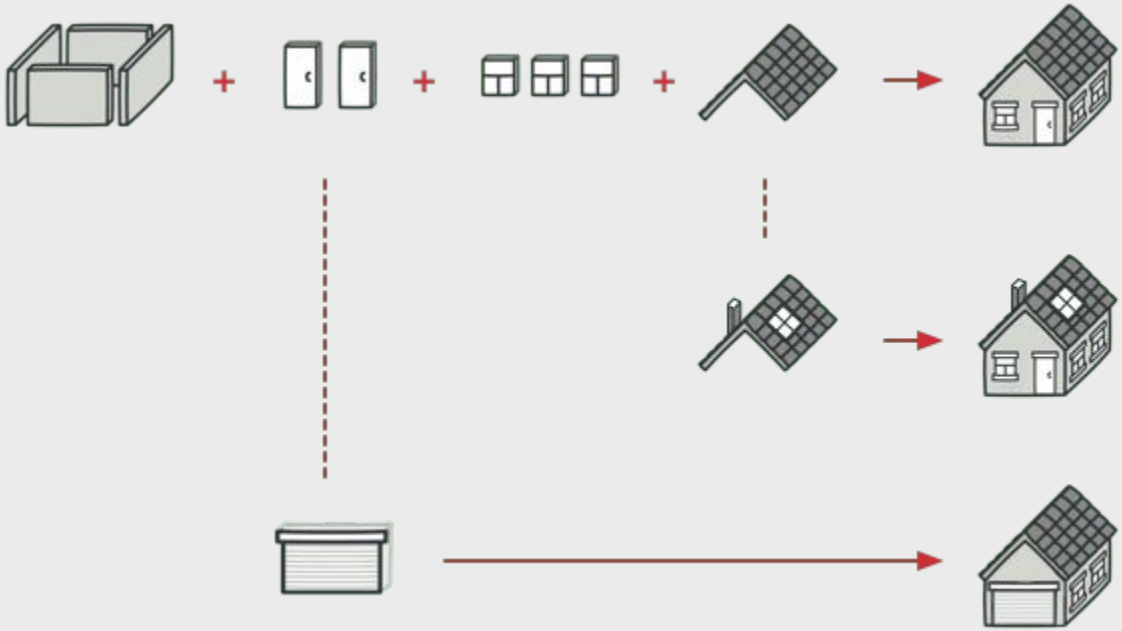
TEMPLATE METHOD

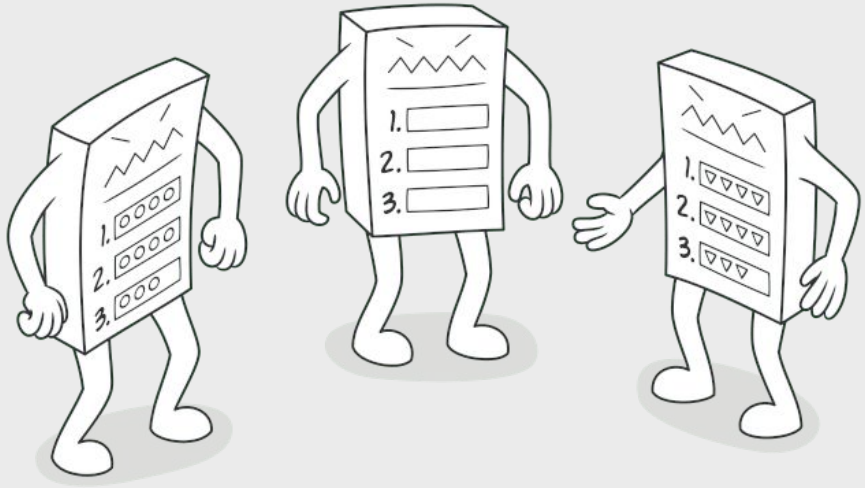
Lizeth Corrales Cortés. C02428
Gabriel González Flores. C03376

ANALOGÍA EN EL MUNDO REAL



CONCEPTO

El Template Method es un patrón de comportamiento en el que se define el esqueleto o la base de un algoritmo en la superclase, pero dando la posibilidad a las subclases que creen sus propias implementaciones del algoritmo sin cambiar su estructura.



