**Taller 5**

• Información general del proyecto: para qué sirve, cuál es la estructura general del diseño, qué grandes retos de diseño enfrenta (i.e. ¿qué es lo difícil?). Deben incluir la URL para consultar el proyecto.

El proyecto contiene ejemplos del uso del Template Method, en que se programa el proceso de hacer algunos tipos de casa por ejemplo una casa de madera o una casa de concreto, la estructura general del programa está compuesta por un folder src que contiene los siguientes package:

example.afterrefactoring: Que contiene la clase abstracta HouseTemplate y las clases ConcreteHouse, Glass House y WoodenHouse.

example.afterrrefactoring.clientcode: Que contiene la clase main y que sería la interfaz por consola del programa

También contiene package sin el uso del Template Method en el programa, lo que nos ejemplifica mejor el porque el uso del método nos cambiaria bastante su optimización.

example.beforerefactoring: Que contiene la clase HouseTemplate

example.beforerefactoring.clientcode: Que contiene la clase HouseClient

Los retos que este puede enfrentar: Como hay bastantes tipos de casas, cada una tendría un método, pero cada vez que queramos agregar otra casa esto podría hacer nuestro proyecto más propenso a errores, además la clase se hace cada vez más grande, además si quisiera innovar el método de hacer las casas tendría que modificarlo uno por uno.

URL del proyecto: <https://github.com/exzhou/ocp-templatemethod-lab.git>

• Información y estructura del fragmento del proyecto donde aparece el patrón. No se limite únicamente a los elementos que hacen parte del patrón: para que tenga sentido su uso, probablemente va a tener que incluir elementos cercanos que sirvan para contextualizarlo.

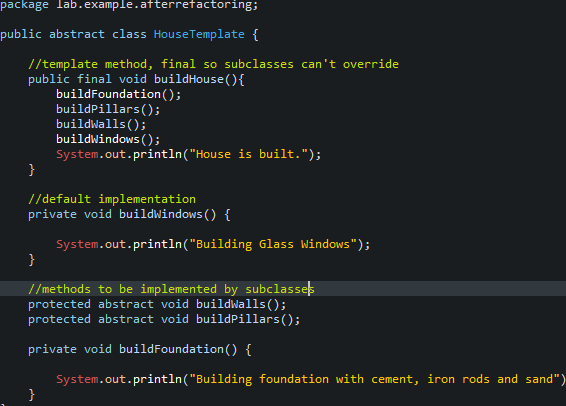
La implementación del Template Method en el proyecto se usa para implementar de la clase base en este caso la clase HouseTemplate ciertos métodos sin cambiar su estructura general en las subclases que en este caso son las clases que son los tipos de casas: ConcreteHouse etc., esto permite que las subclases especifiquen los métodos sin que la clase base dependa de ellas y que en caso de que requiera agregar mas subclases no tenga que modificar esa clase base.

• Información general sobre el patrón: qué patrón es y para qué se usa usualmente.

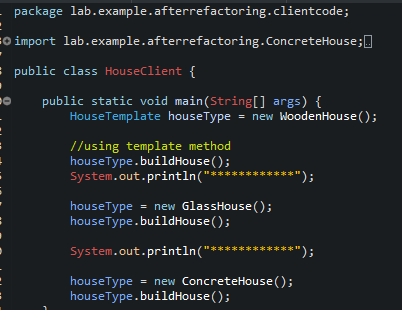
Es un patrón de diseño que pertenece a la categoría de patrones de comportamiento. Su estructura se define en el uso de una clase base y las subclases implementen o personalicen ciertos pasos específicos del algoritmo sin cambiar su estructura general.

Usualmente es utilizado para cuando tienes un algoritmo que sigue una estructura fija, pero permite variaciones en ciertos pasos de este. Al utilizar este patrón, puedes evitar la duplicación de código y centralizar la lógica común en la clase base, mientras que las subclases pueden implementar las partes específicas según sea necesario.

• Información del patrón aplicado al proyecto: explicar cómo se está utilizando el patrón dentro del proyecto.

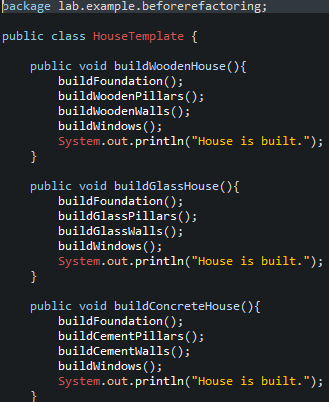
La clase abstracta HouseTemplate proporciona una estructura general para construir una casa utilizando el patrón de diseño "Template Method". Define un método principal "buildHouse()" que sigue una secuencia fija de pasos, llamando a métodos auxiliares para construir los cimientos, pilares, paredes y ventanas de la casa. Los métodos "buildPillars()" y "buildWalls()" deben ser implementados por las subclases, mientras que "buildFoundation()" y "buildWindows()" tienen implementaciones predeterminadas que pueden ser personalizadas si es necesario.

Luego en la clase HouseClient se hace implementación del Template Method para la construcción de cada casa.



• ¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto? ¿Qué ventajas tiene?

La primer ventaja es sin duda evitar sobrescribir código, ya que se pueden tomar atributos que tienen los objetos en común como en este caso todas las casas sin importar de que estén hechas tienen ventanas y cimientos. Además, su estructura va a seguir siendo la misma y si quisiera agregar otro tipo de casa no tendría que hacer muchos cambios en la clase base

Texto

Descripción generada automáticamente

Es importante tener en cuenta que en el proyecto hay una versión en la que no se hace uso del Template Method, como se pueden ver en las imágenes hay notable cambio, ya que hay más código y además que si quiera agregar una característica general de todas las casas tendría que cambiar todos los métodos

• ¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto?

El Template Method requiere una estrecha cooperación entre la clase base y las subclases, lo que puede llevar a un mayor acoplamiento. Esto significa que los cambios en la clase base pueden tener un impacto en las subclases , lo que puede dificultar la evolución independiente de las clases. En el caso del proyecto si en algún momento quiero cambiar algún aspecto de la clase base por ejemplo agregar tejas de un solo material las subclases de las casas se pueden ver afectadas.

• ¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los

problemas que resuelve el patrón?

En lugar de utilizar un método principal con llamadas a métodos auxiliares, se pueden utilizar métodos polimórficos en las clases base y subclases para permitir la implementación y personalización específica de cada paso del algoritmo. Esto puede lograrse utilizando interfaces y herencia, lo que permite una mayor flexibilidad y adaptabilidad.