Patrón de Diseño

Decorador

Valeria Ospina
Carlos Suarez

Funcion

El patrón Decorator (Decorador) te permite añadir funcionalidades a objetos de forma dinámica sin modificar su estructura original. Es como "decorar" un objeto con nuevas características.

Notificaciones

untitled

File Edit Format Run Options Window Help

```python
# Componente base (interfaz)
class Notificador:
    def enviar(self, mensaje):
        pass

# Componente concreto
class NotificadorBasico(Notificador):
    def enviar(self, mensaje):
        return f"Notificación básica: {mensaje}"

# Decorador base
class DecoradorNotificador(Notificador):
    def __init__(self, notificador):
        self.notificador = notificador

    def enviar(self, mensaje):
        return self.notificador.enviar(mensaje)

# Decoradores concretos
class DecoradorSMS(DecoradorNotificador):
    def enviar(self, mensaje):
        return f"{self.notificador.enviar(mensaje)} + enviado por SMS"

class DecoradorEmail(DecoradorNotificador):
    def enviar(self, mensaje):
        return f"{self.notificador.enviar(mensaje)} + enviado por Email"

# Uso
notificador_basico = NotificadorBasico()
notificador_con_sms = DecoradorSMS(notificador_basico)
notificador_completo = DecoradorEmail(notificador_con_sms)

print(notificador_completo.enviar("¡Hola!"))
# Salida: Notificación básica: ¡Hola! + enviado por SMS + enviado por Email
```

File Edit Format Run Options Window Help

```python
# Componente base
class Bebida:
    def obtener_descripcion(self):
        pass

    def calcular_precio(self):
        pass

# Componente concreto
class CafeSimple(Bebida):
    def obtener_descripcion(self):
        return "Café simple"

    def calcular_precio(self):
        return 1000

# Decorador base
class ComplementoBebida(Bebida):
    def __init__(self, bebida):
        self.bebida = bebida

    def obtener_descripcion(self):
        return self.bebida.obtener_descripcion()

    def calcular_precio(self):
        return self.bebida.calcular_precio()
```

```python
# Decoradores concretos
class Leche(ComplementoBebida):
    def obtener_descripcion(self):
        return f"{self.bebida.obtener_descripcion()}, con leche"

    def calcular_precio(self):
        return self.bebida.calcular_precio() + 500

class Azucar(ComplementoBebida):
    def obtener_descripcion(self):
        return f"{self.bebida.obtener_descripcion()}, con azúcar"

    def calcular_precio(self):
        return self.bebida.calcular_precio() + 250

class Canela(ComplementoBebida):
    def obtener_descripcion(self):
        return f"{self.bebida.obtener_descripcion()}, con canela"

    def calcular_precio(self):
        return self.bebida.calcular_precio() + 250
```

Bebidas

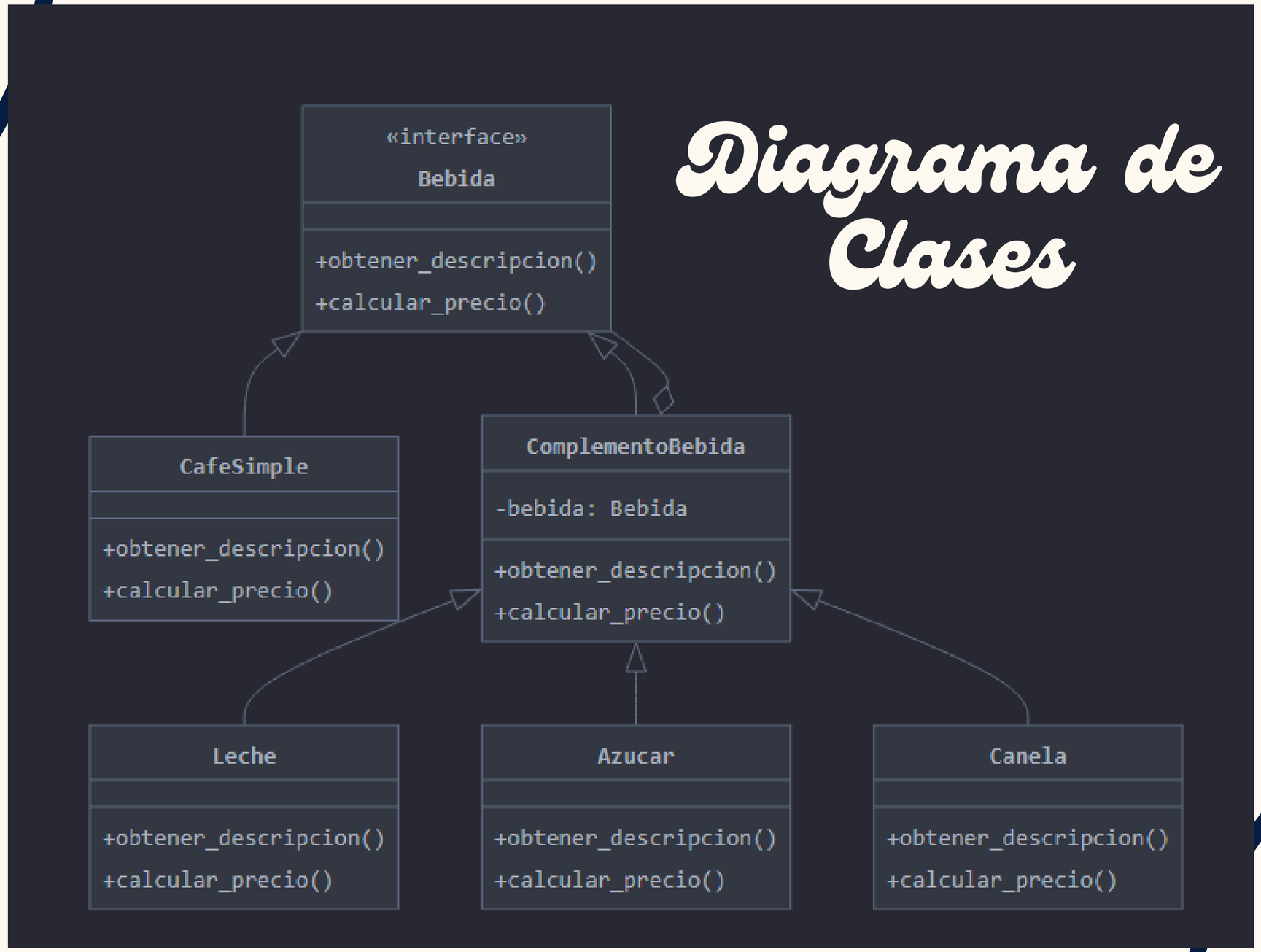
```python
# Uso
mi_cafe = CafeSimple()
print(f"{mi_cafe.obtener_descripcion()} - ${mi_cafe.calcular_precio()}")
# Café simple - $1.000

mi_cafe_con_leche = Leche(mi_cafe)
print(f"{mi_cafe_con_leche.obtener_descripcion()} - ${mi_cafe_con_leche.calcular_precio()}")
# Café simple, con leche - $1.500

mi_cafe_completo = Canela(Azucar(Leche(mi_cafe)))
print(f"{mi_cafe_completo.obtener_descripcion()} - ${mi_cafe_completo.calcular_precio()}")
# Café simple, con leche, con azúcar, con canela - $2.000
```

```
Edit  Shell  Debug  Options  Window  Help
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:4
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for mo
>
========== RESTART: C:/Users/carlo/Desktop/Bebida
Café simple - $1000
Café simple, con leche - $1500
Café simple, con leche, con azúcar, con canela - $2000
>
```

Diagrama de Clases



Gracias