|  |  |  |  |
| --- | --- | --- | --- |
| NOMBRE | Diseño de sistemas | | |
| AREA | Tecnologías Aplicadas | REGIMEN | Semestral |
| MODALIDAD | Por examen final | CARGA HORARIA | |  | | --- | | 6 (seis) semanales,  90 totales | |

Se entrega la práctica en formato digital y papel. El formato digital debe tener el mismo encabezado que el presente texto. El formato en papel debe contener fecha, materia, curso, datos del alumno.

***Alumno:***

***Curso:***

***Fecha de entrega:***

**Tema: Patrones de diseño**

**Memento**

Este patrón de diseño permite capturar y exportar el estado interno de un objeto para que luego se pueda restaurar, sin romper la encapsulación.

Su finalidad es almacenar el estado de un objeto (o del sistema completo) en un momento dado, de manera que se pueda restaurar posteriormente si fuese necesario. Para ello se mantiene almacenado el estado del objeto para un instante de tiempo en una clase independiente de aquella a la que pertenece el objeto (pero sin romper la encapsulación), de forma que ese recuerdo permita que el objeto sea modificado y pueda volver a su estado anterior.

Hoy en día, muchos aplicativos permiten el "deshacer" y "rehacer" de manera muy sencilla. Para ciertos aplicativos es casi una obligación tener estas funciones y sería impensado el hecho que no las posean. Sin embargo, cuando queremos llevar esto a código puede resultar complejo de implementar. Este patrón intenta mostrar una solución a este problema.

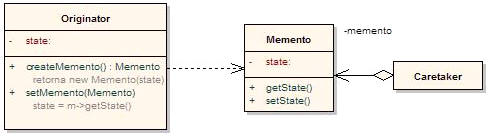
Se usa cuando:

* Se necesite restaurar el sistema desde estados pasados.
* Se quiera facilitar el hacer y deshacer de determinadas operaciones, para lo que habrá que guardar los estados anteriores de los objetos sobre los que se opere (o bien recordar los cambios de forma incremental).
* Este patrón debe ser utilizado cuando se necesite salvar el estado de un objeto y tener disponible los distintos estados históricos que se necesiten. Por ello mismo, este patrón es muy intuitivo para darse cuando debe ser utilizado.

Hoy en día una gran variedad de aplicaciones poseen las opciones de "deshacer" y "rehacer". Por ejemplo, las herramientas de Microsoft Office como Word, Power Point, etc. Es imposible pensar que ciertas herramientas no tengan esta opción, como el Photoshop. También IDEs de programación como Eclipse utilizan una opción de historial local. Una solución para este problema es el patrón Memento.

**Diagramas UML**

**Vista estática: diagrama de clases**



Caretaker: es responsable por mantener a salvo a Memento. No opera o examina su contenido.

Memento: almacena el estado interno de un objeto Originator. El Memento puede

almacenar todo o parte del estado.

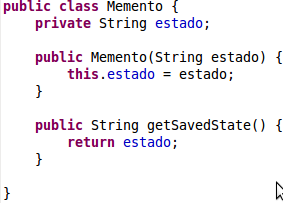
Originator: crea un objeto Memento conteniendo una fotografía de su estado

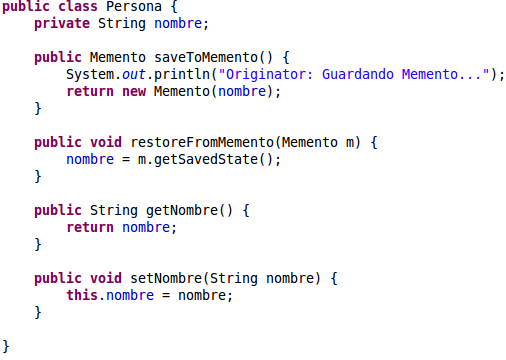
interno

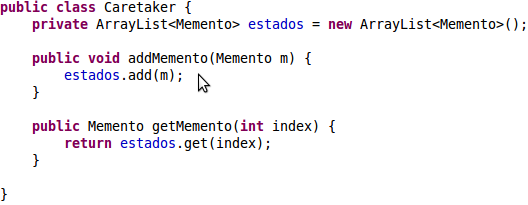
**Vista Dinámica: Diagrama de secuencia**

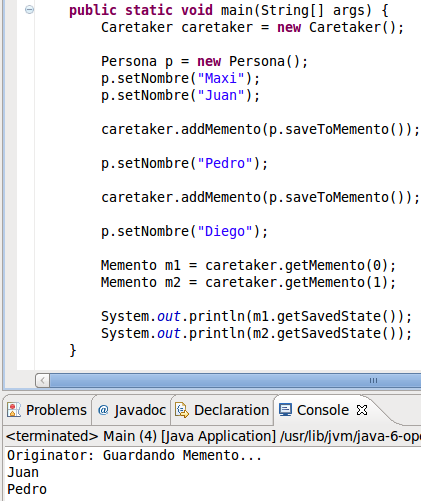
***Caso práctico a desarrollar en clase***

Vamos a realizar un ejemplo de este patrón donde se busque salvar el nombre de una persona que puede variar a lo largo del tiempo.









**Consecuencias**

No es necesario exponer el estado interno como atributos de acceso público, preservando así la encapsulación.

Si el originador tendría que de almacenar y mantener a salvo una o muchas copias de su estado interno, sus responsabilidades crecerían y sería inmanejable.

El uso frecuente de Mementos para almacenar estados internos de gran tamaño, podría resultar costoso y perjudicar la performance del sistema.

Caretaker no puede hacer predicciones de tiempo ni de espacio.

**Temas a tener en cuenta.**

Si bien la implementación de un Memento no suele variar demasiado, cuando la secuencia de creación y restauración de mementos es conocida, se puede adoptar una estrategia de cambio incremental: en cada nuevo memento sólo se almacena la parte del estado que ha cambiado en lugar del estado completo.

Esta estrategia se aplica cuando memento se utiliza para mantener una lista de deshacer/rehacer.

Otra opción utilizada es no depender de índices en la colecciones y utilizar ciertos métodos no indexados como el .previus() que poseen algunas colecciones.