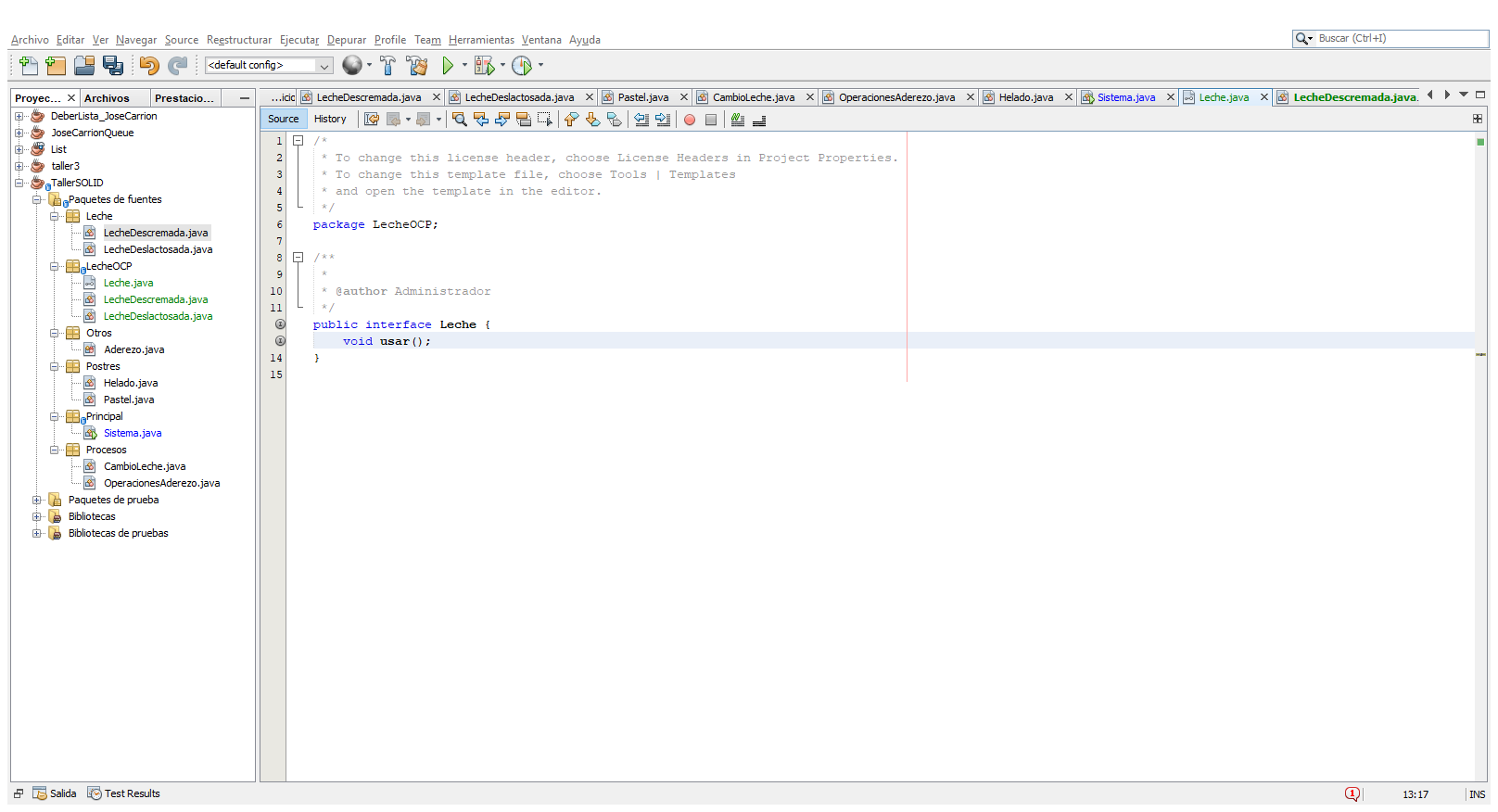
# Desarrollo

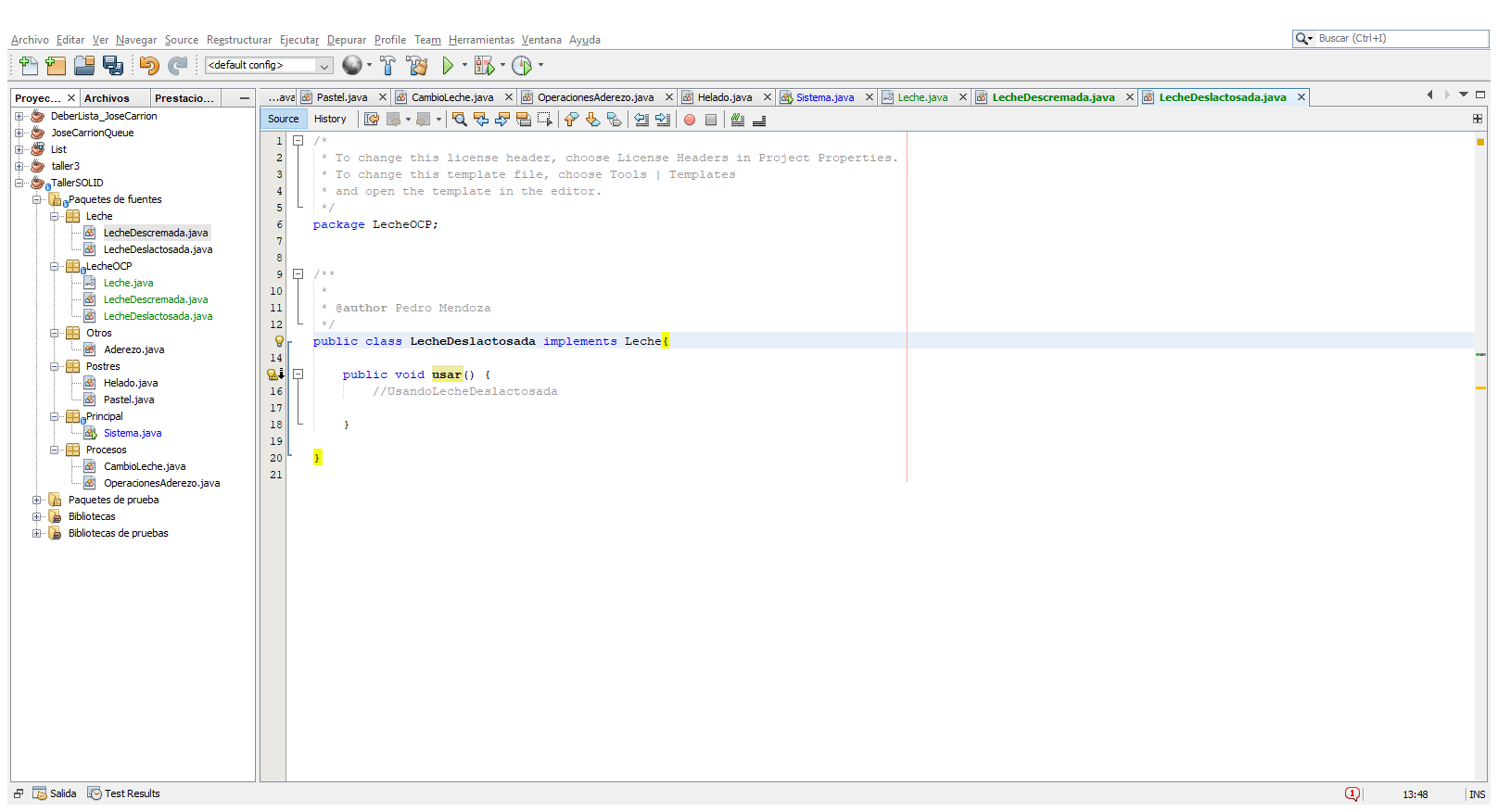