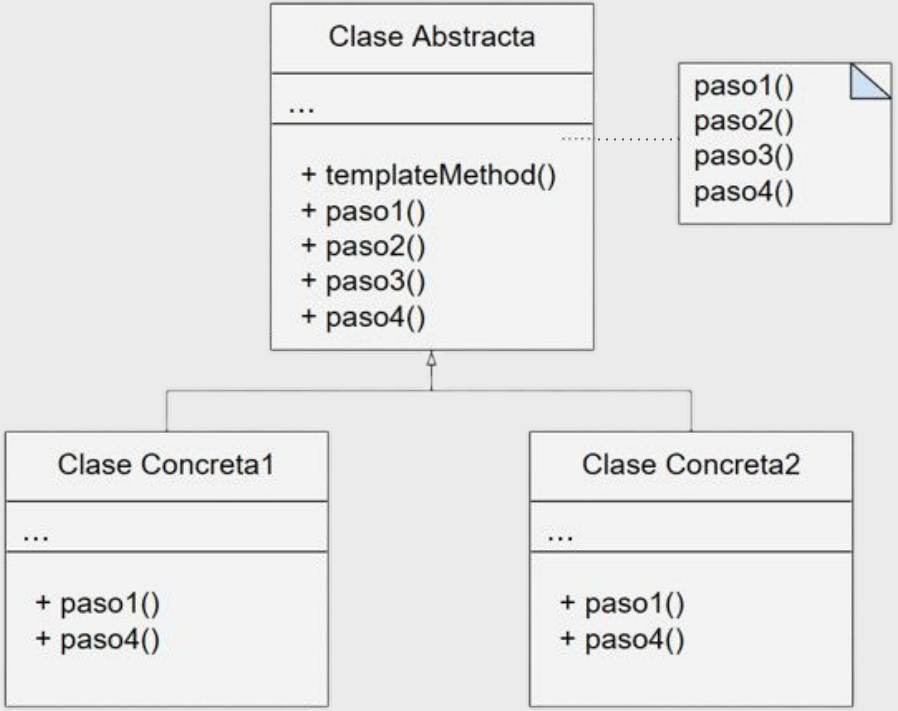
¿CUÁNDO USARLO?

- Cuando se desee que un cliente pueda extender únicamente algunos pasos de un algoritmo, pero siempre siguiendo una misma estructura.

- Cuando se identifique un algoritmo monolítico, por lo que este se separa en varios pasos que se pueden extender en diferentes clases concretas.

- Cuando se identifican muchas clases con algoritmos casi idénticos, pero se diferencian en muy pocos pasos, de modo que se encapsulan dichos pasos a una superclase.

