



Patron de comportamiento

MEMENTO

Por

Mariana Urrego

Natalia Urrego



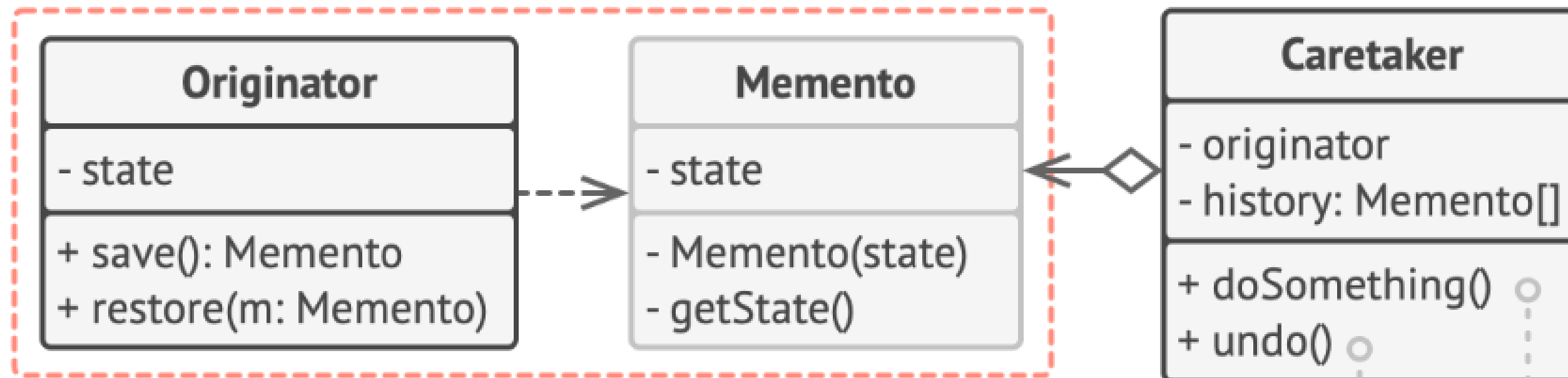
También llamado: Recuerdo, Instantánea, snapshot

Es un patrón de diseño de comportamiento que te permite guardar y restaurar el estado previo de un objeto sin revelar los detalles de su implementación.

SE COMPONE DE TRES ELEMENTOS PRINCIPALES:

1. Originator (Originador): Es el objeto cuyo estado queremos guardar y restaurar.
2. Memento: Es un objeto que almacena el estado interno del Originator en un momento determinado.
3. Caretaker (Guardián o Cuidador): Administra los Mementos y decide cuándo guardarlos o restaurarlos.