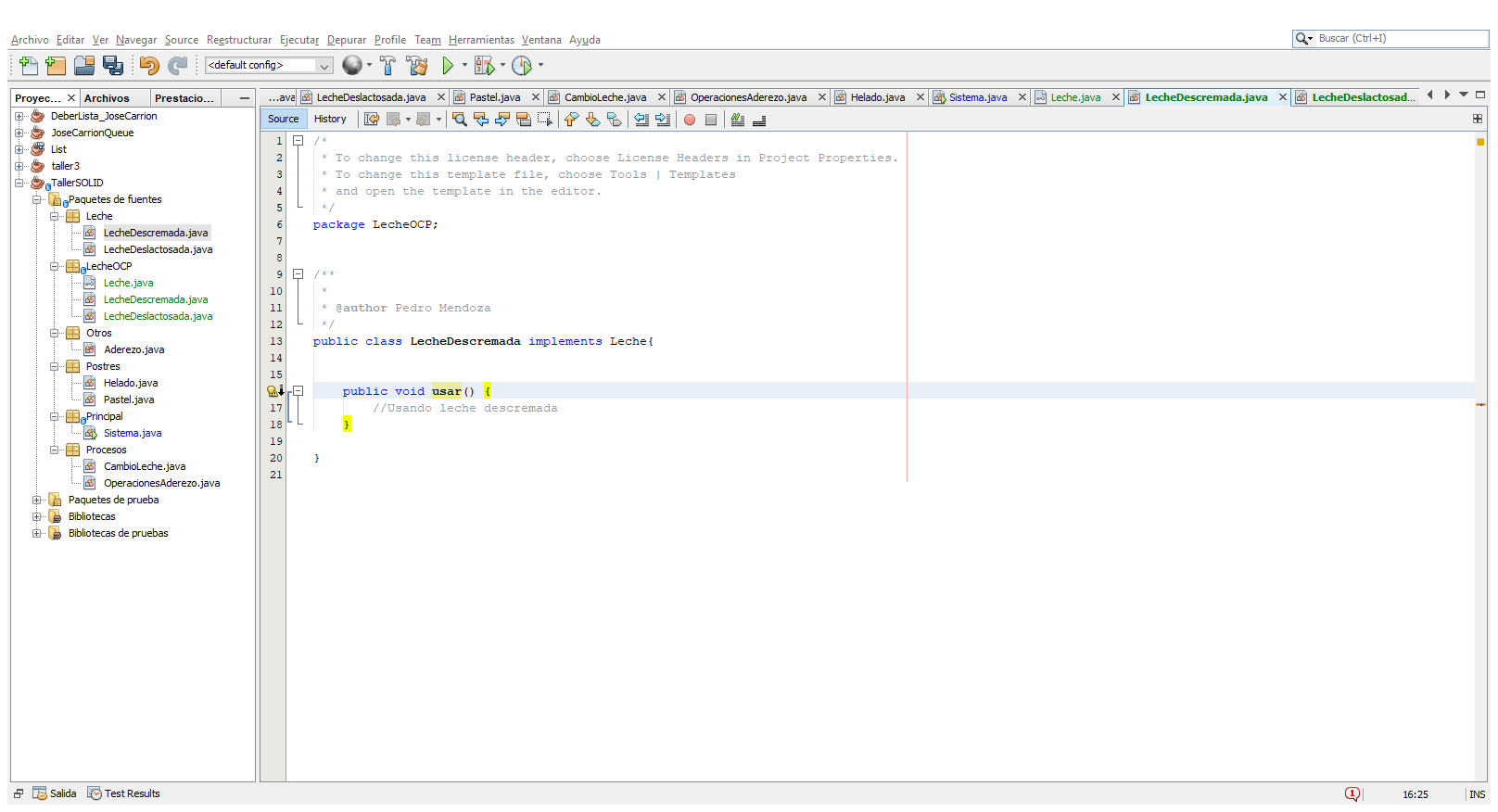
Aquí desarrolle lo anterior.