COMPONENTES

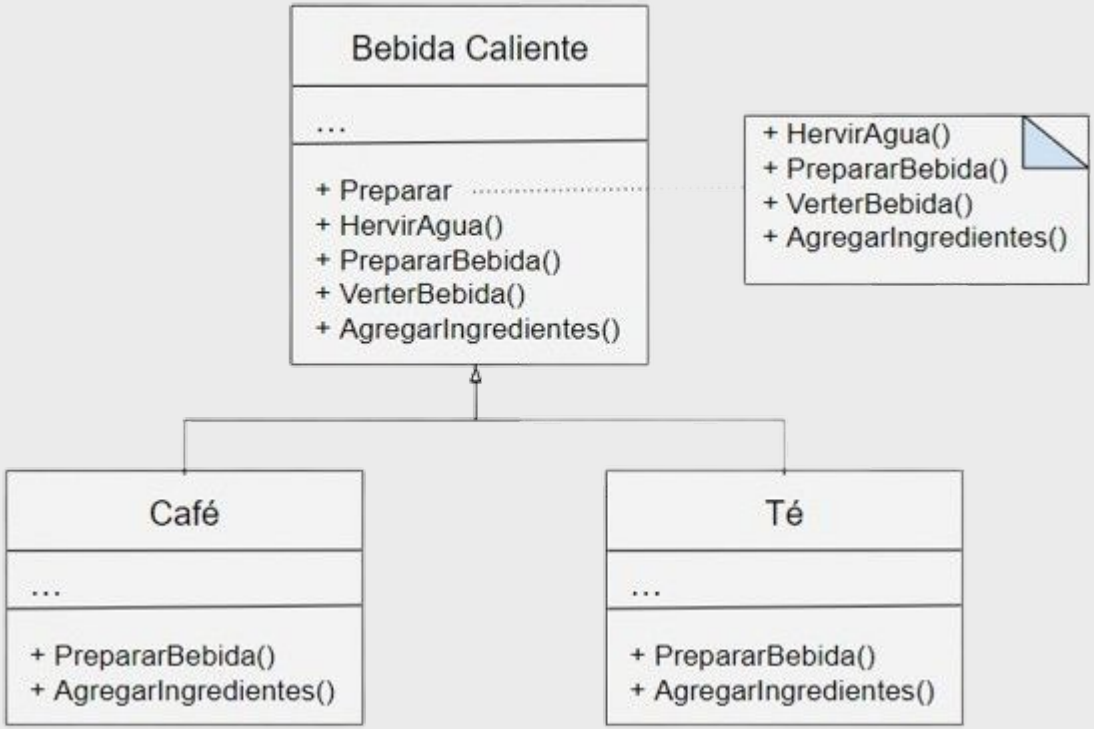


PROBLEMA

Café
...
+ HervirAgua() + PrepararBebida() + VerterBebida() + AgregarIngredientes()

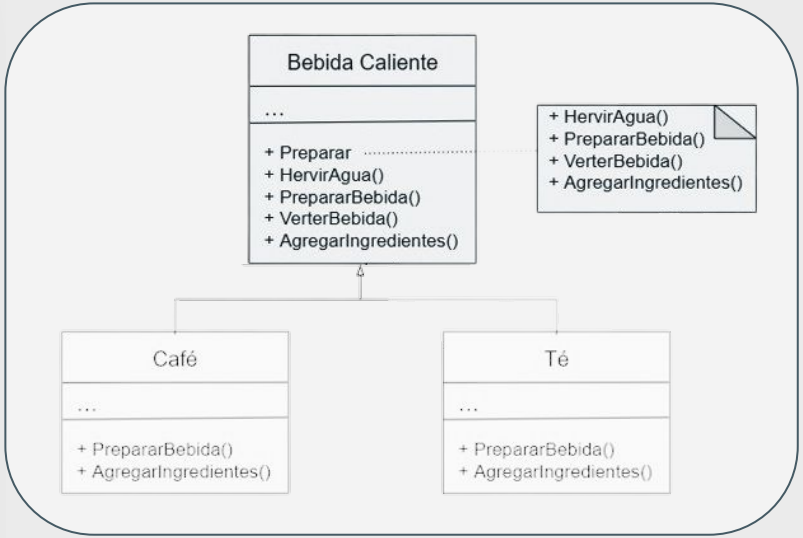
Té
...
+ HervirAgua() + PrepararBebida() + VerterBebida() + AgregarIngredientes()

SOLUCIÓN


Bebida Caliente
...
+ Preparar
+ HervirAgua()
+ PrepararBebida()
+ VerterBebida()
+ AgregarIngredientes()
+ HervirAgua()
+ PrepararBebida()
+ VerterBebida()
+ AgregarIngredientes()
Café
...
+ PrepararBebida()
+ AgregarIngredientes()
Té
...
+ PrepararBebida()
+ AgregarIngredientes()

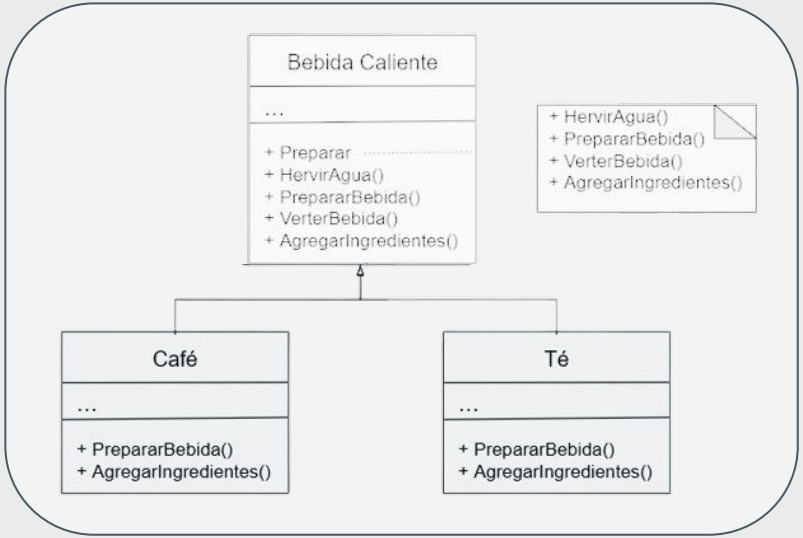
IMPLEMENTACIÓN EN CÓDIGO

```python
from abc import ABC, abstractmethod

  class Bebida(ABC):

    # Método plantilla
    # Define la estructura del algoritmo
    def preparar(self) -> None:
        self.hervir_agua()            # Paso 1
        self.preparar_bebida()        # Paso 2 (Abstracto)
        self.verter_bebida()          # Paso 3
        self.agregar_ingredientes()   # Paso 4 (Abstracto)

    # Estos pasos ya tienen implementaciones.

    def hervir_agua(self) -> None:
        print("Se está hirviendo el agua...")

    def verter_bebida(self) -> None:
        print("Se está vertiendo la bebida...")

    # Estos pasos deben ser implementados por las clases concretas.

    @abstractmethod
    def preparar_bebida(self) -> None:
        pass

    @abstractmethod
    def agregar_ingredientes(self) -> None:
        pass
```