Diagrama

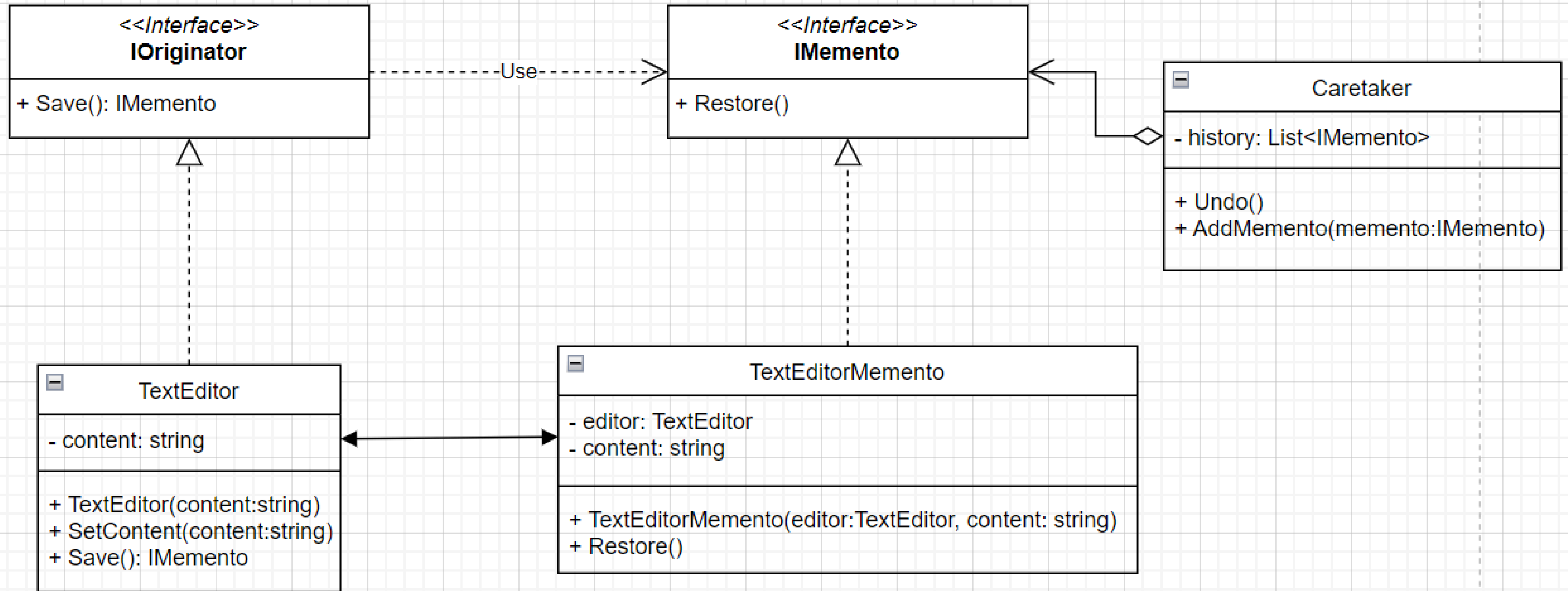


En esta implementación, la clase memento se anida dentro de la originadora. Esto permite a la originadora acceder a los campos y métodos de la clase memento, aunque se declaren privados. Por otro lado, la cuidadora tiene un acceso muy limitado a los campos y métodos de la clase memento, lo que le permite almacenar mementos en una pila pero no alterar su estado.

`m = history.pop()`
`originator.restore(m)`

`m = originator.save()`
`history.push(m)`
`// originator.change()`

Diagrama de Ejemplo



Cuando usarlo

1

Cuando necesitas una funcionalidad de deshacer y rehacer en tu aplicación.

2

Cuando deseas tomar instantáneas del estado de un objeto sin violar su encapsulación.

3

Cuando quieres evitar la dependencia directa entre objetos que manejan el estado y los que lo restauran.

Situaciones del día a día

Funcionalidad de "Deshacer" en un Editor de Texto

- Cada vez que el usuario escribe o borra texto, el sistema guarda un Memento del contenido anterior.
- Si el usuario presiona Ctrl + Z (Deshacer), el sistema restaura el último estado guardado.


Videojuegos


- Cuando un jugador guarda su progreso, el juego crea un Memento con la información del personaje (vida, posición, objetos).
- Si el jugador carga la partida, el juego restaura ese estado.

Transacciones de bancos

- Un sistema bancario guarda un memento del saldo de una cuenta antes de una transferencia. Si ocurre un error (como conexión fallida o saldo insuficiente), el sistema usa el memento para restaurar el saldo original.

Pros

 Puedes producir instantáneas del estado del objeto sin violar su encapsulación.

 Puedes simplificar el código de la originadora permitiendo que la cuidadora mantenga el historial del estado de la originadora.

Contras



La aplicación puede consumir mucha memoria RAM si los clientes crean mementos muy a menudo.



Las cuidadoras deben rastrear el ciclo de vida de la originadora para poder destruir mementos obsoletos.



La mayoría de los lenguajes de programación dinámicos, como PHP, Python y JavaScript, no pueden garantizar que el estado dentro del memento se mantenga intacto.



Bibliografía

<https://refactoring.guru/es/design-patterns/memento>

<https://www.youtube.com/watch?v=VCZK62hXz3E>

<https://refactoring.guru/es/design-patterns/memento/csharp/example#:~:text=Memento%20es%20un%20patr%C3%B3n%20de,encuentra%20dentro%20de%20las%20instancias.>