Código violado: OPEN-CLOSED PRINCIPLE



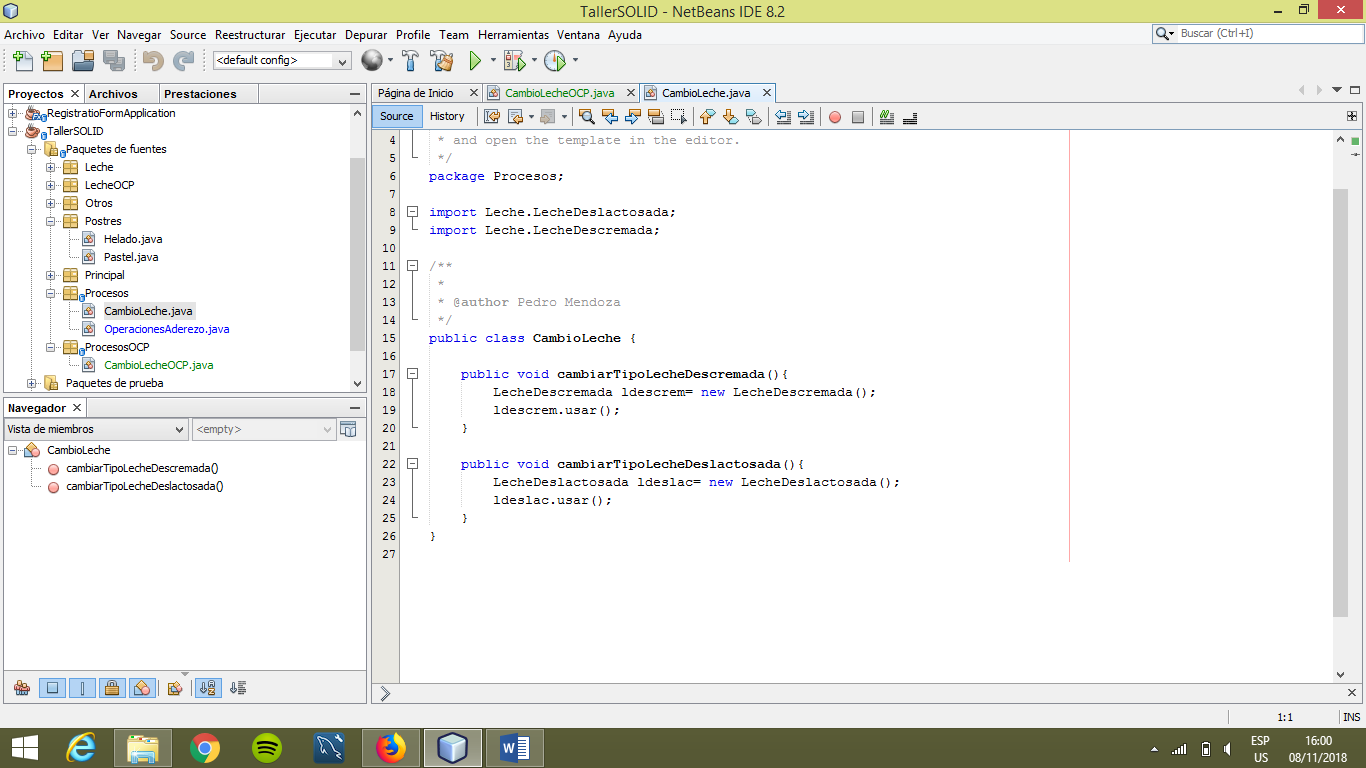
Código corregido



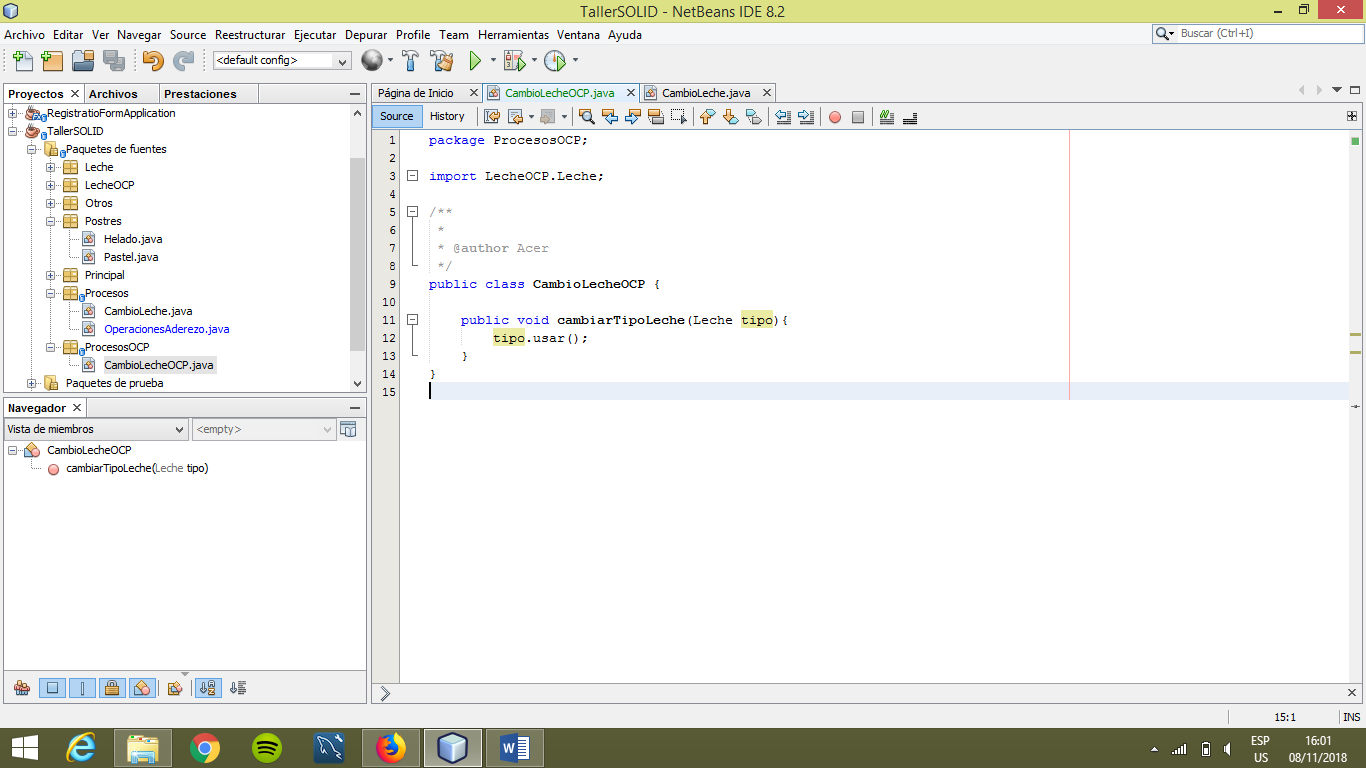


Para corregir este principio lo que se debe hacer es crear una interface que tendrá el método usar, y que las otras dos clases LecheDescremada y LecheLactosa la implementen.