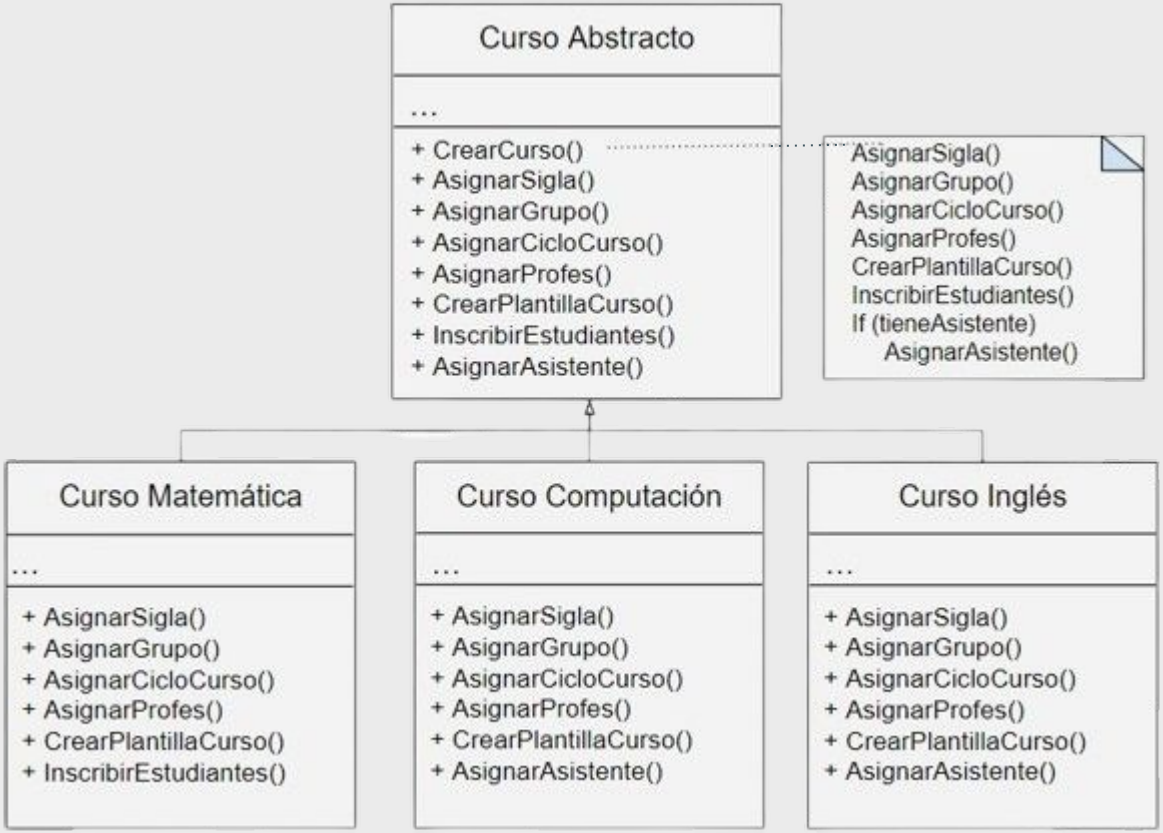


IMPLEMENTACIÓN EN CÓDIGO

```python
class Café(Bebida):

    def preparar_bebida(self) -> None:
        print("Se está preparando el café...")

    def agregar_ingredientes(self) -> None:
        print("Agregando azúcar y leche al café...")

class Té(Bebida):

    def preparar_bebida(self) -> None:
        print("Se están remojando las bolsitas de té...")

    def agregar_ingredientes(self) -> None:
        print("Agregando miel al té...")

def client_code(bebida: Bebida) -> None:
    bebida.preparar()

if __name__ == "__main__":
    print("Se pidió un café")
    client_code(Café())
    print("\n")

    print("Se pidió un té")
    client_code(Té())
    print("\n")
```



