Practica 6. Factory y Observer

Castillo Dominguez Sonia Vanessa

date: "19/09/25"

Problema

Sistema de Notificación de Pedidos en una Cafetería Una cafetería moderna busca digitalizar la gestión de sus pedidos. Actualmente, cuando un cliente solicita una bebida, el barista debe anotar el pedido en papel y avisar manualmente al cajero y a los demás empleados. Este proceso genera retrasos, confusiones y pérdida de información. Un cliente hace un pedido en una cafeteria . Los empleados (baristas y cajero) son observadores que reciben notificaciones cuando llega un nuevo pedido.

codigo

```
from abc import ABC, abstractmethod
# Clase abstracta Producto (bebida)
class Bebida(ABC):
    @abstractmethod
    def preparar(self):
        pass


# Clases concretas (tipos de bebidas)
class Cafe(Bebida):
    def preparar(self):
        return " Preparando un Café"


class Te(Bebida):
    def preparar(self):
```

```
        return " Preparando un Té"


class Chocolate(Bebida):
    def preparar(self):
        return " Preparando un Chocolate"


# Factory que crea bebidas según el tipo
class BebidaFactory:
    @staticmethod
    def crear_bebida(tipo):
        if tipo == "cafe":
            return Cafe()
        elif tipo == "te":
            return Te()
        elif tipo == "chocolate":
            return Chocolate()
        else:
            raise ValueError(" Tipo de bebida no disponible")



# OBSERVER
# Interfaz Observer
class Observador(ABC):
    @abstractmethod
    def actualizar(self, mensaje):
        pass


# Sujeto (cafetería) -> mantiene lista de observadores
class Cafeteria:
    def __init__(self):
        self.observadores = []

    def suscribir(self, obs):
        self.observadores.append(obs)

    def cancelar_suscripcion(self, obs):
        self.observadores.remove(obs)
```

```
    def notificar(self, mensaje):
        for obs in self.observadores:
            obs.actualizar(mensaje)


# Observadores concretos (empleados)
class Barista(Observador):
    def __init__(self, nombre):
        self.nombre = nombre

    def actualizar(self, mensaje):
        print(f" Barista {self.nombre} recibió notificación: {mensaje}")


class Cajero(Observador):
    def __init__(self, nombre):
        self.nombre = nombre

    def actualizar(self, mensaje):
        print(f" Cajero {self.nombre} recibió notificación: {mensaje}")



# SIMULACIÓN
if __name__ == "__main__":
    # Crear el sujeto (cafetería)
    cafeteria = Cafeteria()

    # Crear observadores
    barista1 = Barista("Luis")
    barista2 = Barista("Ana")
    cajero = Cajero("María")

    # Suscribir observadores
    cafeteria.suscribir(barista1)
    cafeteria.suscribir(barista2)
    cafeteria.suscribir(cajero)

    # Crear pedidos usando Factory
    pedido1 = BebidaFactory.crear_bebida("cafe")
    pedido2 = BebidaFactory.crear_bebida("te")
```

```
    # Notificar a los empleados cuando llega un pedido
    cafeteria.notificar(pedido1.preparar())
    cafeteria.notificar(pedido2.preparar())
```

```
Barista Luis recibió notificación:  Preparando un Café
Barista Ana recibió notificación:  Preparando un Café
Cajero María recibió notificación:  Preparando un Café
Barista Luis recibió notificación:  Preparando un Té
Barista Ana recibió notificación:  Preparando un Té
Cajero María recibió notificación:  Preparando un Té
```