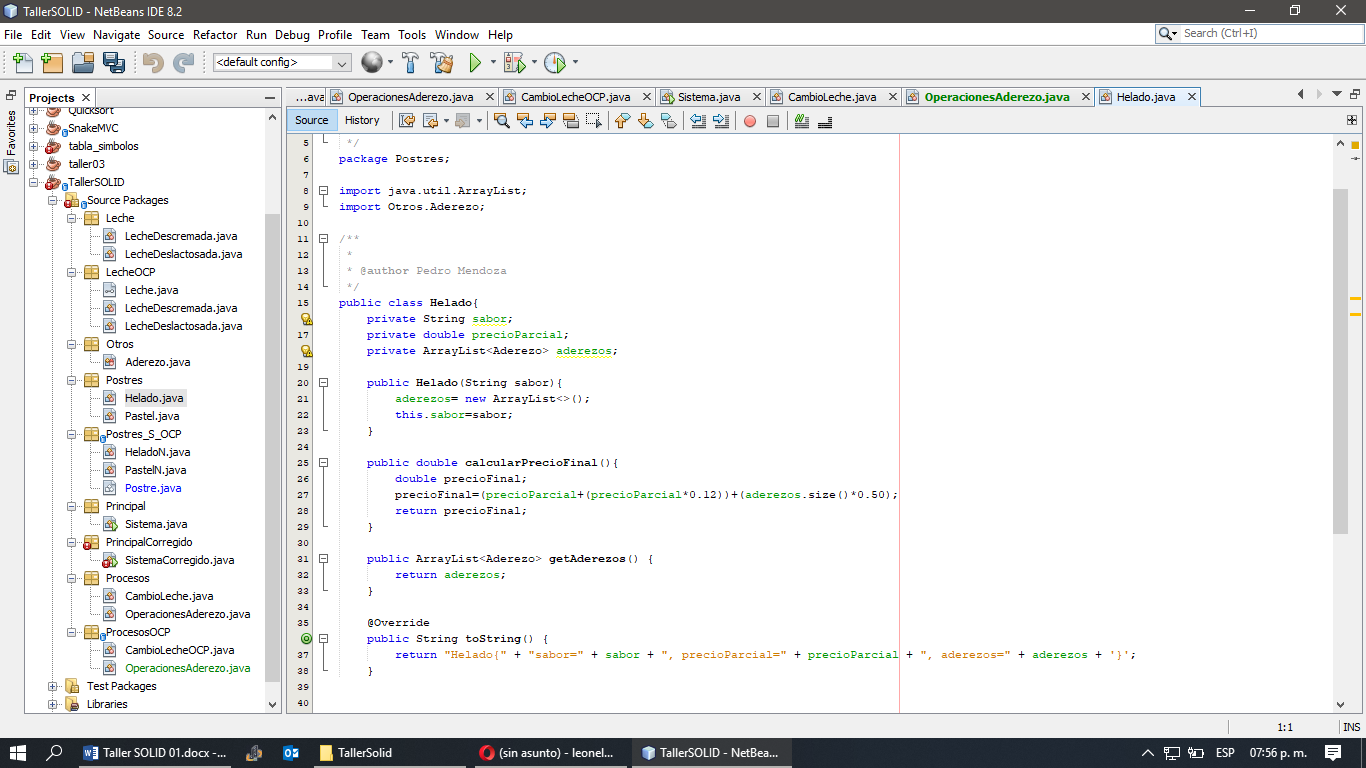
Código violado: OPEN-CLOSED PRINCIPLE

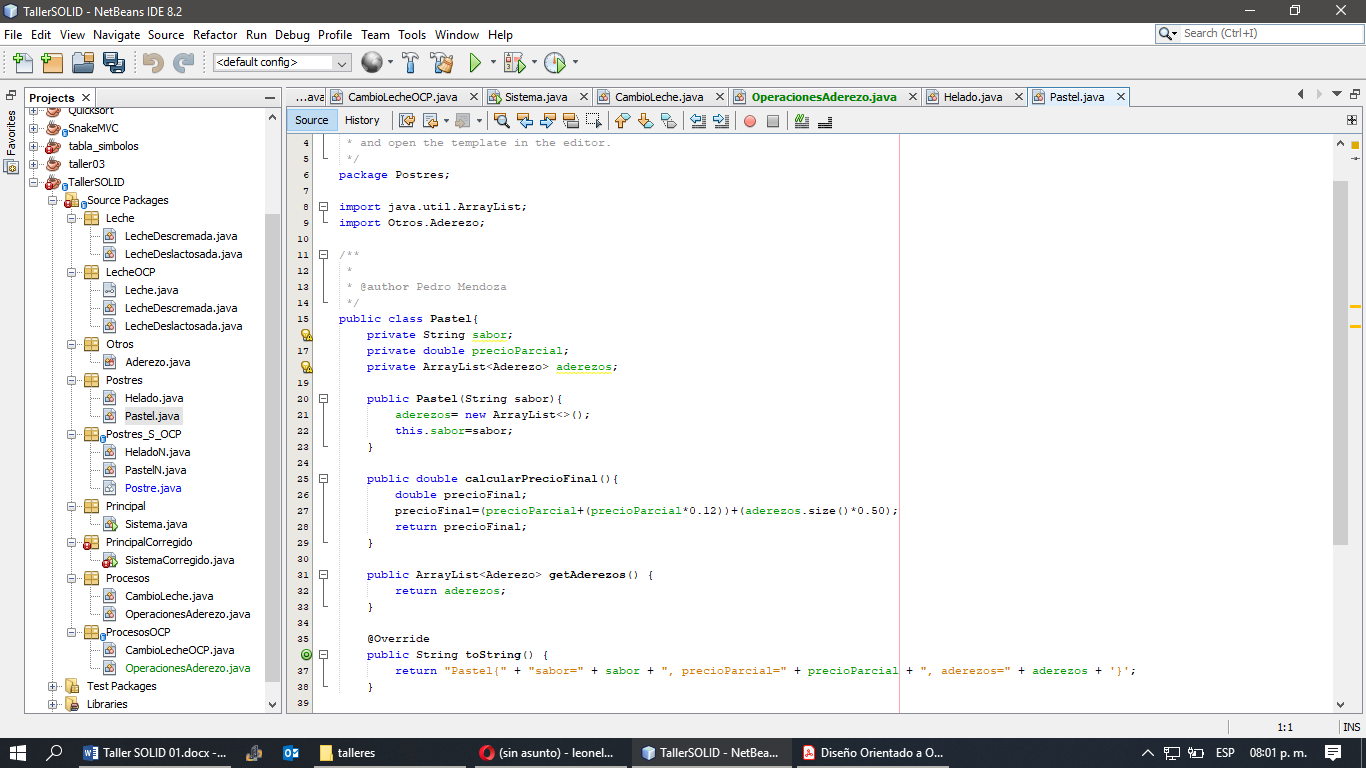


Código corregido:

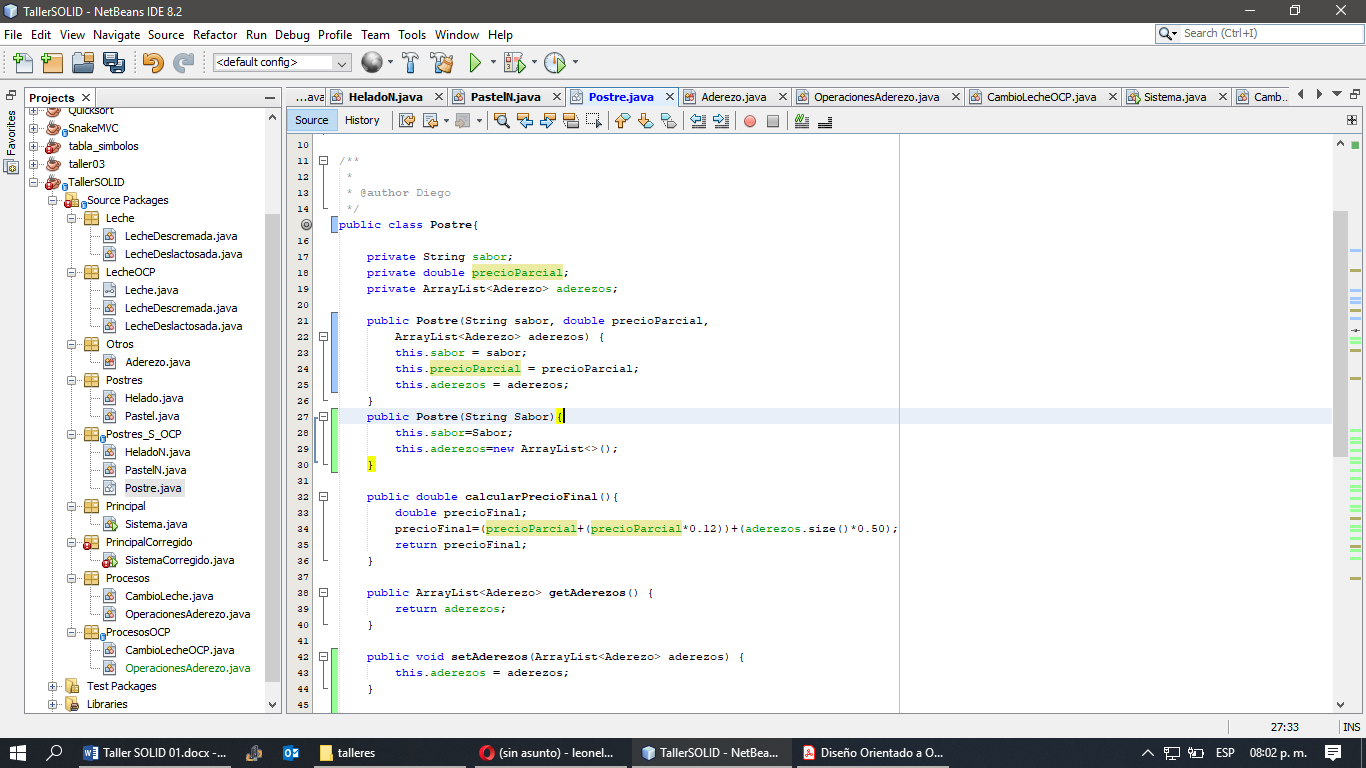


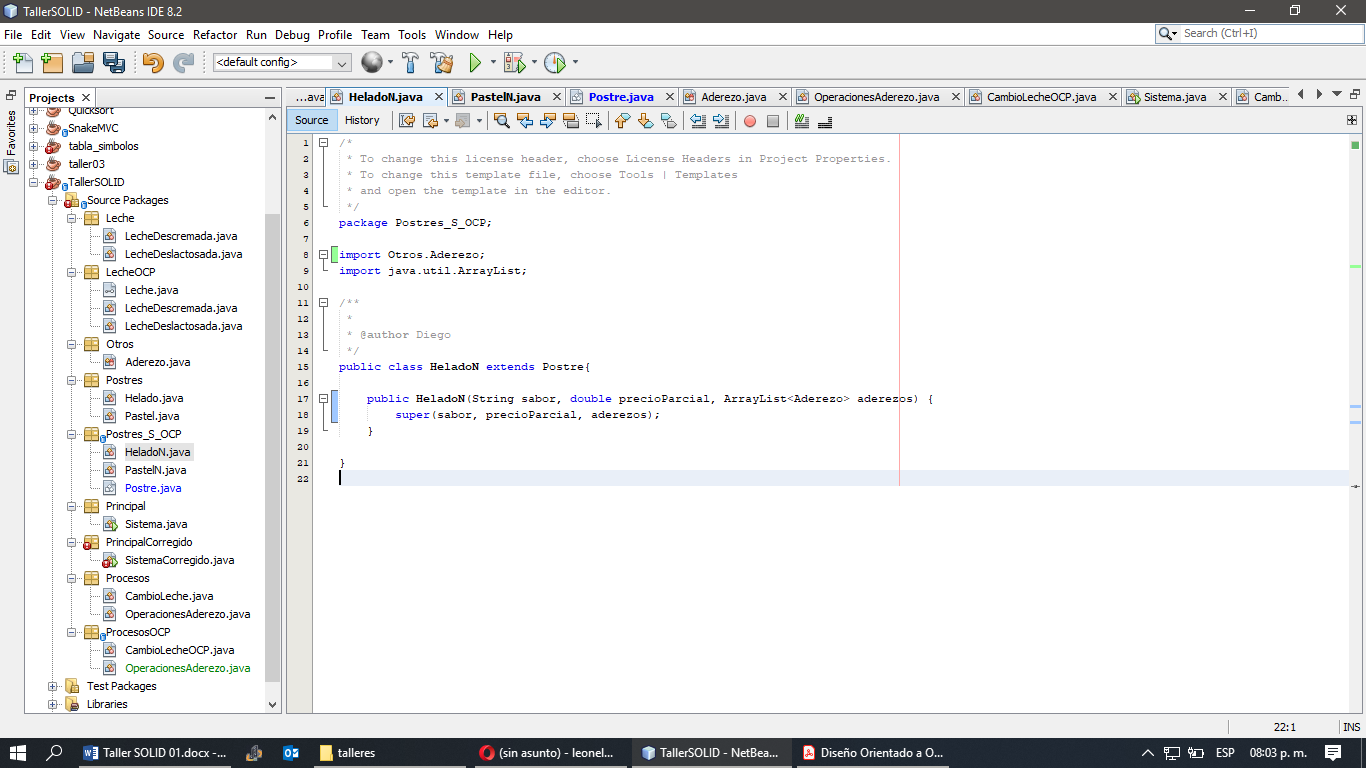
Para corregir este principio lo que se debe hacer es tomar como argumento la interfaz de la corrección anterior y trabajar con eso, como ambas clases de Leche utilizan el método .usar() no se necesita crear dos métodos por separado.

