Práctica 2: Implementación de los patrones de diseño: Iterator, State y Template para la resolución de un problema

UNAM, Facultad De Ciencias.

Curso: Modelado y Programación
Profesor.Rosa Victoria Villa Padilla
Ayudante. Lab. Arturo Lemus Pablo
Ayudante. Fernando López Balcazar
Ayudante. Itzel Azucena Delgado Díaz

Equipo: Prietos en aprietos

SanMartin Macias Juan Daniel No. Cuenta 318181637
López Diego Gabriela No. Cuenta 318243485
Rivera Zavala Javier Alejandro No. Cuenta 311288876

19 de septiembre de 2022.
Ciudad de México.

1. Patrón de diseño State

La idea del patrón STATE es permitir a un objeto alterar su comportamiento dependiendo de su estado interno en cada momento. En otras palabras, encapsula el comportamiento de un objeto dependiendo del estado en el que se encuentre.
Dicho ESTADO lo definimos como el conjunto de características que harán que el objeto tenga características concretas. Y dado un estado, el objeto puede o no cambiar a otro estado. Estas normas de cambio, las llamamos *transiciones* que además, son finitas y predeterminadas.
Para poder implementar State realizamos lo que se señala a conntinuación

1. Creamos una clase CONTEXTO donde almacenaremos una referencia a uno de los objetos de estado concreto. Aquí nos comunicamos con el objeto de estado por medio de la interfaz ESTADO.
2. Creamos una interfaz ESTADO donde declaramos los metodos(comportamientos) específicos del estado.
3. Creamos clases para cada uno de los estados del objeto y que implementen de nuestra interfaz ESTADO. En cada clase ESTADO, se proporcionan implementaciones propias para cada metodo.
   ES decir,

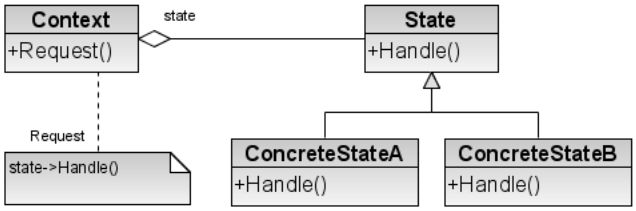

Figura 1: Diagrama UML patron State

Desventajas
Con este patrón podemos llegar a crear una cantidad considerable de clases y objetos en el programa, haciendo nuestro código menos manejable y por ende, mas difícil su mantenimiento.

2. Patrón de diseño Template

Template es un patrón de comportamiento. Aquí definimos el esqueleto de un algoritmo en una superclase donde definimos una o varias operaciones concretas en forma de métodos abstractos que son usados por el método plantilla y que nos permite que las subclases sobrescriban pasos del algoritmo sin cambiar su estructura. De esta manera, evitamos codigo duplicado(o reutilizamos codigo). Además una de las ventajas de este patron es que tiene puntos de extensión y notificación, conocidos como hooks, que pueden ser implementados en caso de ser necesario.
Nos es útil cuando diferentes entidades tienen un comportamiento similar pero que difiere en determinados aspectos.

Desventajas
Una vez que la estructura del algoritmo está establecido y tenemos algunas subclases funcionando es muy complicado modificar el algoritmo, ya que cualquier cambio nos obligaría a modificar todas las subclases.

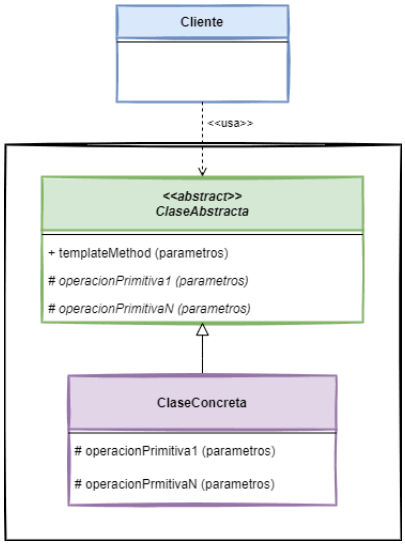

Figura 2: Diagrama UML patron Template

3. Patrón de diseño Iterator

Iterator también es un patrón de diseño de comportamiento que te permite recorrer elementos de una colección sin conocer su estructura interna (lista, pila, árbol, etc.). Aquí extraemos el comportamiento de recorrido de una colección y lo colocamos en un objeto llamado iterador. Dicho objeto nos proporciona los métodos necesarios para recorrer los elementos de la estructura de datos, y donde encapsula todos los detalles del recorrido, como la posición actual y cuántos elementos quedan hasta el final. Los metodos más usados son:
hasNext: Método que regresa un booleano para indicar si existen más elementos en la estructura por recorrer.
next: Regresa el siguiente elemento de la estructura de datos.
remove: elimina algun elemento en la estructura.

Debido a esto, varios iteradores pueden recorrer la misma colección al mismo tiempo, independientemente los unos de los otros. Y como todos los iteradores implementan de la misma interfaz, el código cliente es compatible con cualquier tipo de colección.

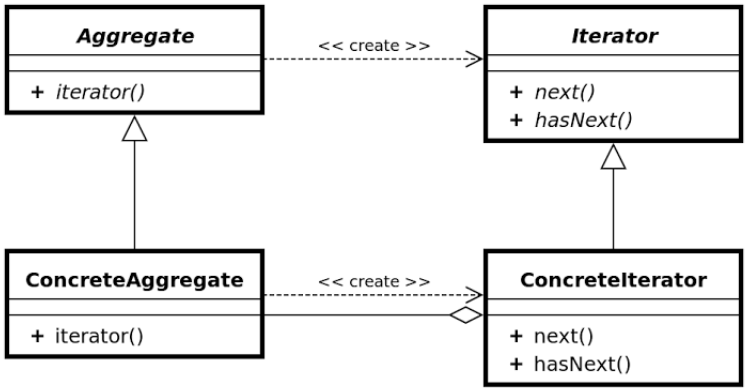

Figura 3: Diagrama UML patron Iterator

Desventajas
Las implementaciones típicas de Iterator tienen un comportamiento no definido cuando la colección llega a ser modificada mientras se está iterando.
Por ultimo,
Para compilar la practica 02 escribimos lo siguiente en terminal

javac *.java

Y para ejecutarlo

java Main