EJEMPLO EN MEDIACIÓN VIRTUAL



CONSECUENCIAS

VENTAJAS

- Permite a los clientes sobrescribir tan solo ciertas partes de un algoritmo grande, de modo que se vean menos afectados por los cambios que tienen lugar en otras partes del algoritmo.
- Puede extraer el código duplicado en una superclase.
- Cumple el principio DRY al evitar la duplicación de código.

DESVENTAJAS

- Algunos clientes pueden estar limitados por el esqueleto proporcionado por un algoritmo.
- Puede violar el principio de Liskov Substitución al suprimir la implementación de un paso predeterminado a través de una subclase.
- Los métodos plantilla tienden a ser más difíciles de mantener cuantos más pasos tienen.

IMPLEMENTACIÓN

- Analizar si el algoritmo puede ser dividido en pasos, considerar qué pasos son comunes para todas las clases concretas y cuáles serán únicos.

- Crear la clase base abstracta y declarar el método plantilla dentro de esta clase.

- Para cada variación que se pueda dar dentro del algoritmo, se debe crear una nueva clase concreta que implemente los pasos abstractos y opcionalmente, sobrescribir los pasos definidos de antemano.

RELACIÓN CON OTROS PATRONES

FACTORY METHOD

Es una especialización del Template Method. Al mismo tiempo, un método de fábrica puede servir como un paso en un método de plantilla grande.

STRATEGY

- Semejante al Template Method excepto en su granularidad.
- Template Method utiliza la herencia para variar parte de un algoritmo
- Strategy utiliza la delegación para variar todo el algoritmo.
- Strategy modifica la lógica de los objetos individuales, mientras que Template Method modifica la lógica de una clase completa.

REFERENCIAS

Refactoring.guru. (s.f.) *Bridge.*
https://refactoring.guru/es/design-patterns/bridge

Shvets, A (2019) Dive Into Design Patterns.

ACTIVIDAD

ACTIVIDAD

- La actividad consta de 3 juegos
- En cada juego se les dará una serie de instrucciones que deberán seguir para poder continuar participando.
- En cada juego se poseerán al menos 3 pasos.
- Cada persona debe prestar atención al paso que le corresponde, según su turno.
- Existen pasos opcionales, los cuales pueden decidir si hacerlos o no, en caso de no llevarlos a cabo deben realizar el siguiente paso obligatorio (Repetir)
- Si se equivocan de paso o tardan en responder, salen del juego,

DIME SU NOMBRE

Para esta actividad deben seguir el siguiente patrón:

- Paso 1: Diga el nombre de las persona que se encuentra a la derecha
- Paso 2: Diga el nombre de las persona que se encuentra a la izquierda
- Paso 3: Diga su propio
- Repita los 3 nombres anteriores

DIME LAS VOCALES (1)

Para esta actividad deben seguir el siguiente patrón:

- Paso 1: Diga una palabra que inicia con la vocal a
- Paso 2: Diga una palabra que inicia con la vocal e
- Paso 3: Diga una palabra que inicia con la vocal i
- Repita 3 palabras

DIME LAS VOCALES (2)

Para esta actividad deben seguir el siguiente patrón:

- Paso 1: Diga una palabra que inicia con la vocal **u**
- Paso 2: Diga una palabra que inicia con la vocal **o**
- Paso 3: Diga una palabra que inicia con la vocal **i**
- Paso 4: Diga una palabra que inicia con la vocal **e**
- Paso 5: Diga una palabra que inicia con la vocal **a**
- Repita las 5 palabras

FORMEMOS ORACIONES

Para esta actividad deben seguir el siguiente patrón:

- Paso 1: Diga un **artículo**
- Paso 2: Diga un **sustantivo**
- Paso 3: Diga un **adjetivo**
- Paso 4: Diga un **verbo**
- Repetir