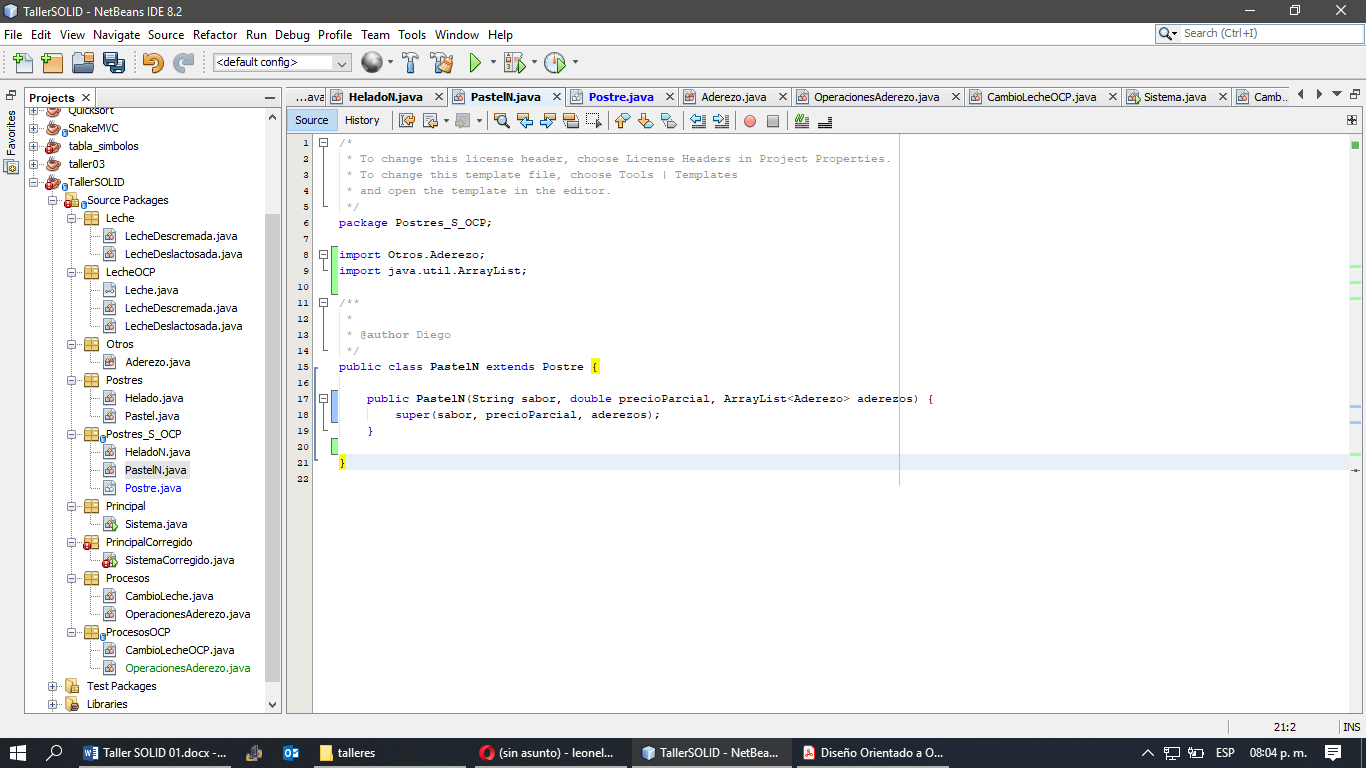
Codigo violado: SRP y OCP



Código corregido:

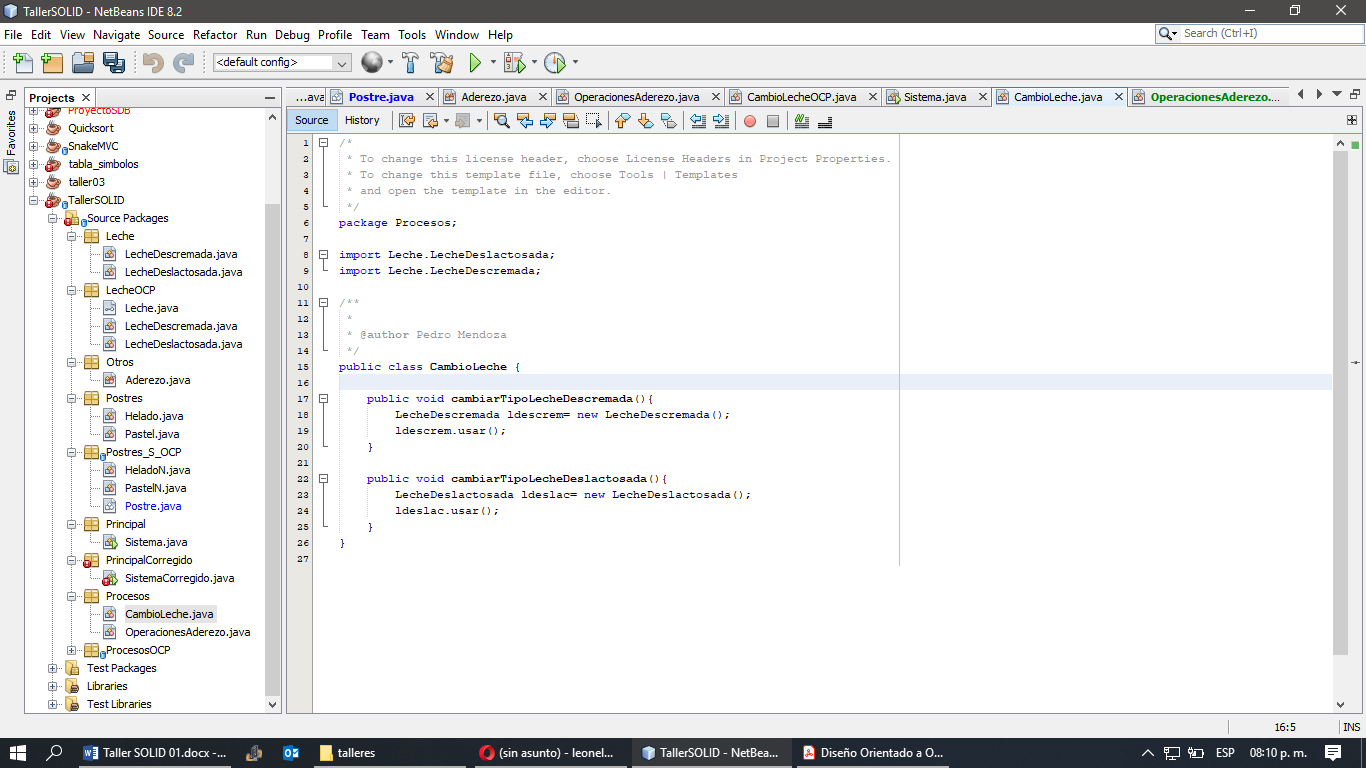


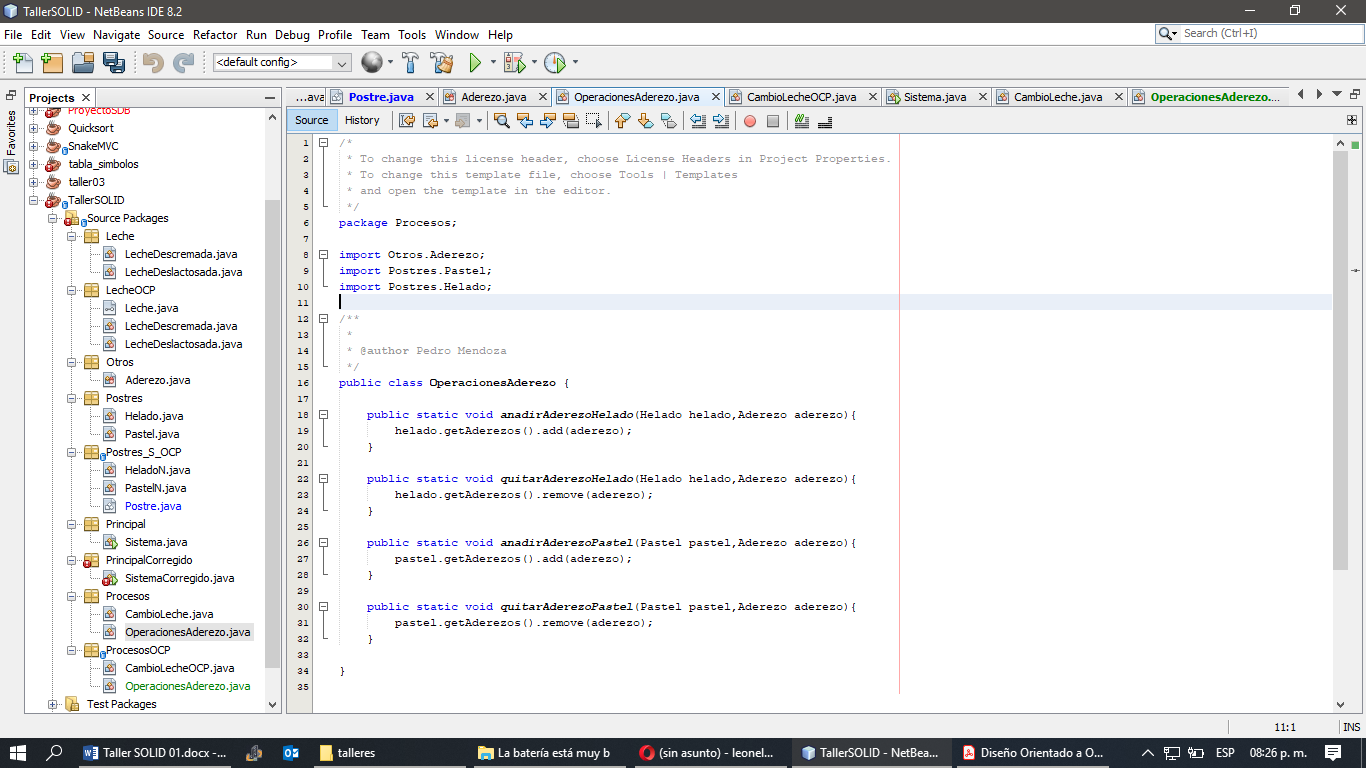




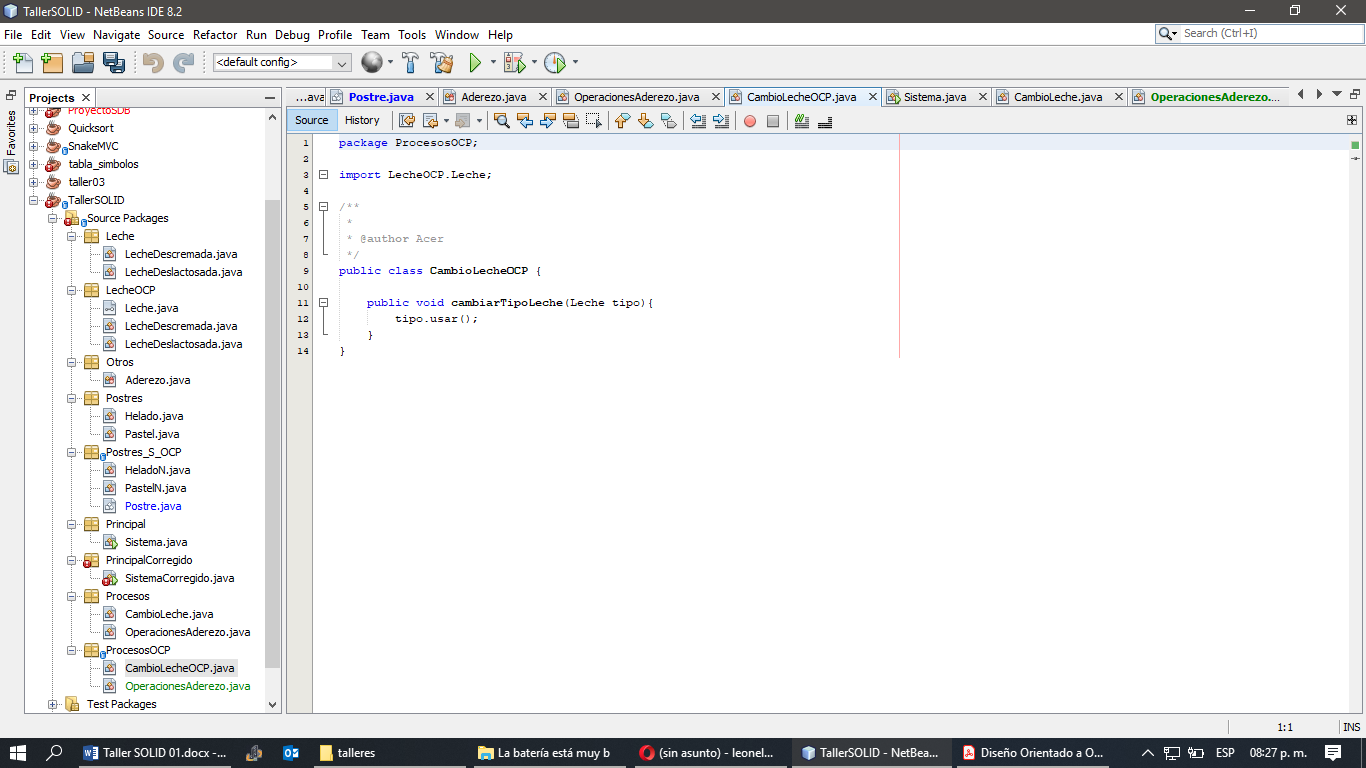
Se está obteniendo una violación a SRP y OCP, debido a que las clases pastel y helado no tenía una función específica que cumplir por lo que ambas tenían código repetidos, esto conllevo a crear una clase padre llamada Postre que implemente el método y lo realice, debió a que es el mismo método para ambas, a la vez integra todos los atributos que tenían tanto la clase helado y pastel, ahora pastelN y HeladoN extienden de Postre.

