Práctica 02
Modelado y Programación

Lugo Cano Daniel 315320015
Méndez Gallegos Ligia Natalia 314641250

- **State**

Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. Busca que un objeto pueda reaccionar según su estado interno. Si bien muchas veces esto se puede solucionar con un boolean o utilizando constantes, esto suele terminar con una gran cantidad de if-else, código ilegible y dificultad en el mantenimiento. La intención del State es desacoplar el estado de la clase en cuestión.

UML
Context: mantiene una instancia con el estado actual.
State: define interfaz para el comportamiento asociado a un determinado estado del contexto.
StateConcreto: cada subclase implementa el comportamiento asociado con un estado del contexto.

Desventajas
Se incrementa el número de sus clases.
El patrón no indica exactamente dónde definir las transiciones de un estado a otro.

- **Iterator**

Provee un mecanismo estándar para acceder secuencialmente a los elementos de una colección; define una interface que declara métodos para acceder secuencialmente a los objetos de una colección. Una clase accede a una colección a través de dicha interface.

UML
Agregado/Aggregate: define una interfaz para crear un objeto iterator.
Iterator: define la interfaz para acceder y recorrer los elementos de un agregado.
IteradorConcreto: implementa la interfaz del iterador y guarda la posición actual del recorrido en cada momento.
AgregadoConcreto: implementa la interfaz de creación de iteradores devolviendo una instancia del iterador concreto apropiado.
Cliente: solicita recorrer una colección y lo hace siguiendo los métodos dados por la interfaz Iterator.

Desventajas

Si necesitas actualizar la estructura que estás iterando, mientras estás iterando, no se puede por la forma en la que el iterador guarda su posición.

- **Template**

Define una estructura algorítmica cuya lógica quedará a cargo de las subclases. Para ello, escribe una clase abstracta que contiene parte de la lógica necesaria para realizar su finalidad. En ella se define una estructura de herencia que sirve de plantilla ("Template" significa plantilla) de los métodos en las subclases.

UML
AbstractTemplate o AbstractClass: implementa un método plantilla que define el esqueleto de un algoritmo y define métodos abstractos que deben implementar las subclases concretas
TemplateConcreto o ConcreteClass: implementa los métodos abstractos para realizar los pasos del algoritmo que son específicos de la subclase.

Desventajas
Seguir la secuencia de las clases puede ser difícil de entender.
A veces es redundante implementar los métodos, pueden diferir de algo muy pequeño como para ponerlo con un if,

Creemos que la práctica está bien implementada pero fue un poco difícil establecer qué clases serían los patrones diferentes. Faltó optimizarla y quitar los métodos que no se necesitaran.
Para el patrón Template usamos la clase abstracta Hamburguesa y la extienden HamburguesaCarne y HamburguesaVegetariana. Usar al robot como patrón State era muy claro y para el iterador fue el recorrer las diferentes estructuras.