UNIVERSIDAD DE LIMA

FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS

Ingeniería de Software II

Laboratorio N°3

Patrones Creacionales

Repositorio: https://github.com/anderalarcon/ing-sw-2-2024-1-java

Adapter

Permite que dos interfaces incompatibles trabajen juntas. Convierte la interfaz de una clase en otra interfaz que un cliente espera encontrar

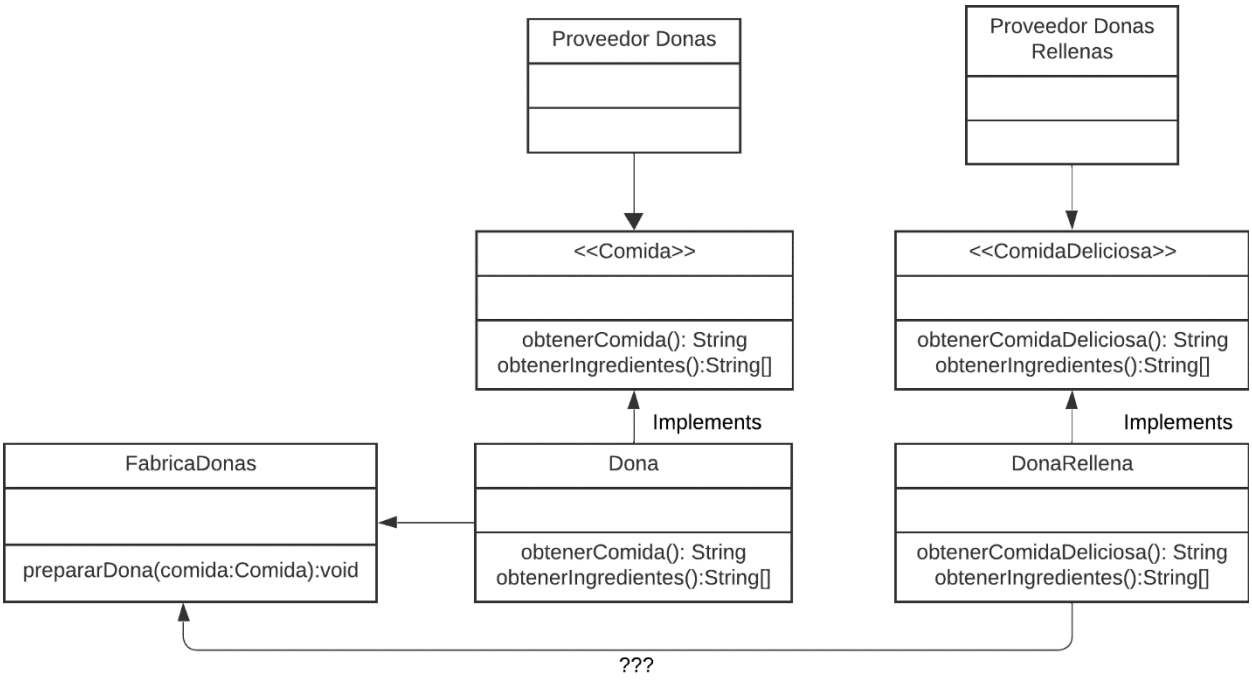
Contexto: Tu sistema ha estado trabajando con un proveedor de donas normales durante un tiempo considerable. Sin embargo, ahora tu empresa ha decidido expandirse y trabajar con un nuevo proveedor que ofrece donas rellenas. Desafortunadamente, la interfaz del nuevo proveedor no es compatible con la que tu sistema espera.

Objetivo: Implementar el patrón Adapter para integrar el nuevo proveedor de donas rellenas en tu sistema existente, asegurando que pueda funcionar sin problemas junto con el proveedor de donas normales.

Descripción del ejercicio:

1. Define una interfaz **Comida** que tenga métodos para obtener el nombre de la comida y los ingredientes.
2. Implementa la clase **Dona**, que representa donas normales, e implementa la interfaz **Comida**.

3. Define una nueva interfaz **ComidaDeliciosa** que tenga métodos para obtener el nombre de la comida y los ingredientes, específicamente para las donas rellenas.
4. Implementa la clase **DonaRellena**, que representa donas rellenas, e implementa la interfaz **ComidaDeliciosa**.
5. Crea una clase **Adaptador** que implemente la interfaz **Comida** pero que utilice una instancia de **ComidaDeliciosa** internamente. Este adaptador servirá para integrar el nuevo proveedor de donas rellenas en tu sistema existente.
6. En la clase **Main**, crea instancias de **Dona** y **DonaRellena**, y luego utiliza el adaptador para preparar las donas rellenas y que puedan ser manejadas por tu sistema existente.



Facade

Permite proporcionar una interfaz unificada y simplificada para un conjunto de interfaces o subsistemas más complejos

Contexto:
Estás trabajando en el desarrollo de un sistema de comercio electrónico que necesita interactuar con múltiples subsistemas, como gestión de usuarios, gestión de productos y gestión de pedidos. Para mantener el código organizado y simplificar la interacción con estos subsistemas, decides implementar el patrón Facade.

Objetivo:
El objetivo de este ejercicio es implementar un sistema de comercio electrónico utilizando el patrón Facade en Java. Al finalizar el ejercicio, deberías tener una clase Facade que simplifique la interacción con los subsistemas y permita completar pedidos de manera eficiente.

Descripción del ejercicio:

1. **Definición de subsistemas:**
    - Crea las clases necesarias para gestionar usuarios, productos y pedidos. Cada clase debe tener métodos para realizar operaciones específicas, como crear un usuario, crear un producto y crear un pedido.
2. **Implementación de la clase Facade:**
    - Crea una clase Facade llamada **FachadaTiendaOnline** que encapsule la complejidad de los subsistemas. Esta clase debe tener referencias a los subsistemas de gestión de usuarios, productos y pedidos.
    - Implementa un método en la clase Facade llamado **completarOrden** que reciba los datos necesarios para realizar una compra (por ejemplo, nombre de usuario, dirección de correo electrónico, nombre del producto y precio) y que utilice los subsistemas para completar la orden.

3. **Programa principal:**
    - Implementa una clase **Main** que contenga un método **main**.
    - En el método **main**, instancia un objeto de la clase **FachadaTiendaOnline** y llama al método **completarOrden** para realizar una compra de ejemplo.