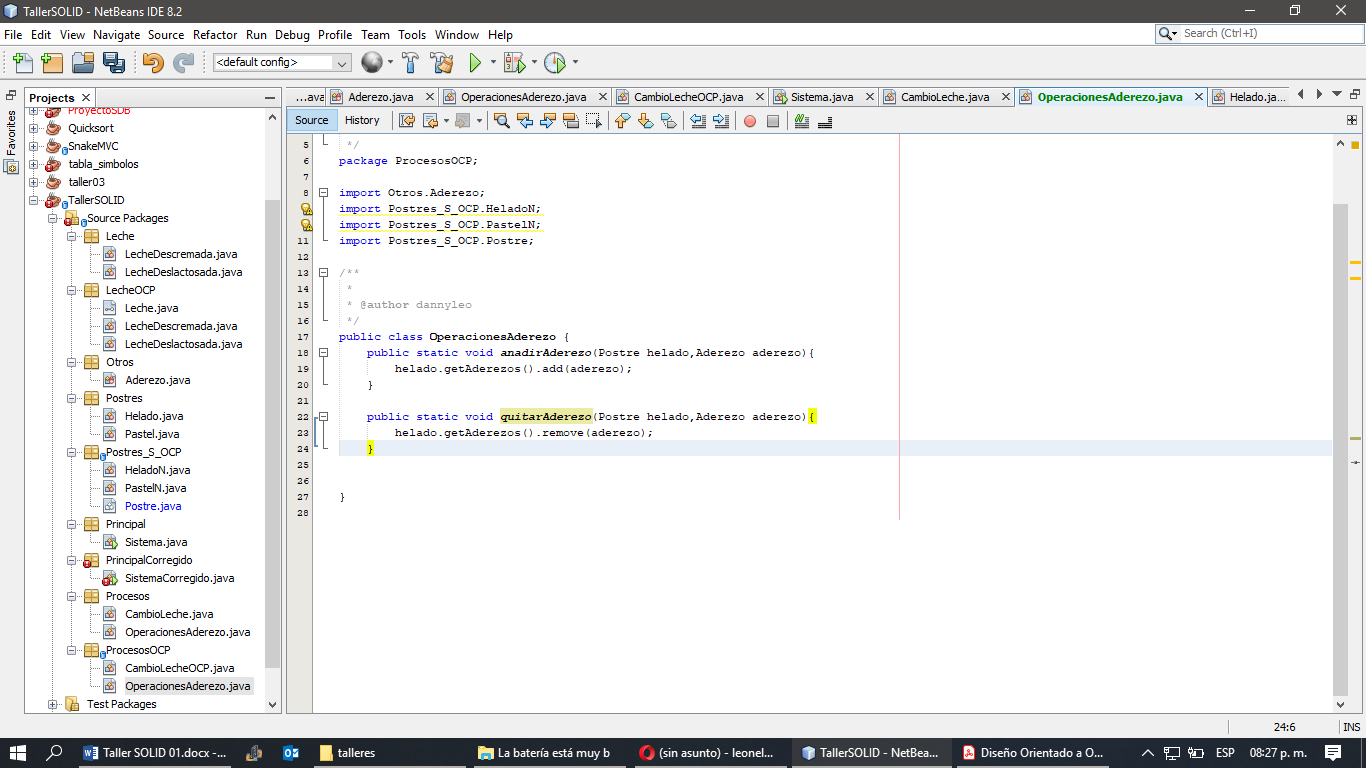
Codigo violado: SRP y DIP





Codigo Corregido:





Se incumple los principios de SRP, debido a que las clases cambioLehe y OperacionesAderezo, realizan diversos métodos y los cuales se pueden simplificar gracias a la interfaz Leche, la cual tiene el método usar y DIP debido a que la abstracción que existe en el momento de usar el método en Operaciones Aderezo de quitar o añadir Aderezo, solo se necesito saber que es un postre para quitar y añadir el aderezo.